UNIVERSITÀ DI PISA

Scuola di Dottorato in Ingegneria "Leonardo da Vinci"

Corso di Dottorato di Ricerca in
INGEGNERIA DELL'INFORMAZIONE

Tesi di Dottorato di Ricerca

LARGE-SCALE NETWORKS: ALGORITHMS, COMPLEXITY AND REAL APPLICATIONS

Autore:

Simone Mainardi

Relatori:

Prof. *Luciano Lenzini*

Ing. *Enrico Gregori*

Prof. *Enzo Mingozzi*

Anno 2014
SSD ING-INF/05

Ai miei genitori, Gabriella e Luciano, e a Giulia
per me fonte continua di forza e perseveranza.

Acknowledgement. This thesis would not have been possible without the support of many people over the years. I take advantage of this opportunity to thank some of them.

First and foremost, I would like to thank my advisors Prof. Luciano Lenzini and Enrico Gregori for their constant encouragement and support. They have generously provided me the best resources to study, learn and do research.

My sincere thanks go to Prof. Luca Deri for his openness, advice, patient indulgence and unwavering belief that I can do anything if I want to enough.

My gratitude goes to my colleagues at the Institute for Informatics and Telematics (IIT) of the Italian National Research Council (CNR). They have made the office a pleasant and stimulating place to work.

Last but not the least, I would like to say thank you to my family for the unbending love and support given me throughout all the phases of my life. Nothing I have done would have been possible without my family.

Sommario. Le reti hanno ampia applicabilità ai sistemi reali grazie alla loro capacità di modellare e rappresentare relazioni complesse. La scoperta e la previsione di schemi e regolarità nei sistemi, rese possibili grazie alle reti, sono al centro dell'intelligenza e delle capacità analitiche di scienza, industrie e governi. Scoperte e previsioni, in particolar modo oggi nell'era dei *big-data*, non possono prescindere da algoritmi in grado di processare reti massive in modo veloce ed efficiente.

Gli algoritmi per reti massive sono il primo filone di ricerca che sviluppiamo in questa tesi. In particolare, progettiamo, analizziamo teoricamente e implementiamo algoritmi efficienti e paralleli dimostrando rigorosamente le loro complessità di tempo e di spazio. I nostri contributi principali in questo filone sono algoritmi innovativi e paralleli per l'estrazione delle k-clique community, particolari gruppi, all'interno delle reti, ampiamente utilizzati per studiare fenomeni complessi. Gli algoritmi proposti hanno una complessità di spazio che è la radice quadrata di quella dell'attuale stato dell'arte. La complessità di tempo raggiunta è ottimale, ossia inversamente proporzionale al numero di unità elaborative disponibili agli algoritmi. Per confermare l'efficienza degli algoritmi proposti, conduciamo un' esaustiva serie di esperimenti anche in relazione allo stato dell'arte esistente. Misuriamo sperimentalmente uno speedup lineare, che convalida le performance ottimali raggiunte.

Il secondo filone di ricerca sviluppato riguarda l'applicazione delle reti per la comprensione di sistemi reali. Inizialmente proponiamo metodologie innovative per individuare le correlazioni all'interno di reti che evolvono nel tempo. Dopodiché istanziamo queste metodologie per lo studio di Internet, uno dei sistemi tecnologici moderni più diffusi. In particolare, investighiamo le dinamiche della connettività tra le aziende operanti nel settore Internet, le quali si interconnettono al fine di assicurarne il funzionamento globale. Successivamente, combinando le dinamiche di connettività con le quotazioni delle stesse aziende sui mercati finanziari di tutto il mondo, troviamo che aziende geograficamente vicine e con portfolio servizi simili sono guidate dagli stessi fattori economici. Forniamo inoltre prove riguardo all'esistenza e alla natura dei fattori che governano le dinamiche della connettività Internet. In conclusione, proponiamo modelli di rete per studiare il traffico del sistema dei nomi Internet, noto come Domain Name System (DNS). Al fine di mostrare l'efficacia di questi modelli, li impieghiamo per ottenere classificazioni dei domini Internet e per individuare possibili attività anomale o illegittime.

Abstract. Networks have broad applicability to real-world systems, due to their ability to model and represent complex relationships. The discovery and forecasting of insightful patterns from networks are at the core of analytical intelligence in government, industry, and science. Discoveries and forecasts, especially from large-scale networks commonly available in the *big-data* era, strongly rely on fast and efficient network algorithms.

Algorithms for dealing with large-scale networks are the first topic of research we focus on in this thesis. We design, theoretically analyze and implement efficient algorithms and parallel algorithms, rigorously proving their worst-case time and space complexities. Our main contributions in this area are novel, parallel algorithms to detect k-clique communities, special network groups which are widely used to understand complex phenomena. The proposed algorithms have a space complexity which is the square root of that of the current state-of-the-art. Time complexity achieved is optimal, since it is inversely proportional to the number of processing units available. Extensive experiments were conducted to confirm the efficiency of the proposed algorithms, even in comparison to the state-of-the-art. We experimentally measured a linear speedup, substantiating the optimal performances attained.

The second focus of this thesis is the application of networks to discover insights from real-world systems. We introduce novel methodologies to capture cross correlations in evolving networks. We instantiate these methodologies to study the Internet, one of the most, if not the most, pervasive modern technological system. We investigate the dynamics of connectivity among Internet companies, those which interconnect to ensure global Internet access. We then combine connectivity dynamics with historical worldwide stock markets data, and produce graphical representations to visually identify high correlations. We find that geographically close Internet companies offering similar services are driven by common economic factors. We also provide evidence on the existence and nature of hidden factors governing the dynamics of Internet connectivity. Finally, we propose network models to effectively study the Internet Domain Name System (DNS) traffic, and leverage these models to obtain rankings of Internet domains as well as to identify malicious activities.

Contents

List of Figures

List of Tables

Nomenclature

Roman Symbols

$\mathbf{A}$	The adjacency matrix of a network
$a_{i,j}$	Element located at row i (column j) of a matrix $\mathbf{A}$
$\mathbf{B}$	The adjacency matrix of a bipartite network
$\mathbf{C}$	The matrix of bivariate Pearson's correlation coefficients
c	A clique (complete graph) of a network
cc_n	The clustering coefficient of a node n
co_n	The coreness of a node n
d_n	The number of edges with node n as an endpoint (degree)
E	The set of links (or edges, or entities) in a network
x_n	The eigenvector centrality of a node n
F	A collection of disjoint sets
f_k	Degree frequency, i.e., number of nodes having degree equal to k
G	A network
knn_n	The average degree of neighbors of a node n
l	The number of maximal cliques in a network
L_k	The number of maximal cliques with size at least k in a network
l_k	The number of maximal cliques with size equal to k in a network
n	An generic node (or vertex, or entity) in a network
$N(n)$	The set of neighbors of a node n
P_n	Feature P of a node n (e.g., the degree d_n)
V	The set of nodes (or vertices, or entities) in a network

Other Symbols

α	Functional inverse of Ackermann's function
$\wedge$	Logical *and*
μ	The arithmetic mean (average) of a collection of numbers
$\mid s \mid$	The cardinality (number of elements) of a set s
$\binom{h}{k}$	The binomial coefficient
$s_1 \cup s_2$	Union of two sets s_1 and s_2
$\lfloor a \rfloor$	Largest previous integer of number a
$a \leftarrow b$	Algorithmic assignment: a get the value of b
$\vee$	Logical *or*
$r \in s$	Element r is an element of set s
σ	The standard deviation of a collection of numbers
$s_1 \subseteq s_2$	Zero to $\mid s_1 \mid$ elements of set s_1 are also elements of set s_2
σ^2	The variance of a collection of numbers
$s_1 \times s_2$	Cartesian product of two sets s_1 and s_2

Acronyms

AS	Autonomous System
BGP	Border Gateway Protocol
CCDF	Complementary Cumulative Distribution Function
ccTLD	country code Top-Level Domain
cdn	Content Distribution Network
CONNECT_ME	Connected Components Merging
COS	Clique Percolation Method on Steroids
CPM	Clique Percolation Method
CPU	Central Processing Unit
GCE	Greedy Clique Expansion
GPU	Graphics Processing Unit
IP	Internet Protocol
IXP	Internet eXchange Point
MST	Minimum Spanning Tree
OSLOM	Order Statistics Local Optimization Method
POP	Point of Presence

RTT	Round Trip Time
SCP	Sequential Clique Percolation
s.t.	Such That
t1tp	Tier-1 Transit Provider
tp	Transit Provider
TTL	Time To Live

1

Introduction

"Whole compounded of several parts or members"
Definition of $\sigma\acute{\nu}\sigma\tau\eta\mu\alpha$, the greek root of the word **system**.
— LIDDELL AND SCOTT, A Greek-English Lexicon

"A large system consisting of many similar parts that are connected together to allow movement or communication between or along the parts or between the parts and a control centre"
Definition of the word **network**.
— Cambridge Advanced Learner's Dictionary

1.1 Motivation

1.1.1 Large-Scale Networks to Extract Knowledge from Real-World Systems

Networks are of paramount importance to model the relationships between entities of a system in order to gain new knowledge. Currently, networks find application in many heterogeneous fields including finance [25] [141], management [78], technology [179] [21], communication [196], and common social interactions [203], among others. Networks have had direct implications on the ability of human beings to understand the structure and function of real systems, which has benefited all areas of society including medicine [188] [95], public safety [203], critical services for the population [193] [105] [159], business [6], and global markets [100] [26], along with other matters.

Fundamentally, a network is an abstract model of a system, in which entities and relationships are represented with *nodes* and *links*. If two entities have some kind of relationship in the system, then their nodes are joined together with a link in the net-

work. This introductory definition, which might appear fairly abstract at a first glance, is fundamentally flexible and effective. Indeed, it does not matter if we are interested into the intricate web of interdependencies between banks, financial institutions and governments [32], or we are figuring out the hierarchies in a corporation [78], the basic concepts of networks can be applied to many systems. We can, therefore, virtually always resort to networks whenever we want to study a system composed of somehow interrelated entities.

1.1.1.1 The Intriguing Case of The Internet

Let us now continue from this introductory chapter with a closer look at one of the most prominent modern technological systems, *the Internet*, which has come forward as a usable technology which brings the dream of a seamlessly connected world closer to reality.

The Internet is studied from (at least) four different levels of abstraction. At each level it is composed of different building blocks. Specifically, the Internet can be seen as:

1. A collection of electronic devices — the *routers* — that actually exchange packets of information over physical cables [192] [118].
2. A set of communicating, geographically annotated *Points of Presence* (POPs)[1].
3. A collection of *Autonomous Systems* (ASes)[2], whose interconnections are business-driven and result from multi- or bi-lateral commercial agreements [139].
4. A set of *Internet eXchange Points* (IXPs)[3], through which ASes exchange their customers' traffic [35] [98].

Each level of abstraction actually yields a different network representation of the Internet. Entities (nodes) are associated to routers, POPs, ASes, and IXPs respectively. Analogously, relationships (links) are manifold and their nature varies greatly among levels. These are *physical* interconnection cables, such as fiber optic cables,

[1] A POP is a concentration of routers in a facility from which Internet connectivity is provided to a geographical area such as city [58].
[2] An AS is a body of routers owned and administered by a single company [139].
[3] An IXP is a physical facility that allow ASes to interconnect directly to reduce costs or increase bandwidth speed [71].

at the router- and POP-level of abstraction. Alternatively, these are business-driven Border Gateway Protocol (BGP) sessions at the AS- and IXP-level [139].

Particularly critical to ensure global Internet connectivity is the AS-level network. As we have eluded to above, ASes are sets of routers, each one owned by a single company. The Internet, which is made of tens of thousands of ASes worldwide, strongly relies on inter-AS connections. Indeed, the traffic must be able to flow from (to) every AS, otherwise portions of the Internet would be unreachable. In order to allow inter-AS traffic to flow, AS owners agree to route the traffic directed to (originated from) their routers. Agreements always have financial outcomes for the companies owning ASes, e.g., revenue increases and cost reductions [90, 69, 123, 47, 195, 139]. A failure of the AS owners to negotiate and make business agreements will inevitably cause portions of the Internet to become unreachable — regardless of the physical underlying router-level interconnections. Historically, this event has occurred several times, causing blackouts of significant parts of the Internet [79, 200, 94]. Outages have also occurred due to malicious attacks [43, 155, 11]. Therefore, understanding the AS-level network is of high impact for the protection and deployment of the global Internet [105, 104, 179, 21, 105, 159, 150]. AS-level connectivity is also essential to design routing algorithms [189, 21, 137, 204], assess infrastructure resilience and robustness [178, 105, 153], and optimize content dissemination [174, 129, 8].

Another critical component to the relentless worldwide Internet connectivity is the *The Domain Name System (DNS)* [120]. Similar in the spirit to an address-book, the DNS is a naming system that locates hosts within the Internet — e.g., computers, servers and routers. More precisely, it translates textual *host names* to Internet Protocol (IP) *addresses*. Hidden amid a jumble of other systems, the DNS is actually a core Internet service. For example, it is used to translate every single name we type in our web browser address bar to surf the web. The DNS has been the target of several attacks [191, 84, 131]. Understanding and and analyzing the DNS is of fundamental importance for its protection [10, 181, 4, 202], as well as for the protection of its users [18, 89, 171].

1.1.2 Network Algorithms and the Cost of Extracting Knowledge

We have seen that networks are widely used to analyze real systems. We have also discussed the Internet as a representative example of a real-world system that can effectively be modeled using networks. This might suffice to convince the reader of the value of networks to understand modern-day systems. At this point we turn our discussion to the challenges that hamper the use of networks for *large-scale* systems analysis.

The effectiveness of networks as a tool for knowledge extraction needs, now more than ever, to be traded-off with the extra complexity inherent in large-scale networks. Networks are currently considered a *big data problem* [175] [185], particularly in disciplines such as engineering and computer science. At the present time, their sizes are typically in the order of tens, or even hundreds, of thousands entities, which are interrelated via a number of relationships which is often at least one order of magnitude greater than the entities. This is the case for all the examples we have cited above. There are also extreme cases of networks with millions (billions) of entities (relationships). Such extreme cases encompass online social platforms such as Facebook [187] [53] and Twitter [103] [186], which contain millions of users linked via billions of relationships.

It may be intuitive to the reader then, that the storage space required to maintain such massive networks may represent a real challenge. Indeed, finding efficient ways to store large-scale networks is now a hot research topic, and has led to the implementation of the so-called *graph databases* [162], among which we recall OrientDB [184, 44] and Neo4j [198, 133]. Even when the space does not represent an issue on its own, knowledge extraction typically remains challenging. Indeed, the intrinsic complexity associated to network size often causes *network algorithms* to suffer from scalability problems. In fact, the majority of network algorithms was designed prior to the rise of *big-data*, in which massive amounts of information have become available for analysis [122, 88, 208]. Only recently have researchers begun addressing scalability issues by proposing network algorithms which exploit parallel and distributed architectures such as multi-core Central Processing Units (CPUs) [124], distributed systems [175] [126], and Graphics Processing Units (GPUs) [77].

Network algorithms are at the basis of new knowledge extraction, since they enable key system features to be brought to light. Basically, such algorithms opportunely traverse the network, seeking for particular patterns of relationships among entities, often buried into the massive amount of existing entities and relationships.

1.1.2.1 Community Detection Algorithms

Community detection algorithms have become one of the most prominent classes of network algorithms. They look for special groups of entities in the network —the *communities*. A community is characterized by dense relationships among its entities, and sparser relationships between its entities and the remainder of the network. Communities are now widely acknowledged as being fundamental to uncover hidden patterns in the structure and function of networks [59, 108, 156, 158, 164, 166, 20, 110, 112]. The main reason is that, in the vast majority of real systems known, entities tend to organize in communities, and to establish relationships primarily with the other members of the communities they belong to. This is the principle known as *homophily* — the love for the similar — in social sciences [130] [117].

k-Clique Communities

Among all the definitions of communities proposed, k-clique communities are of keen interest due to their unique features and broad applications [68, 81, 83, 82, 86, 85, 205, 34, 147]. As it can be inferred from the name, cliques are at the basis of these communities. In some sense, a clique is the most tight concept of community, since *all* possible pairs of nodes in a clique are connected each other. The underlying clique structure of networks has been shown to strongly determine the structural properties of networks [74].

This is the main motivation that has led us to focus on the k-clique communities [148], which are unions of cliques well-interwoven and reachable to each other through paths involving other cliques only. To the best of our knowledge, that of k-clique community is the only definition which is based on the concept of clique and at the same time:

- *Is formally defined*, i.e. is based on network properties only and uses neither heuristics nor function optimizations.

- *Is totally deterministic*, i.e. there are no stochastic sources either in the definition or in the detection algorithm.
- *Allows overlap*, i.e. communities can be partially (even almost completely) superimposed.
- *Is local*, i.e. each community exists independently of the other communities.

It is interesting to note that k-clique communities paved the way for a novel analysis of the Internet [68]. They have also been used to design efficient forwarding algorithms for mobile telecommunications [81]. Social message forwarding schemes based on k-clique communities have been proposed as well [83, 82]. In the social sciences, k-clique communities are used, for example, to characterize the collaborations between scientists, and the calls between mobile phone users [146]. In other studies of note, they have been used to track *knowledge evolution* [86, 85].

Despite proven efficacy in advancing research, detecting k-clique communities remains highly inefficient. Currently algorithms prevent k-clique communities to be detected from large-scale networks. Some limited contributions have been made to improve their efficiency [102]. Nevertheless, improvements turn out to be very limited. In general, k-clique community detection continues to be cumbersome. In addition, to the best of our knowledge, theoretical worst-case complexity analyses of k-clique community detection algorithms have never been made before. Being able to rigorously quantify space and time complexities would enable to estimate, *a priori*, the amount of resources necessary to obtain these communities. Currently, algorithms are merely executed, and one has only to wait (usually a very long time) for a *possible* termination. No assumptions on the execution time can be made. Neither it can be determined in advance whether the algorithm will eventually produce the desired communities — rather than exhausting all the computational resources available without successfully produce the output.

1.2 Original Contributions

In this thesis we explore two areas of network research. We begin with the more theoretical area of parallel network community detection algorithms. Then, we move

to the more practical area of the application of networks to the real-world Internet system.

1.2.1 Parallel Network Community Detection Algorithms

Motivated by the unique features of the k-clique communities, and stimulated by the challenges hindering their detection from large-scale networks, in this thesis we contribute towards the study, analysis, and implementation of algorithms to detect these communities in parallel.

Our main contributions in this area are the following:

- We theoretically analyze the state-of-the-art algorithms for k-clique community detection, shedding light into previously unknown scalability issues.
- We present an innovative algorithm to obtain the connected components of networks which enables large-scale networks to be decomposed into an arbitrary number of smaller networks.
- We propose novel parallel algorithms to detect k-clique communities, making the source code freely available.
- We provide theoretical tight bounds on the space and time complexities of the algorithms proposed.
- We validate the algorithms on real large-scale networks and empirically measure:
  - Performances close to the theoretical speed limit;
  - Dramatic improvements via comparisons with the state-of-the-art.

1.2.2 Network-Based Methodologies to Study the Internet

Stimulated by the high impact that its understanding may have to protect and deploy resilient worldwide connectivity [179, 21, 105, 159, 150, 10, 181, 4, 202], we drill-down into real large-scale data sets to analyze the Internet from two alternative points of view. Specifically, we leverage on networks and network algorithms to investigate:

- The evolution of the AS-level network, which we combine with data from *stock markets*.
- The Internet DNS traffic directed to the entire set of ".it" domains.

Effective network models able to capture hidden features and key relationships are proposed for the aforementioned cases. Methodologies are then devised to derive new knowledge from networks. Validations are carried out using real data sets.

Our main contributions in the area of the application of networks to the Internet can be summarized as follows.

Internet Companies, ASes, and Stock Markets

We propose a general methodology to investigate the synchronous cross correlations of the connectivity features of nodes in evolving networks. To the best of our knowledge, such correlations have never been studied before. We believe they may be relevant for a better understanding of the complex techno-socio-economic factors underlying modern systems. In addition, understanding these features may contribute significantly to the design of novel evolutionary or predictive models. We instantiate this methodology to study the dynamics of connectivity among Internet companies, which physically interconnect to ensure global Internet access. We then combine connectivity dynamics with historical data from stock markets, and produce graphical representations to visually spot high correlations.

We find that geographically close Internet companies offering similar services are driven by common economic factors. We also provide evidence on the existence and nature of factors governing the dynamics of connectivity.

The Domain Name System (DNS)

We propose network models to effectively study the DNS traffic, which is necessarily generated to translate domain names into their corresponding Internet IP addresses. Relying on those network models, we also describe general methodologies that we use, for example, to rank Internet domains and unveil their relationships. We carry out validations on a large-scale, using DNS traffic records of all the Italian *dot-it* domains.

We discover that the “interest” shown for Italian domains, quantified via network metrics, follows a scale-free distribution [12]. A consequence is that very few domains are the target of almost all the DNS traffic. We also demonstrate how the mere DNS

traffic, if opportunely modeled into a network, actually carries valuable information to group similar sites and to spot trends and interests of the Internet community.

1.3 Organization of The Thesis

This thesis is organized as follows. In the next chapter we introduce and discuss fundamental notions related to networks and network theory. This constitutes the background on which this manuscript is built upon. The experienced reader may choose to directly proceed to Chapters 3 and 4 where the main results are illustrated. In Chapter 3 our original contributions in the field of parallel k-clique community detection are presented. In Chapter 4 we present our original contributions in the field of application of networks to real-world complex systems, with special emphasis on the Internet. We attempted to make each of these two chapters self-contained insofar as possible. Although we do not recommend it, one may decide to proceed directly to the chapter of interest.

The thesis concludes with a summary and outlook in Chapter 5. The appendix A contains the extended mathematical proofs of the theorems introduced in the thesis.

2

Background and Definitions

2.1 Networks: the Bridge between Raw Data and Knowledge

Networks are the building blocks upon which this thesis is built upon. In a certain sense, they lie exactly in between raw data and refined knowledge. Indeed, raw data is mapped into networks that, in turn, are opportunely processed via network algorithms in order to gain new insights.

Formally, a network $G = (V, E)$, is a pair of sets (V, E), with V the set of $|V|$ nodes and $E \subseteq V \times V$ the set of links. Networks are also known in the literature as a *graphs*. Similarly, nodes and links are also referred to as *vertices* and *edges*, respectively. In the remainder of this thesis we use all those terms interchangeably.

If edges $(i, j) \in E$ are unordered pairs, then G is said to be *undirected*. Two nodes $i, j \in V$, $i \neq j$, are said to be adjacent if $(i, j) \in E$. If G is *directed*, the pairs $(i, j) \in E$ are ordered and semantically represent "from i to j" relationships. Adjacency relationships can be represented for each pair of nodes i and j $(i, j = 1, \cdots, |V|)$ with an $|V|$-square adjacency matrix $\mathbf{A}$ whose off-diagonal elements $a_{i,j}$ are equal to 1 if $(i, j) \in E$ or 0 otherwise. In-diagonal elements $a_{i,i}$ are always equal to 0 when self-edges are not allowed. $\mathbf{A}$ can be generalized by associating real numbers to its elements $a_{i,j}$ in order to encode general tie strengths between nodes rather than binary adjacencies. In the latter case the graph is said to be *weighted*. Whenever V can be divided into two disjoint sets R and D such that each edge joins a node in R to a node in D, then G is said to be *bipartite*. Any bipartite graph can be represented with an adjacency matrix of the form

$$\mathbf{A} = \begin{pmatrix} \mathbf{0} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{0} \end{pmatrix},$$

where $\mathbf{B} = [b_{r,d}]$ is a matrix with $|R|$ rows and $|D|$ columns, uniquely identifying the bipartite graph. Rows (columns) of $\mathbf{B}$ represent nodes in R (in D) and elements $b_{r,d}$ are equal to 1 whenever r and d are adjacent or 0 otherwise.

2.2 Connectivity Features of Network Nodes

In this section we discuss some of the connectivity features that can be used to characterize a node in a network. We divide connectivity features into three groups, namely: direct, local and global features.

2.2.1 Direct Connectivity

A connectivity feature is direct if it necessarily takes into account only node interconnections. More precisely, given a node n, a connectivity feature is local if it only considers pairs $(i, j) \in E$ such that i or j are equal to n. In other words, only the edges that have n as an endpoint are taken into account.

The most common feature that belongs to this group is the *degree*. The degree of node n, which we indicate with the symbol d_n, counts the number of edges with n as an endpoint. The degree is of immediate interpretation, since it tells the propensity of a node to establish relationships with others. However, depending on the specific characteristics of a graph, the definition of degree may be slightly extended.

If a graph is directed, two different degrees can be obtained for each node, namely: the *in-degree*, and the *out-degree*. The in-degree (out-degree) counts the number of edges that terminate (originate) on node n. Formally, the in-degree (out-degree) count the number of pairs $(i, j) \in E$ such that $i = n$ ($j = n$). Similarly, if the graph admits self-edges, we can consider the self-degree, that counts the number of edges a node has with itself, i.e., the number of pairs $(i, j) \in E$ such that $i = j = n$.

For weighted graphs we can identify the *intensity* as a direct connectivity property, that is, the sum of the weights of edges that originates from (terminates in) node n — or both.

2.2.2 Local Connectivity

A connectivity feature is local if it necessarily takes into account only: node interconnections; and interconnections between node neighbors. The set of neighbors of a node n is formally expressed as $N(n) = \{j|(n, j) \in E \wedge (j, n) \in E\}$.

One common connectivity feature, that we can place in this category, is the *average neighbor degree* [136]. Formally, the average neighbor degree knn_n for node n reads

$$knn_n = \frac{1}{k_n} \sum_{i \in N(n)} k_i. \tag{2.1}$$

This feature gives indication on the propensity of node neighbors in establishing relationships with the other members of the network. It can also be interpreted as the attitude of a node to establish connection with hubs, i.e., highly connected nodes in the network.

Another well-known connectivity feature that we can place in this category is the *clustering coefficient* [197]. This feature gives and indication of the propensity of the neighbors of a node to interact each other. When all the neighbors are interacting each other — that is, when they form a clique — then the clustering coefficient has the maximum value equal to 1. Specifically, for undirected graphs, the clustering coefficient cc_n of node n reads

$$cc_n = \frac{|\{(i, j)|i, j \in N(n) \wedge (i, j) \in E\}|}{1/2 \cdot d_n(d_n - 1)}. \tag{2.2}$$

The product $1/2 \cdot d_n(d_n - 1)$ is used as a normalization factor, since it equals the maximum number of edges that can exist between d_n nodes in an undirected graph.

In this case, i.e., when a group of nodes form a complete subgraph, it is said to be a clique.

Indeed, cliques are another relevant feature that can be placed among local connectivity features. In general, they are a good indicator for an high number of interactions among their members — actually, it is the maximum number possible. Formally, a k-clique, that is, a clique with size k, is a subset of the vertex set $c \subseteq V$ such that there is an edge $(i, j) \in E$ between any two nodes $i, j \in c$ and $|c| = k$. The size of a clique is very important in quantifying its relevance. The larger the size, the higher the number of interactions. Broadly speaking, large cliques are symptoms of strongly interacting groups. If, as we have done for the other connectivity features, we focus on the single node, we can count the number of cliques it belongs to as a local connectivity property. Indeed, this may be a good signal of node propensity in having groups of interests — e.g., clubs, friends, businesses, partners, etc. Similarly, if we want to exclude small groups, we can count only cliques greater than a given size. This, for example, the approach followed by Palla *et al.* [148].

The number of 3-cliques a node has, that is known in the literature also as the number of triangles, is, of course, another connectivity feature of the local kind. Triangles have been shown to be extremely relevant, especially in (online) social networks to predict future relationships formation [119].

2.2.3 Global Connectivity

A connectivity feature is global if it necessarily takes into account: node interconnections; interconnections between node neighbors; between node neighbors' neighbors; and so on, possibly until all the nodes in the network have been considered. Broadly speaking, such features are able to characterize the node in the context of the whole network.

Among global features, we cite the *eigenvector centrality* [24] x_n for a node n, which is defined as

$$x_n = \lambda^{-1} \sum_{j \in N(n)} x_j = \lambda^{-1} \sum_j a_{n,j} x_j. \tag{2.3}$$

The previous formula expresses the eigenvector centrality of n as the sum of the eigenvector centrality of its neighbors — up to a factor of scale. This is a recursive formula, since neighbors eigenvector centrality depends, in turn, on the eigenvector centrality of their neighbors, and so on. Intuitively, the higher the centrality, the higher the relevance of the node in the network. Using matrix notation, we can rewrite the previous formula as the eigenvector equation

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x}. \tag{2.4}$$

In general, many different eigenvectors exists. However, if we choose the largest eigenvalue, then, all the eigenvector components are positive (Perron-Frobenius theorem) and the feature can be seen as representative of node relevance.

Many other global connectivity features, based on the concept of centrality, exist in the literature. For the sake of example, we mention the betweenness centrality [27]. This is a fundamental feature, since it counts the number of shortest paths in the network that traverse the node. The higher this feature, the more central the position of the node in the network.

Among other global connectivity features we also recall the *coreness*. This feature is related to the k-cores of a graph [7]. A k-core is a maximal subgraph of G in which each node has at least degree k — maximal means that we cannot include any other node in the subgraph and preserve the property on the minimum degree. From this definition it follows that k-cores are nested, i.e., a 5-core is contained in a 4-core that, in turn, is contained in a 3-core, and so on. A node n, that belongs to a k-core, but not to a $(k+1)$ core, is said to have coreness k. An high coreness is, reasonably, participating to the most densely interconnected zones of the network. Cores have been shown, for example, to be crucial to the spreading of information in networks [96].

Another global connectivity feature that somehow related to the coreness, is the *denseness*. Similarly to the k-cores, we can define the k-denses of a graph [169]. A k-dense is a maximal subgraph of G, in which each pair of adjacent nodes has at least k common adjacent nodes in the subgraph. Again, k-denses are nested. In

Feature	*kcliques*	*infomod*	*infomap*	*blondel*	*kdense*	*oslom*	*gce*
weighted networks	-	-	+	+	-	+	-
directed networks	-	-	+	-	-	+	-
overlapping communities	+	-	-	-	-	+	+
hierarchical communities	+	-	-	+	+	+	-
deterministic	+	-	-	-	+	-	-
optimization based	-	+	+	+	-	+	+
arbitrary thresholds	-	+	+	-	-	+	+

Table 2.1. Features of community detection algorithms

addition, a k-dense is a subset of a k-core. A node n is said to have denseness k if it belongs to a k-dense but not to a $(k+1)$ dense. Denses have been used to identify the nucleus of the Internet [65].

2.3 Community Detection in Networks

The number of network community detection algorithms that have been developed during last years is huge. Therefore, a comparative analysis of all these algorithms is not feasible and a selection of a representative group is necessary. We choose a representative group in the remainder of this section, whereas we thoroughly discuss k-clique community detection algorithms in the next chapter.

In order to identify a significant group, we have chosen three of the best performing algorithms analysed in [106]. The first two were proposed by Rosvall and Bergstorm [164] [166], and the third by Blondel et al. [20]. Since these three algorithms introduce heuristics and rely on optimization problems, we also selected another algorithm solely based on the k-denses of a graph (Saito et al. [169]). Finally, we have also chosen two recent algorithms for the detection of *overlapping* community structure (Lancichinetti et al. [110] and Lee et al. [112]). Below we provide a short description of each algorithm considered. Their main features are summarized, and compared with k-clique community detection (*kcliques*), in Tab. 2.1.

Structural Algorithm by Rosvall and Bergstorm

(*infomod*) [164]. A description of a network by means of its communities can be viewed as a lossy compression of network topology. Here, the process of detecting communities is turned into the information-theoretical problem of optimally compressing network structure. Specifically, information about the original network description X is encoded as a compressed description Y, which tells most of the original X — Y is chosen with the aim of maximizing the mutual information between network and description. Simulated-annealing is used to accomplish such optimization since an exhaustive search is computationally infeasible.
The implementation of this algorithm is open-source and freely available at [163].

Dynamic Algorithm by Rosvall and Bergstorm

(*infomap*) [166]. This algorithm is in the spirit of *infomod*. Here communities are identified by finding an optimally compressed description of how information flows on the network. The rationale is that a group of nodes among which information flows quickly can be grouped into a single community. The optimal compressed description is obtained by minimizing the minimum description length [161] [72] of a random walk, which is used as a proxy for information flow. A greedy search algorithm is used to obtain the results which are refined with a simulated-annealing technique.
The source code of this algorithm is freely available and can be downloaded from [165].

Fast Modularity Optimization by Blondel et al.

(*blondel*) [20]. This heuristic algorithm is based on a local modularity [134] optimization in the neighborhood of each node. This algorithm is divided into two phases. The optimization is carried out in the first phase in order to identify communities — this process terminates when a local maxima of the modularity is attained. In the second phase, communities are replaced by super-nodes and the procedure is repeated iteratively until modularity stops increasing.
Source code of this algorithm is freely available for download at [19].

k-Dense Method by Saito et al.

(*kdense*) [169]. This algorithm deterministically extracts k-dense communities from a given network. k-dense communities are defined only according to their topological properties and their extraction do not require function optimizations. These communities are well-interconnected sub-networks such that each pair of adjacent nodes in a community must share a minimum number of neighbors within that community. More precisely, a k-dense community $D(k)$ is a sub-network such that each pair of adjacent nodes in $D(k)$ has at least $(k-2)$ neighbors in common in $D(k)$.
Since the implementation of this algorithm is not publicly available, we implemented this algorithm on our own.

Order Statistics Local Optimization Method by Lancichinetti et al.

(*oslom*) [110]. This algorithm is based on the local optimization of a fitness function expressing the statistical significance of communities with respect to random fluctuations — a network community is statistically significant if it is unlikely to find it in a random network with the same degree distribution. Basically, *oslom* consists of the following sequential phases. First, it looks for significant communities by minimizing the statistical significance. Then, it analyses the set of communities obtained, trying to detect their internal structure or possible unions thereof. Finally, it detects the hierarchical structure of the communities.
The implementation of *oslom* is open-source and available at [109].

Greedy Clique Expansion by Lee et al.

(*gce*) [112]. This algorithm selects maximal cliques as initial seeds and then adopt the general strategy of expanding these seeds via greedy local optimization of a fitness function. The function used here, introduced by Lancichinetti et al. in [107], is simply the ratio between community's internal and total degree. Therefore, it well captures the generally agreed concept of community as of a group of nodes highly interconnected each other but relatively less interconnected with the remainder of the network.

Source code of *gce* implementation is publicly available and can be downloaded from [111].

3

Parallel Network Community Detection Algorithms

3.1 What, Why and How of k-Clique Communities

Networks — in the sense in which they are used in this thesis — are graphs modeling real-world complex systems. Detecting communities from these networks may be decisive in the understanding of their structural and functional properties [60] [148]. Examples, which we have thoroughly discussed in the introductory chapter of this thesis, include, but are not limited to, the Internet [68] and the World Wide Web [101] as well as mobile phone [142], collaboration [62], citation [38] and biological [62] networks.

Cliques can be thought of as being building blocks of networks. In fact, structural properties of networks can be viewed as consequences of their underlying clique structure [74]. In addition, cliques represent the most tight concept of community — all possible pairs of nodes in a clique are *interacting* each other. Those are the main reasons behind the widespread diffusion of k-clique communities as a tool to investigate the structure and function of networks. Indeed, k-clique communities are unions of cliques well-interwoven and reachable each other through paths involving other cliques only. In the introduction of this thesis we discuss the unique features of k-clique communities and present some real-world use cases. Here we recall some examples for the sake of completeness. For instance, in [81] the authors identify k-clique communities among the participants of Infocom06 and the students in the MIT Media Laboratory and exploit this information to design efficient forwarding algorithms for mobile networks. Similarly, in [83] the authors propose a distributed

k-clique community detection algorithm to be used for *social-based* message forwarding. k-clique communities also find application in social sciences. For example, in [146] they are used to capture the relationships characterizing the collaboration between scientists and the calls between mobile phone users.

The first algorithm for extracting k-clique communities is the Clique Percolation Method (CPM) [149], which is prohibitively memory and time demanding. To the best of our knowledge, this had prevented k-clique communities from being extracted from large-scale networks such as those considered in this thesis. Here, we adopt a theoretical approach to shed light on CPM scalability issues. Then, we design the novel CPM On Steroids (COS) parallel algorithm. COS is the refinement of a working proof-of-concept, which is presented first in order to clarify the problem of parallel k-clique community detection. COS exploits parallel processing to reduce execution time and has a low memory footprint. Its maximum degree of parallelism, unbounded, user-configurable and input-independent, enables hardware resources to be used efficiently. In addition, we provide analytical tight upper bounds on its execution time and space requirements, which are given as function of: *i)* the number of maximal cliques in the network; *ii)* the size of the maximal cliques; and *iii)* the number of processors available. These bounds prove that COS has a *linear* space dependence on the number of maximal cliques and a worst-case execution time inversely proportional to the number of processors. By means of the aforesaid bounds we can answer questions such as “Is memory available on this hardware enough to extract k-clique communities from this network?” or “If the number of processors installed on a particular machine is doubled, would COS halve its execution time?”. Therefore, we are providing a framework with which it is possible not only to extract k-clique communities efficiently, but also to estimate in advance the required amount of computing resources required. These theoretical bounds are validated in a series of experiments. We experimentally measured a *linear speedup*: COS execution time halves when the number of processors it uses is doubled. Dramatic reductions in execution time and memory footprint are brought to light by comparisons with other state-of-the-art k-clique community detection algorithms. The implementation of COS is open-source and freely available [66].

Another major contribution of this thesis is the innovative CONNECted componenTs MErging (CONNECT_ME) algorithm. By taking advantage of CONNECT_ME, it is possible to split a network into an arbitrary number of subnetworks, and still be able to obtain its connected components. This novel low-complexity algorithm plays a key role in COS, by combining together partial results from all processors.

The rest of this chapter is structured as follows. The next section contains a brief overview of the efforts toward efficient community detection, with special emphasis on k-clique communities. In Sect. 3.3 we formulate the problem of k-clique community detection. We discuss CPM in Sect. 3.4, highlighting its scalability issues. In Sect. 3.5 we present the novel CONNECT_ME technique. A working proof-of-concept parallel k-clique community detection algorithm and its worst-case complexities are presented in Sect. 3.6, whereas in Sect. 3.7 we propose COS and two enhancements at the basis of its functioning. In Sect. 3.8 we examine algorithms performances via experiments. The chapter concludes with a summary in Sect. 3.9.

3.2 Related Work

Traditionally, the relevance of the discovered network communities has been traded off with the complexity required for their extraction. For example, k-core communities [169] can be obtained with low complexity [15] but they are loosely-connected and non-overlapping. Conversely, k-clique communities [148] are fine-grained, overlapping and tightly-connected but their extraction is extremely demanding in terms of computational resources [149]. Very little work has been done to avoid trading off the quality of the extracted communities for complexity. Parallelism has been proposed in [207] and [168] as a means to alleviate computational costs. In [207] the authors heuristically evaluate the *propinquity*, i.e. the probability that a pair of nodes is involved in a coherent community. They update the original network by adding (removing) edges if the propinquity is higher (lower) than a given threshold. A parallel algorithm is used to update propinquity incrementally, in order to reflect network changes. Through this, they were able to extract meaningful communities from the huge Wikipedia linkage network. Rather than introducing a new definition of community, in [168] the authors propose a algorithm to reduce the size of the networks. In

parallel, they use an heuristic to locate quasi-cliques and assign them as nodes in a reduced graph to be used with standard community detection algorithms. These reduced graphs have a size which is approximately one half of the original size. Alternatively than exploiting parallelism, computational costs can be mitigated by designing efficient heuristics and greedy local function optimizations. To the best of our knowledge, algorithms proposed in [20] and [166] are two of the fastest (and best-performing according to [106]) optimization-based community detection algorithms. We analyse their performances on real-world network data in Sect. 3.8. Their key concepts are explained in Sect. 2.3.

The first k-clique community detection algorithm is the Clique Percolation Method (CPM) [149]. It first lists all the maximal cliques from the input network and then analyses the overlap between each possible pair of them. The number of maximal cliques in a network could be exponential with the number of nodes [132]. Nevertheless, none of the real-world networks we studied has a number of maximal cliques greater than few millions — this means that their actual number is from tens to tens of thousands orders of magnitude smaller that their maximum theoretical number. As a consequence, we were able to obtain the whole list of maximal cliques from the networks considered in this thesis in at most a couple of minutes with a serial algorithm [29]. Therefore, we do not add anything new to this point and we refer the interested reader to [22, Sect. 5] for a review, or to [49], [170] and [206] for parallel algorithms. The real challenge is finding an efficient way to store and analyse the overlap between maximal cliques. In Sect. 3.4 we show that this has a complexity proportional to the *square* of the number of maximal cliques.

In [102] the first effort towards efficient k-clique community detection was made. The authors proposed the Sequential Clique Percolation (SCP) algorithm, which enables k-clique communities to be detected at multiple weight thresholds in a single run. Although SCP can detect communities on weighted networks, it cannot produce k-clique communities for each possible k in a single execution. Moreover, since it enumerates cliques rather than maximal cliques, it only works well on sparse networks. In fact, as also highlighted by the authors, given that a clique with size h contains $\binom{h}{k} \approx h^k/k!$ smaller cliques with size k, the huge number of cliques it generates on networks with fairly large maximal cliques — as those considered in this thesis —

prevents communities to be obtained in a reasonable amount of time.
In [70] we drew our first ideas on how to enhance CPM and proposed a simple parallel k-clique community detection algorithm. Although this algorithm is parallel and has a reduced memory footprint with reference to CPM, it does have several drawbacks. For example, its maximum degree of parallelism is: *i)* upper bounded by the size of the largest maximal cliques; *ii)* strongly affected by maximal cliques distribution; and *iii)* a decreasing function of the execution time.

3.3 Problem Formulation

Let $G = (V, E)$ be an undirected, unweighted graph without isolated nodes (vertices) and self-edges. V is its vertex set and $E \subseteq V \times V$ its edge set. A k-clique in G is a subset of the vertex set $c \subseteq V$ such that there is an edge $(i, j) \in E$ between any two nodes $i, j \in c$ and $|c| = k$. A clique is a k-clique for some k. Two k-cliques are *adjacent* if they have $(k - 1)$ nodes in common. Based on this notion of adjacency, we can define a k-clique community as follows.

Definition 1. *A k-clique community is the union of all the k-cliques that can be reached by each other through a series of adjacent k-cliques.*

Figure 3.1 shows a graph with its k-clique communities at $k = 3$ and $k = 4$. At $k = 2$ there exists only one community, corresponding to the whole graph.

Now observe that each h-clique always contains a number $\binom{h}{k}$ of *adjacent* k-cliques for each $k \leq h$. In other words, an h-clique is (in a) k-clique community for each $k \leq h$. For example the clique $\{1, 2, 3, 4\}$ of Fig. 3.1 contains $\binom{4}{3} = 4$ adjacent 3-cliques: $\{1, 2, 3\}$, $\{1, 2, 4\}$, $\{1, 3, 4\}$ and $\{2, 3, 4\}$ and therefore is in a 3-clique community. Similarly, it has $\binom{4}{2} = 6$ adjacent 2-cliques and hence is also in a 2-clique community.

Furthermore, if the h-clique is *not* contained in any other larger clique, i.e. it is a *maximal* h-clique, it belongs only to k-clique communities with $k \leq h$. Indeed, a maximal clique — that is, a maximal k-clique for some h — cannot share a number of nodes greater than or equal to its size. Otherwise it would be contained in a larger clique and, in turn, it would not be maximal. For instance clique $\{6, 9, 10\}$ of Fig. 3.1,

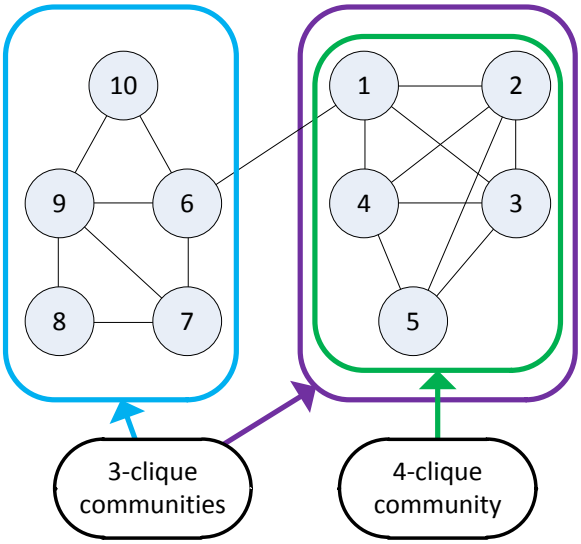


Fig. 3.1. A graph with its k-clique communities for $k = 4$ and $k = 3$.

which is maximal since there not exists another larger clique containing it, is in a 3-clique community as well as in a 2-clique community, but it cannot belong to any community with size greater than or equal to 4. Instead, the clique $\{3, 4, 5\}$, which is not maximal, belongs also to a 4-clique community. These observations allow us to formulate an equivalent definition of k-clique community.

Definition 2. *A k-clique community is the union of all the maximal h-cliques, $k \leq h$, that can be reached by each other through a series of adjacent k-cliques.*

Accordingly, the problem of k-clique community detection on networks *(a)* is to find all the possible unions of maximal cliques satisfying Def. 2; equivalently, *(b)* it is to find all the possible unions of cliques satisfying Def. 1. In the remainder of this thesis we concentrate on formulation *(a)*.

From formulation *(a)*, it follows that the lower the k, the higher the number L_k of maximal h-cliques, $k \leq h$, among which to search for k-clique communities. If l_k denotes the number of maximal k-cliques in G, we can express this number as $L_k = \sum_{h=k}^{k_{max}} l_h$, where k_{max} is the maximal cliques maximum size. L_k is maximum for $k = 2$. In fact, L_2 is equal to the number $l = \sum_k l_k$ of maximal cliques in G.

$c_0 = \{1,2,3,4\}$
$c_1 = \{2,3,4,5\}$
$c_2 = \{6,9,10\}$
$c_3 = \{6,7,9\}$
$c_4 = \{7,8,9\}$
$c_5 = \{1,6\}$

(a)

	0	**1**	**2**	**3**	**4**	**5**
0	4	3	0	0	0	1
1	3	4	0	0	0	0
2	0	0	3	2	1	1
3	0	0	2	3	2	1
4	0	0	1	2	3	0
5	1	0	1	1	0	2

(b)

Fig. 3.2. (a) The $l = 6$ maximal cliques $c_0, \cdots, c_5$ extracted from the graph in Fig. 3.1. (b) The resulting clique-clique overlap matrix. Maximal clique c_i is associated with row (column) i.

3.4 Scalability Issues of k-Clique Community Detection

In this section we discuss the Clique Percolation Method (CPM) algorithm, pointing out its scalability issues. We partition CPM into three subsequent phases for the sake of simplifying the presentation, namely: Maximal Cliques Listing; Clique-Clique Overlap Matrix Construction; and k-Clique Community Extraction. However, as already discussed in Sect. 3.2, maximal cliques listing does not represent an issue when dealing with real-world networks, at least with those considered in this thesis. For that reason, in the remainder of this section we concentrate only on the latter two phases.

Clique-Clique Overlap Matrix Construction

Given the whole list of maximal cliques, CPM builds a clique-clique overlap matrix as described in [52]. Each maximal clique is associated with a row (column) and the elements of the matrix represent the number of shared nodes between the corresponding maximal cliques. In the remainder of this thesis, we assume maximal clique c_i to be always associated with row (column) i. Figure 3.2(a) shows the list of the $l = 6$ maximal cliques extracted from the graph in Fig. 3.1, whereas Fig. 3.2(b)

shows the resulting clique-clique overlap matrix. The clique-clique overlap matrix is symmetric and diagonal elements represent the size of the maximal cliques.

It is clear that with a standard storage format, the space complexity of the matrix scales quadratically with l — in spite of simple optimizations that take into account, for example, the symmetry of the matrix. More efficient storage formats have been proposed for sparse matrices [45], however experimental results have shown that clique-clique overlap matrices can be very dense, i.e. have almost all non-zero elements. This quadratic dependence on l represents the first scalability issue that makes CPM inapplicable on networks modeling real-world systems. The second issue concerns the worst-case time complexity for computing the clique-clique overlap matrix which is in $\Omega(l^2)$ since overlap has to be computed for each of the $\binom{l}{2}$ possible pairs of maximal cliques.

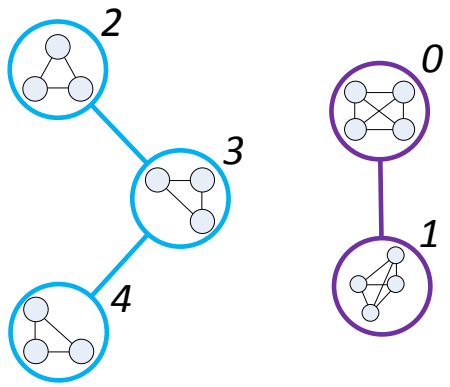
k-Clique Community Extraction

CPM extracts k-clique communities starting from the clique-clique overlap matrix as follows. It *i)* puts at 1 every on-diagonal element greater than or equal to k and every off-diagonal element greater than or equal to $(k-1)$; then, it *ii)* zeroes each other element, obtaining a binary matrix. Finally, it extract communities by carrying out a component analysis of this binary matrix.

Rather than accomplishing such analysis, we can relate k-clique communities to the connected components of a graph G_k, which we call henceforth the *clique-clique graph*. More precisely, if $G_k = (V_k, E_k)$ is a graph whose adjacency matrix is obtained according to *i)* and *ii)* above, and if no node whose row (column) has all zero elements is in V_k, then k-clique communities are the unions of maximal cliques associated with nodes in the connected components of G_k. Indeed, it is easy to check that *i)* and *ii)* assure that an edge exists between two nodes of G_k iff the corresponding maximal cliques have size greater than or equal to k and share at least $(k-1)$ nodes. Figure 3.3(a) shows the binary matrix obtained from the clique-clique overlap matrix of Fig. 3.2(b) for $k = 3$. The row with index 5 contains only zeros since it relates to $c_5 = \{1, 6\}$, which cannot share $(k-1) = 2$ nodes with any other maximal clique. The resulting clique-clique graph $G_3 = (V_3, E_3)$ is shown in

	0	1	2	3	4	5
0	1	1	0	0	0	0
1	1	1	0	0	0	0
2	0	0	1	1	0	0
3	0	0	1	1	1	0
4	0	0	0	1	1	0
5	0	0	0	0	0	0

(a)



(b)

Fig. 3.3. (a) The binary matrix obtained from the clique-clique overlap matrix of Fig. 3.2(b) for $k = 3$. (b) The resulting clique-clique graph G_3.

Fig. 3.3(b). It has $|V_3| = L_3 = 5$ nodes. Edges represent the condition of having 2 nodes in common. The two connected components of G_3, highlighted with different colors, contain maximal cliques corresponding to the two 3-clique communities of G.

3.5 Algorithms to Extract and Merge Connected Components

In Sect. 3.4 we have shown a relation between k-clique communities and the connected components of a graph. Here we propose the innovative CONNECted componenTs MErging (CONNECT_ME) algorithm, which enables the connected components of the union of two graphs to be obtained without knowing their topologies. CONNECT_ME will be used in the next sections when combining parallel processors' partial results. Since CONNECT_ME has to manipulate disjoint sets to efficiently maintain the connected components, in this section we also briefly discuss the *set union problem* and a well-known algorithm for its solution.

Connected Components as a Solution to the Set Union Problem

Connected Components, which are disjoint sets of nodes, can be obtained using any algorithm for solving the *set union problem* [182]. This problem consists in maintaining a collection F of disjoint sets under an intermixed sequence of $find_F$ and $union_F$

Algorithm 1: $MERGE\_SETS(F, p, q)$

Input: A collection F of disjoint sets and two elements p and q
Ensure: sets containing p and q are merged in F

```
1 begin
2     P ← find_F(p)
3     Q ← find_F(q)
4     if P ≠ Q then
5         union_F(P, Q)
```

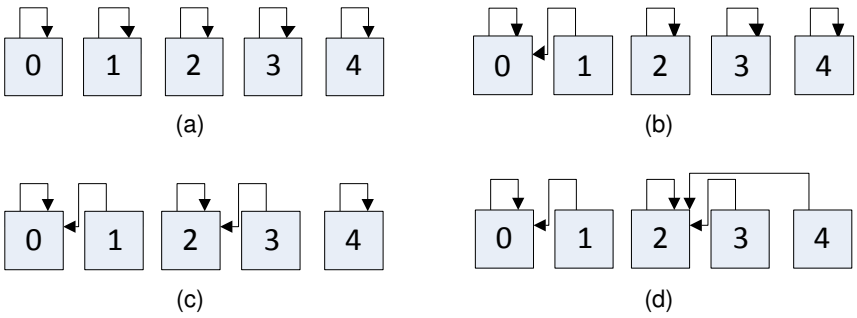


Fig. 3.4. Dynamic evolution of a collection F of disjoint sets.

operations. $find_F(p)$ returns the canonical element of the set containing element p — the canonical element is an arbitrary but unique element identified within each set, which is used to represent the set. $union_F(P, Q)$ combines the sets whose canonical elements are P and Q into a single set, and make P the canonical element of the new set.

If we initialize F with $|V|$ singleton sets $\{v\}$ such that $v \in V$, we can obtain the connected components of a graph in F after $MERGE\_SETS(F, p, q)$ has been called on each edge $(p, q) \in E$ [51]. $MERGE\_SETS$, which is presented in Algorithm 1, retrieves the canonical elements P and Q of the sets containing p and q via two $find_F$ operations. If $P \neq Q$, then p and q are in two different sets, and they are merged with a $union_F$.

Figure 3.4 shows the dynamic evolution of a collection F of disjoint sets resulting from the call of $MERGE\_SETS(F, p, q)$ on each edge (p, q) of the clique-clique graph G_3 in Fig. 3.3(b). Initially F contains five singletons, one for each node of the graph (Fig. 3.4(a)). Then, $MERGE\_SETS(F, 0, 1)$, for the edge $(0, 1)$,

is called the on the collection of singleton sets. The call determines the execution of $find_F(0)$ and $find_F(1)$, which returns canonical elements 0 and 1, respectively. Hence, $union_F(0,1)$ is executed as well, resulting in the collection in Fig. 3.4(b). The second call is $MERGE\_SETS(F,2,3)$, for the edge $(2,3)$, that alters the state of the collection from Fig. 3.4(b) to Fig. 3.4(c). Finally, $MERGE\_SETS(F,3,4)$ is called for the edge $(3,4)$. The call causes the execution of $find_F(3)$ and $find_F(4)$, which returns canonical elements 2 and 4, respectively. Thus, $union_F(2,4)$ alters the collection to the state in Fig. 3.4(d). In this final state, disjoint sets in F are the connected components of G_3.

To the best of our knowledge, the fastest algorithm for the solution of the set union problem is presented and analysed in [183]. This algorithm represents each set in F as a *rooted tree*[1] whose nodes are the elements in the set and whose root is the canonical element. Each node has an outgoing link to its *father*-node — itself if the root — in the tree. $find_F(p)$ returns the root of the tree containing p and $union_F(P,Q)$ combines the trees whose roots are P and Q, by making P the new root of Q. If two simple optimization rules are applied, this algorithm reaches an $O(f\alpha(f,g))$ worst-case time complexity for f operations on g initially singleton sets, assuming $f = \Omega(g)$. α is a functional inverse of Ackermann's function. For $i,j \geq 1$ let $A(i,j)$ be defined by:

$$
\begin{aligned}
A(1,j) &= 2^j && \text{for} && j \geq 1,\\
A(i,1) &= A(i-1,2) && \text{for} && i \geq 2,\\
A(i,j) &= A(i-1,A(i,j-1)) && \text{for} && i,j \geq 2.
\end{aligned}
$$

Then, $\alpha\,(f,g) = min\{i \geq 1 : A(i,\lfloor f/g \rfloor) > \log_2 g\}$. This function grows very slowly and for all practical purposes is a constant no larger than four [182].

An Algorithm to Merge the Connected Components

We now present CONNECT_ME which, starting from the connected components of two graphs $H_1 = (V_1,E_1)$ and $H_2 = (V_2,E_2)$, $V_2 \subseteq V_1$, enables the con-

[1] In the remainder of this section we use rooted trees to graphically represent disjoint sets.

Algorithm 2: $CONNECT\_ME(F_1, F_2)$

Input: Two collections of sets, F_1 and F_2, corresponding to the connected components of the graphs $H_1 = (V_1, E_1)$ and $H_2 = (V_2, E_2)$, $V_2 \subseteq V_1$, respectively.

Ensure: Disjoint sets in F_1 correspond to the connected components of $H_1 \cup H_2$

```
1 begin
2     foreach u ∈ V_2 do
3         U ← find_{F_2}(u)
4         if U ≠ u then
5             MERGE_SETS(F_1, U, u)
```

nected components of their union $H_1 \cup H_2$ to be obtained. If F_1 and F_2 contain disjoint sets equivalent to the connected components of H_1 and H_2 respectively, $CONNECT\_ME(F_1, F_2)$, produces the connected components of the union of the two graphs without any information neither on the edges E_1 nor on the edges E_2. $CONNECT\_ME$ is described in Algorithm 2. For each element $u \in V_2$, the canonical element U of the set containing u is found in F_2 and both U and u are merged in F_1. The basic idea behind this algorithm is: "given that u and U are in the same connected component of H_2, they must also be in the same connected component of $H_1 \cup H_2$".

CONNECT_ME is formalized in the following theorem.

Theorem 1. *If F_1 and F_2 are collections of sets corresponding to the connected components of graphs $H_1 = (V_1, E_1)$ and $H_2 = (V_2, E_2)$, $V_2 \subseteq V_1$, then $CONNECT\_ME(F_1, F_2)$ ensures F_1 contains sets corresponding to the connected components of $H_1 \cup H_2$.*

Proof. See Appendix A.1.

In Fig. 3.5 we illustrate an example. We show two graphs H_1 and H_2 and collections of disjoint sets F_1 and F_2 corresponding to their connected components. Suppose we call $CONNECT\_ME(F_1, F_2)$. Changes are applied to F_1 only when $u = 1$ and $u = 4$ in the `foreach`. In the other cases (i.e., when u is equal to 0, 2 and 3) the condition in line 4 is not met.

When $u = 1$, $find_{F_2}$ returns 0. Therefore, MERGE_SETS combines the sets containing 0 and 1 in F_1. After this merge, F_1 becomes identical to Fig. 3.4(c). Similarly,

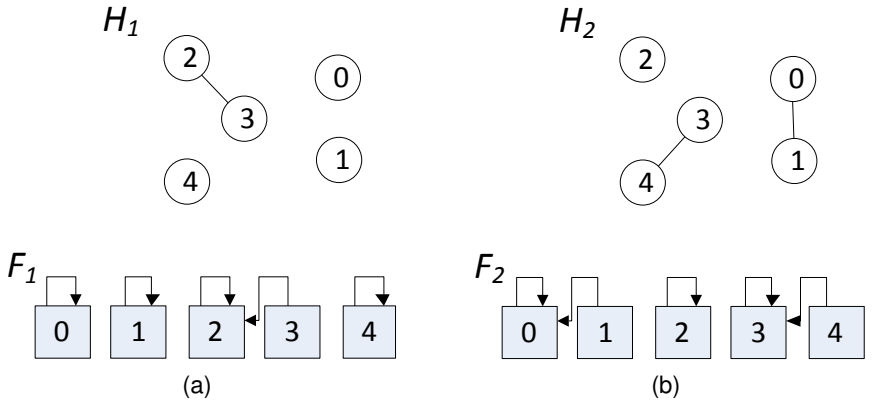


Fig. 3.5. Two graphs H_1 and H_2, and collections of disjoint sets F_1 and F_2 corresponding to their connected components. F_1 becomes identical to Fig. 3.4(d) after $CONNECT\_ME(F_1, F_2)$ has been called.

when $u = 4$, $find_{F_2}$ returns 3 and MERGE_SETS combines the sets containing 3 and 4 in F_1, which becomes identical to Fig. 3.4(d). This latter figure contains disjoint sets corresponding to the connected components of $H_1 \cup H_2$. Since we chosen the graphs such that $H_1 \cup H_2 = G_3$, we obtained the connected components of the clique-clique graph in Fig. 3.3(b) — via the connected components of its subgraphs only.

A general approach, which uses CONNECT_ME to merge the connected components of an arbitrary number of subgraphs, is developed in the next section to detect k-clique communities in parallel.

3.6 A Parallel Algorithm to Detect k-Clique Communities

In this section we present a k-clique community detection algorithm which serves as a working proof-of-concept that: *i)* reduces the execution time by exploiting parallel architectures; *ii)* efficiently distributes the load between the processors; *iii)* drastically reduces memory requirements; and *iv)* enables the analytic determination of the resources to be provisioned.

The proof-of-concept CPM On Steroids (COSpoc) is described in Algorithm 3. COSpoc is designed for a p-processor shared-memory architecture. The algorithm

Algorithm 3: $COSpoc(c_0, \cdots, c_{l-1})$

Input: $c_0, \cdots, c_{l-1}$ // c_i=list of nodes in the i-th maximal clique in G
Output: $(k_{max} - 1)$ collections of disjoint sets F_k, $k \in [2, k_{max}]$, corresponding to the k-clique communities of G

```
1  begin
2      all processors q s.t. q ∈ [0, p − 1] do in parallel
           // Initialize collections of disjoint sets
3          foreach k ∈ [2, k_max] do
4              F_{q,k} ←< L_k singletons {0}, ··· , {L_k − 1} >
           // Extract k-clique communities
5          foreach i ∈ [0, l − 1] s.t. i mod p = q do
6              for j ← i + 1 to l − 1 do
7                  ov_{i,j} ← OVERLAP(c_i, c_j)
8                  foreach k ∈ [2, ov_{i,j} + 1] do
9                      MERGE_SETS(F_{q,k}, i, j)
       // Join partial results
10     foreach k ∈ [2, k_max] do
11         foreach q ∈ [1, p − 1] do
12             CONNECT_ME(F_{0,k}, F_{q,k})
13     return F_{0,k}, k ∈ [2, k_max]
```

takes as input $c_0, \cdots, c_{l-1}$, where c_i contains the list of nodes in the i-th maximal clique in G and $c_i \prec c_j$ iff c_i has a size greater than or equal to the size of c_j.

Immediately after the beginning, in line 2, p processors start their execution in parallel. At first, each processor q, $q \in [0, p-1]$, initializes $(k_{max} - 1)$ collections $F_{q,k_{max}}, \cdots, F_{q,3}, F_{q,2}$ on which it will be the only one to operate on. Collection $F_{q,k}$ has size L_k. Processor q uses $F_{q,k}$ to extract the connected components of a subgraph $G_{q,k} = (V_k, E_{q,k})$ of the clique-clique graph G_k. This subgraph has the same vertex set V_k of G_k and an edge set $E_{q,k} \subseteq E_k$ which is determined by the condition in line 5. Formally, $E_{q,k} = \{(i,j) \in E_k : q = i \bmod p \ \wedge \ j > i\}$. Processor q obtains the connected components of each subgraph $G_{q,k}$ as follows. First, it executes $OVERLAP(c_i, c_j)$[2] to obtain the number $ov_{i,j}$ of nodes in common between maxi-

[2] In practice, if c_i and c_i are represented as ordered vectors, this function can be efficiently implemented by performing a binary search on the larger vector for each element of the smaller one.

mal cliques c_i, c_j. Then, since c_i and c_j belong to the same k-clique community for each $k \in [2, ov_{i,j}+1]$, it merges disjoint sets containing i and j in $F_{q,2}, \cdots, F_{q,ov_{i,j}+1}$.

When each processor p terminates execution, connected components of the subgraphs $G_{q,k}$ are merged together in the loop starting at line 10. For each k, $F_{0,k}$ is updated with the connected components of $G_{q,k}$, $q \in [1, p-1]$, through $CONNECT\_ME(F_{0,k}, F_{q,k})$. Therefore, after the i-th iteration of the loop, according to Theorem 1, $F_{0,k}$ contains the connected components of a graph $\bigcup_{q\in[0,i]} G_{q,k}$. Now, by observing that the remainder of a division by p is always a number between 0 and $p-1$, each possible pair of maximal cliques is processed since we have p processors with indices $q \in [0, p-1]$. Hence, $\bigcup_{q\in[0,p-1]} G_{q,k} = G_k$ and $F_{0,k}$ contains the connected components of G_k after the completion of the loop starting at line 10. These connected components are equivalent to the k-clique communities of G.

3.6.1 Worst-Case Algorithm Complexities

In this section we analytically derive both worst-case time and space complexities of COSpoc. In order to ease analytic tractability we introduce the following assumptions: *i)* operations on collections of disjoint sets are in $O(1)$; and *ii)* perfect load balancing is achieved. The rationale behind *i)* is that the function α, which is used to give an upper on the cost of operations using the algorithm discussed in Sect. 3.5, actually grows *very* slowly and does not assume values greater than 4 for any practical input [182]. The rationale behind *ii)* is three-fold. First, the fine-grained row assignment achieved with the modulo operation. Second, the ordering in the input maximal cliques. Third, the structural properties of real-world networks in which maximal cliques with similar size generally have similar overlap values. In Sect. 3.8 we show that these apparently strong assumptions are in fact well-justified and supported by the experiments.

COSpoc worst-case time complexity is given in the following theorem.

Theorem 2. *If operations on collections of disjoint sets are in $O(1)$, perfect load balancing is achieved and overlap is calculated through binary searches, then $COSpoc(c_0, \cdots, c_{l-1})$ worst-case time complexity is in*

$$O\left(\frac{l^2}{p}k_{max}\log_2 k_{max}\right). \tag{3.1}$$

Proof. See Appendix A.2.

Although COSpoc time complexity is inversely proportional to the number of processors, the bound derived does not allow to analytically determine the speedup, i.e. the ratio between the execution time of the sequential algorithm and the execution time of the parallel algorithm. However, as we discuss in Sect. 3.8, we experimentally measured a *linear* speedup of the algorithm, which is as good as we can possibly hope for. Therefore, the assumption on the perfect load balancing is actually sound and well-supported by experiments.

Worst-case space complexity is given in the following theorem.

Theorem 3. $COSpoc(c_0, \cdots, c_{l-1})$ *worst-case space complexity is in*

$$O(p \cdot l \cdot k_{max}).$$

Proof. See Appendix A.2.

This complexity depends linearly on l, while in CPM this dependence is quadratic. The substantial reduction in the space required enabled COS to extract k-clique communities from real-world networks, such as those shown in Sect. 3.8. The advantages arising from the linear dependence of the space on l far outweigh the disadvantages arising from the linear dependence on p. In fact, $l^2 \gg p$ in any realistic case. The lack of dependence of CPM on p is due to the fact that it is not a parallel algorithm.

3.7 A Parallel Algorithm to Detect k-Clique Communities (on Steroids)

In this section we introduce CPM On Steroids (COS). Compared to the proof-of-concept COSpoc, in COS we drastically reduce the number of operations on collections of disjoint sets, by ensuring that MERGE_SETS is called at most one time for

each possible pair of maximal cliques. To achieve this improvement we: *i)* use a *sliding window* over the clique-clique overlap matrix; and *ii)* exploit the fact that k-clique communities are nested [67] — nested in the sense that each k-clique community is contained in one and only one h-clique community for each $h < k$.

Prior to illustrate COS, we provide a throughout description of the *sliding window*, which is exploited in the algorithm to ease cooperation between threads. This, in turn, leads to a drastically reduced number of operations on collections of disjoint sets.

3.7.1 A Sliding Window To Enable Thread Cooperation

The *sliding window* enables multiple threads to process a matrix as if it were wholly stored in main memory, while actually only a small chunk physically resides in memory. The idea of using a sliding window comes from the observation that when multiple flows of execution are available, the clique-clique overlap matrix can be used to facilitate cooperation between threads. The sliding window uses a fixed W-bytes-size buffer where it places the chunks. The size of the buffer is a user-configurable parameter. The name "sliding window" reflects the idea of a (tiny) window which we can slide over a huge matrix. By looking through this window we can only see a certain number of consecutive rows. If we want to see all the rows, we can (for example) *slide* the window from the beginning of the matrix down to the end. The principle is the same one could adopt for observing the whole sky through a telescope. One could point the telescope at a patch of the sky, then could move it to another patch and so on until she or he has explored the whole sky.

Actually, the window is slid over the *upper triangular part* of the matrix, which is enough to contain non redundant information on the overlap. Hence, The number of rows that can fit in the buffer is not constant. Let w be the number of bytes required by each element of the matrix. The maximum number of elements that can fit in the buffer, constant and known a priori, is $\eta = \lfloor W/w \rfloor$. Conversely, the maximum number of consecutive rows that can fit in it, let them have indices in the range $[s, e]$, depends both on η and s (or, equivalently, e). Assuming s is given, and indices range from 0 to $l - 1$ globally, we can determine e by solving

$$\sum_{j=s}^{x} (l - (j+1)) = \eta. \tag{3.2}$$

If we distribute the sum and use the formula for the sum of the first integer numbers, we can rewrite (3.2) as $-x^2/2 + x(l - 3/2) + (s-1)(s/2 - l + 1) = \eta$. This equation has two solutions for x: $x_1 = l - 3/2 - 1/2\sqrt{\Delta}$ and $x_2 = l - 3/2 + 1/2\sqrt{\Delta}$, where $\Delta = (2s - 2l + 1)^2 - 8\eta$. Solutions are real if *a)* $l \geq 2$ (i.e., the number of maximal cliques is greater than two); *b)* $\eta \geq l - 1$ (i.e., the buffer is sized to contain at least the largest row) and *c)* $0 \leq s \leq l - 1$ (i.e., the index s must be one valid matrix index). The index e can be determined as follows. If $x_1 < 0$, $e = l - 1$ because all the rows up to the last can fit in the buffer. If $x_1 = l - 2$ and $x_2 = l - 1$, $e = l - 1$. Otherwise, $e = \lfloor x_1 \rfloor$.
Each element (i, j) such that $i \in [s, e]$ and $j \in [i+1, l-1]$ can be located in the buffer at offset

$$w \left(\sum_{h=s}^{h=i-1} [l - (h+1)] + j - i - 1 \right). \tag{3.3}$$

According to the previous results, we can provide three simple functions. $SLIDE(s)$ which, given s as input, computes and returns e. $READ(i, j)$ and $WRITE(i, j, value)$, which provide read/write access to the elements with indices (i, j) such that $i \in [s, e]$ and $j \in [i + 1, l - 1]$. In order to provide access to the elements, these functions: *i)* use (3.3) to compute the offset of element (i, j); and *ii)* add the offset to the base address of the buffer in memory.

Figure 3.6 shows an example of the sliding window over the clique-clique overlap matrix of Fig. 3.2(b). The buffer has size $W = 8$ bytes and each element requires $w = 1$ byte: at most $\mu = 8$ elements can fit in it. After $e \leftarrow SLIDE(1)$ has been called, elements (i, j), $i \in [1, 2]$ and $j > i$ are mapped into the buffer, since the function returned $e = 2$. The $value$ of each element (i, j) has been written at the right offset through a $WRITE(i, j, value)$ and can be retrieved with $READ(i, j)$. $WRITE(i, j, value)$ and $READ(i, j)$, to properly locate the element in memory, compute the offset of (i, j) according to (3.3) and add it to the base address b. For example, the offset for ele-

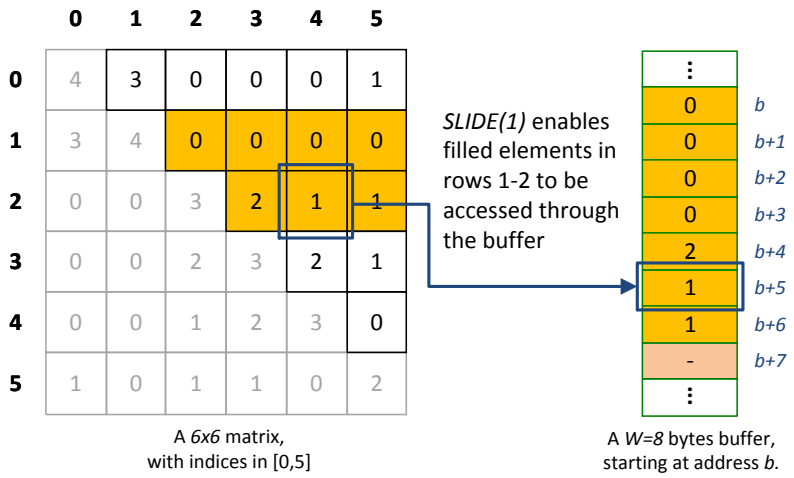


Fig. 3.6. The sliding window over the matrix of Fig. 3.2(b). The buffer has size $W = 8$ bytes and each element requires $w = 1$ byte.

ment $(2, 4)$ is 5, since $\sum_{h=s=1}^{h=i-1=1}[l-(h+1)]+j-i-1 = [6-(1+1)]+4-2-1 = 5$. The element can be located in memory at address $b + 5$.

With the sliding window, the $O(l^2)$ worst-case space complexity required for the clique-clique overlap matrix becomes $O(W)$, with W constant and user-configurable. Worst-case time complexity can definitely be neglected since the most expensive operation consists in solving a second order equation.

3.7.2 Algorithm Description

COS, which is designed for a p-processors shared-memory architecture, is described in Algorithm 4. It uses a sliding window (see previous section) to efficiently process the clique-clique overlap matrix in chunks of a configurable size. Moreover, COS leverages on k-clique community nesting [67] to drastically reduce operations on collections of disjoint sets.

COS takes as input $c_0, \cdots, c_{l-1}$, where c_i contains the list of nodes in the i-th maximal clique in G and $c_i \prec c_j$ iff c_i has a size greater than or equal to the size of c_j. At first, $(k_{max}-1)$ collections $F_{\text{global},k_{max}}, \cdots, F_{\text{global},3}, F_{\text{global},2}$ are initialized. Collection $F_{\text{global},k}$ has L_k elements and initially each element is in a disjoint singleton set. Processors will place their partial results in these global collections.

Algorithm 4: $COS(c_0, \cdots, c_{l-1})$

Input: $c_0, \cdots, c_{l-1}$ // c_i=list of nodes in the i-th maximal clique in G
Output: $(k_{max} - 1)$ collections of disjoint sets F_k, $k \in [2, k_{max}]$, corresponding to the k-clique communities of G

```
1  begin
       // Initialize set union data structures
2      foreach k ∈ [2, k_max] do
3          F_{global,k} ←< L_k singleton sets {0}, ··· , {L_k − 1} >
       // Extract k-clique communities
4      s, e ← 0
5      while e < (l − 2) do
6          e ← SLIDE(s)
7          // Overlap Computation
8          all processors q, q ∈ [0, p − 1] do in parallel
9              foreach i ∈ [s, e] s.t. i mod p = q do
10                 for j ← i + 1 to l − 1 do
11                     ov_{i,j} ← OVERLAP(c_i, c_j)
12                     WRITE(i, j, ov_{i,j})
13         // Overlap Processing
14         all processors q, q ∈ [0, p − 1] do in parallel
15             k ← k_max
16             F_q ←< l singletons {0}, ··· , {l − 1} >
17             while k > 1 do
18                 foreach i ∈ [s, e] s.t. i mod p = q do
19                     for j ← i + 1 to L_k − 1 do
20                         ov_{i,j} ← READ(i, j)
21                         if ov_{i,j} = (k − 1) then
22                             MERGE_SETS(F_q, i, j)
23                             WRITE(i, j, 0)
24                 CONNECT_ME(F_{global,k}, F_q)
25                 k ← k − 1
26         // Update s
27         s ← e + 1
28     return F_{global,k}, k ∈ [2, k_max]
```

After the initialization of global collections, the sliding window comes into play. The upper triangular part of the clique-clique overlap matrix is processed in chunks, starting from the first row. Rows in each chunk are mapped into the buffer in line 6 through $SLIDE(s)$. These chunks map consecutive rows since the index s is always updated with $(e + 1)$ in line 27. Furthermore, the whole upper triangular part is pro-

cessed because the *while* cycles until e has reached the last index $(l-2)$. Hence, the overlap between each possible pair of maximal cliques is processed.

For each chunk, parallel operations are divided into two blocks. Two barriers are introduced at the end of each block (just before lines 13 and 26). For $CONNECT\_ME$ — and only for it in the whole algorithm — mutually exclusive execution must be guaranteed.

In the first parallel block, starting at line 8, the $OVERLAP(c_i, c_j)$ is computed for *each* pair of maximal cliques c_i, c_j, $i \in [s, e]$ and $j \in [i+1, l-1]$ — for each i always exists one and only one processor $q \in [0, p-1]$ such that $q = i \bmod p$. When the overlap is computed, it is written to the buffer. Write operations are performed simultaneously since no two processors ever write to the same location.

In the second parallel block, starting at line 14, the overlap is analysed in order to extract the connected components of the clique-clique graphs G_k, according to a strictly decreasing order of k, i.e. from k_{max} down to 2. More precisely, for each chunk, processor q uses the collection F_q to update the connected components of a subgraph of $G_{k_{max}}$. Then, it exploits the information already encoded in F_q to update the connected components of a subgraph $G_{k_{max}-1}$, and so, on until G_2. This information can be exploited in accordance to the theorem in [67]. The theorem guarantees that each k-clique community is contained in one and only one h-clique community, $h \in [2, k]$, implying $G_k \subseteq G_{k-1}$ for each k.

Let us now discuss in more detail the operations each processor q performs in the second parallel block. Given a k, q reads the overlap between a subset of pairs of maximal h-cliques, $h \geq k$ — these are the only maximal cliques that belong to k-clique communities (see Def. 2). This subset is determined by the condition in line 18. If $ov_{i,j} = (k-1)$, maximal cliques c_i and c_j belong to the *same* k-clique community and, hence, sets in F_q containing i and j are merged and *the overlap value is zeroed*. After the overlap has been processed for each possible pair of maximal h-cliques, $h \geq k$, F_q contains disjoint sets corresponding to the connected components of a subgraph of the clique-clique graph G_k. This subgraph is on the same nodes of G_k but has edges $(i, j) \in E_k$ s.t. $i \in [s, e]$, $j > i$, and $q = i \bmod p$. Since the union of these subgraphs over all the chunks and over all the processors produces G_k, their connected components have to be merged together in order

to obtain the connected components of G_k — merging is accomplished by calling $CONNECT\_ME(F_{\mathrm{global},k}, F_q)$. Once the merging has been done the while continues with a k decreased by one.

The zeroing of the overlap value, which is performed simultaneously since no two processors ever write to the same location, significantly speeds up operations. Indeed, it *avoids doing more than one merging* for each pair of maximal cliques. We can still obtain the connected components of any clique-clique graph, since k values are processed in decreasing order. The relation $G_k \subseteq G_{k-1}$ enables us to re-use the same F_q to progressively include information on the connected components of clique-clique graphs at gradually smaller k values.

After all the chunks have been processed, $F_{\mathrm{global},k}$ contains the connected components of G_k, $k \in [2, k_{max}]$ which are equivalent to the k-clique communities of G. COS worst-case complexities are given in Appendix A.2.

3.8 Experimental Results

3.8.1 Experimental Setup and Input Data

We implemented COSpoc and COS in C, in a freely and publicly available software [66]. Maximal Cliques were listed using the open-source implementation of the (serial) Bron-Kerbosh (BK) algorithm available in the `igraph` library [42]. For parallel programming we used the standard POSIX Threads[3]. We used a CPM implementation available in CFinder[4] [148] and a python SCP implementation retrieved from `http://www.lce.hut.fi/~mtkivela/kclique.html`. The machine on which we ran the experiments has four Intel Xeon processors E7-4850[5] and 128 GB RAM. It runs a GNU/Linux Operating System (OS) with a kernel Linux 3.0.6.

Graphs used in the experiments, together with the type of complex system they model, their references and their number of nodes $|V|$ and edges $|E|$ are reported in Tab. 3.1. LINX graph was obtained according to [71] whereas the others were retrieved from [14] and [113]. All the graphs were considered undirected, unweighted,

[3] POSIX.1c, Threads extensions (IEEE Std 1003.1c-1995)
[4] Experiments were carried out with version 2.0.5, 64 bit
[5] 24M Cache, 2.00 GHz, 10 cores, 20 threads, HT capable

Graph	Type	Ref.	$\|V\|$	$\|E\|$
LINX	Autonomous Systems	[71]	345	14,188
NDwww	Web	[14]	325,729	1,090,108
SFwww	Web	[93]	281,903	1,992,636
CAquery	Web	[97]	9,664	15,969
Yeast	Protein Interactions	[31]	2,361	6,646
NetSci	Collaboration	[135]	1,589	2,742
Erdos	Collaboration	[14]	6,927	11,850
Geom	Collaboration	[91]	7,343	7,343
Amazon	Product co-purchase	[114]	403,394	2,443,408
AstroPh	Collaboration	[115]	18,772	198,050
CondMat	Collaboration	[115]	23,133	93,439
HepTh	Citation	[115]	27,770	352,285
EmlEnron	Communication	[116]	36,692	183,831

Table 3.1. Graphs used in the experiments

without isolated nodes and without self- and multiple-edges. Table 3.2 reports the total number of maximal cliques l in each graph, their maximum size k_{max}, their average size $\mu = l^{-1} \sum_k k \cdot l_k$, their variance $\sigma^2 = l^{-1} \sum_k l_k (k - \mu)^2$ and the number l_2 of maximal cliques with size 2. In addition a fine estimation $\tilde{s}$ of the size of the clique-clique overlap matrix CPM has to build is reported. This estimation was computed as the square of the number of maximal cliques with size strictly greater than 2, assuming that a byte is used for each element. With this estimation it is possible to know, a priori, which graphs can be processed by CPM on our 128 GB memory machine.

3.8.2 Experiments

Comparison of COS and CPM

CPM runtime memory footprint is shown and compared with that of COS in Fig. 3.7. NDwww graph was used as input since it is the graph requiring the greatest amount

Graph	l	k_{max}	μ	σ^2	l_2	$\tilde{s}$ [GB]
LINX	384,494	34	23.01	11.29	1	113.11
NDwww	495,947	155	3.15	1.49	294,706	37.72
SFwww	1,055,936	61	7.00	5.65	108,831	835.4
CAquery	17,548	10	1.92	0.18	15,660	> 0.01
Yeast	5,012	9	2.45	0.39	3,644	> 0.01
NetSci	741	20	2.88	0.94	349	> 0.01
Erdos	9,210	8	2.47	0.32	6,503	0.01
Geom	5,817	22	2.68	0.85	3,167	0.01
Amazon	1,023,572	11	3.82	0.75	264,874	536.09
AstroPh	36,428	57	6.87	3.88	2,236	1.09
CondMat	18,502	26	3.95	1.01	3,888	0.2
HepTh	464,873	23	7.71	1.99	15,466	188.1
EmlEnron	226,859	20	8.08	1.36	14,070	42.17

Table 3.2. Features of the graphs used in the experiments

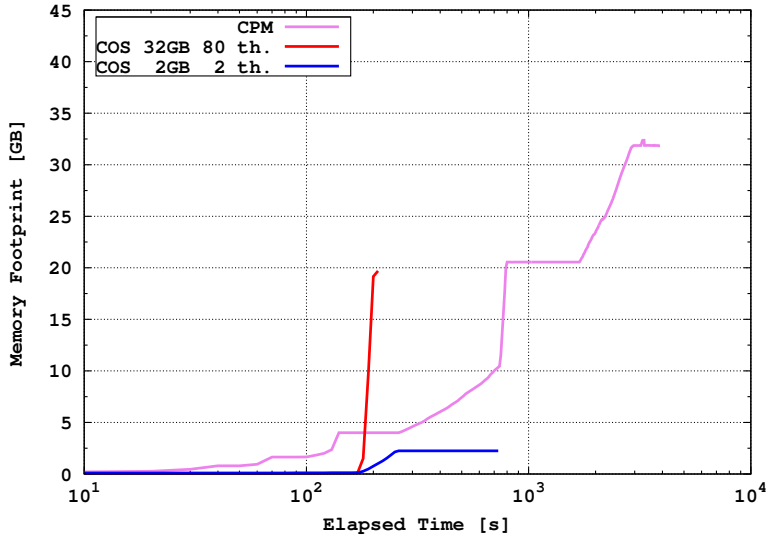


Fig. 3.7. Runtime memory footprint of CPM and COS on NDwww.

of memory on which CPM executed without errors on our hardware. We decided to plot runtime memory footprint for two configurations of COS: 80 threads with a 32GB sliding window buffer; and 2 threads with a 2GB sliding window buffer. The aim of the former configuration is to demonstrate the suitability of COS in high performance

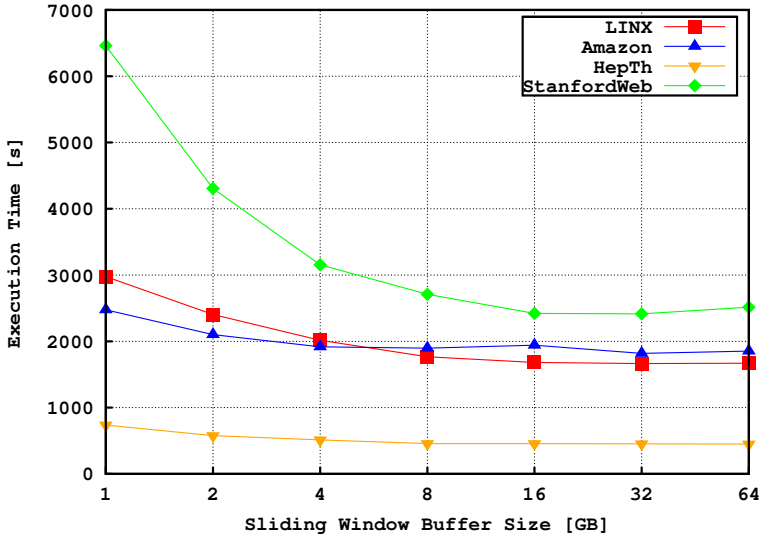


Fig. 3.8. Execution time of COS versus sliding window size.

systems, whereas the aim of the latter is to give evidence of the effectiveness of COS even on standard machines. For "COS 2GB 2 th." in the figure, maximum memory footprint is, at most, approximately equal to the size of the sliding window buffer. This suggest that the upper bound on the worst-case space complexity derived may be considered W in practice. For "COS 32GB 80 th." memory footprint is approximately 20GB — i.e., about one half of the estimated clique-clique overlap matrix size for NDwww. This arises because the sliding window processes only the upper triangular part of that matrix. From this observation it follows also that the 32GB buffer, which is not fully used, enables the sliding window to process the whole upper triangular part in only one chunk. Finally, we note also that COS is approximately 20 (6) times faster than CPM when configured with 80 (2) threads.

The Impact of the Sliding Window Buffer Size on the Execution Time

We performed this experiment with the aim of determining how changes in the sliding window buffer size W impact on the execution time. We executed COS several times, starting with $W = 1$ GB, and doubling this size until $W = 64$ GB. We ran COS with 80 threads since this quantity corresponds to the maximum number of processors visible to our OS. This number does not equal the maximum physical degree of par-

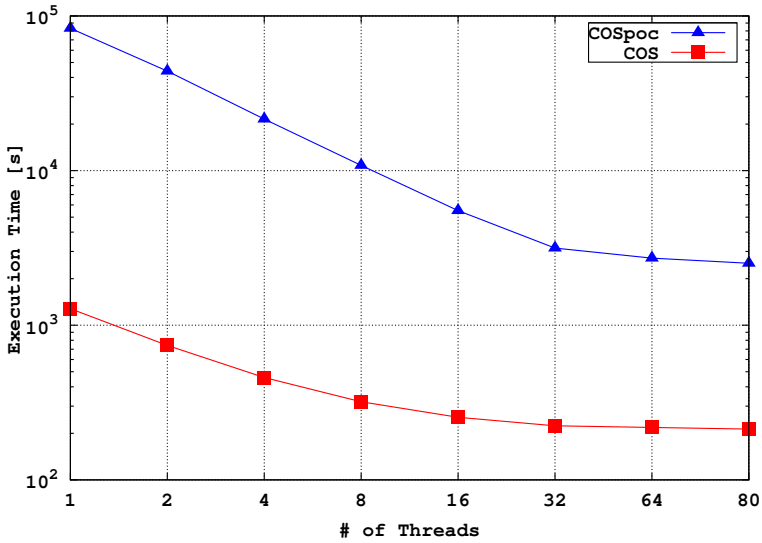


Fig. 3.9. Execution time comparison of COSpoc and COS on NDwww.

allelism, which is lower, but can still optimize OS behaviour on our hyper-threading capable hardware. As inputs, we chosen graphs for which COS requires to place more than one chunk in the buffer — i.e. graphs whose estimated clique-clique overlap matrix size is always greater than twice the W. As shown in Fig. 3.8, the execution time decreases as the buffer size increases. This can be explained by the lower the number of mutually exclusive operations that need to be made on global collections of disjoint sets (line 24 of COS).

Nevertheless, while this reduction is significant in the range 1-4 GB, it is much less pronounced for sizes greater than 8 GB. In the range 16-64 GB the execution time is minimized, and almost constant values suggest that COS has low sensitivity to buffer sizes in the latter range. The size $W = 32$ GB is chosen as the default sliding window buffer size for the subsequent experiments.

Comparison of COS and COSpoc

In this experiment we compare COSpoc and COS execution time, versus number of threads, on the graph NDwww. We demonstrate that the techniques introduced in COS dramatically improve the overall performance. Results are shown in Fig. 3.9. Execution time reductions achieved with COS are extremely important, ranging from

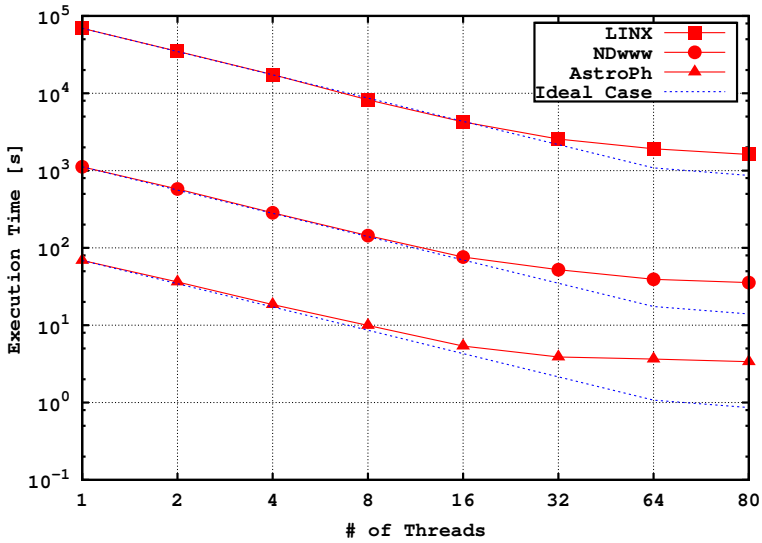


Fig. 3.10. Execution time of COS versus number of threads.

one to two orders of magnitude. We can also observe that COSpoc execution time decreases exponentially in the range from 1 to 32 threads. Therefore, in this range the algorithm achieves a linear speedup — which is the best one could hope for. This means that doubling the number of threads actually leads to an halving of the execution time. In other words, the algorithm has a linear speedup. Since the maximum physical degree of parallelism of our machine is 40, it is worth noting that it is impossible to experiment linear speedups for 64 and 80 threads.

COS execution time decreases less than exponentially with the increase of the number of threads. This is because plotted values include the time to extract maximal cliques. This time, although negligible if compared with COSpoc total execution time, is comparable with that of COS and therefore it does not allow to achieve a linear speedup. In the next section we exclude maximal cliques listing time and show that also COS achieves a linear speedup.

The Impact of Number of Threads on the Execution Time (speedup)

In Fig. 3.10 we show the execution time of COS versus an exponentially increasing number of threads. Plotted values do not include the time required to list maximal cliques with the BK algorithm. LINX, NDWWW and AstroPh were chosen as inputs

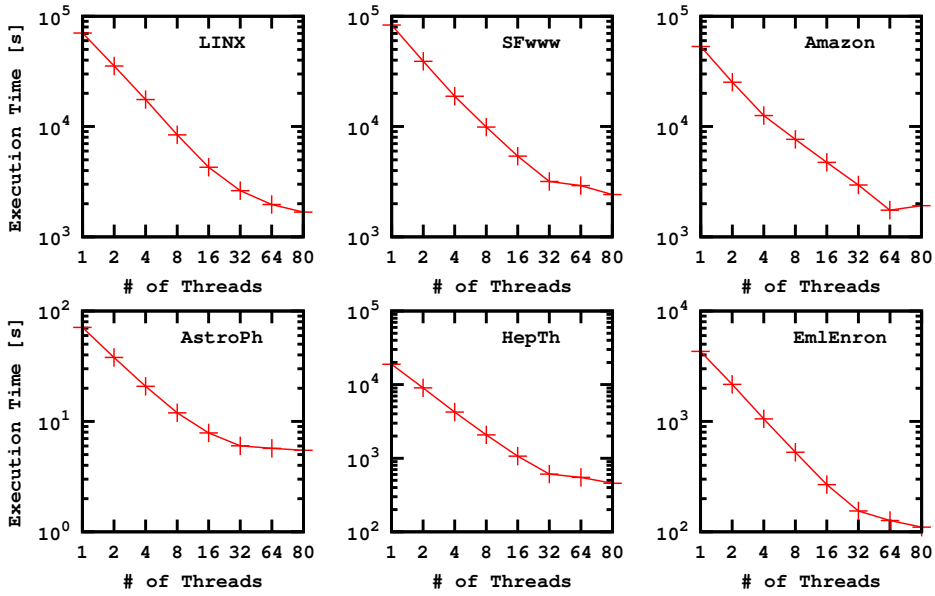


Fig. 3.11. Execution time of COS versus number of threads.

since their execution times always differ for at least one order of magnitude, regardless of the number of threads. For each input we also draw a dashed line, which represent the ideal case where doubling the number of threads halves the execution time. It is worth noting how COS reaches — or is very close to — the ideal case, at least up to 32 threads. The distance from the ideal case that is experienced for 64 and 80 threads is due to the fact that our hardware physical degree of parallelism is 40. This is evidence that COS speedup on our machine is *linear* with the number of physical cores available. Additional evidence is provided with the following experiments.

In Fig. 3.11 we show the time COS took to execute on graphs for which it was not possible to run CPM on. In particular, it was not possible to execute CPM on SFwww, Amazon and HepTh due to their clique-clique overlap matrix size, exceeding the amount of memory available on our hardware. Conversely, despite matrix sizes of LINX, AstroPh and EmlEnron would allow CPM to run, we (on LINX and AstroPh after two days) or the OS (on EmlEnron after 12 hours) stopped the execution. Values plotted include the time taken by serially extracting maximal cliques. Even with the inclusion of this time, COS continues to achieve a very good speedup. Hence — as also stated in Sect. 3.2 — maximal cliques listing time is negligible if compared with

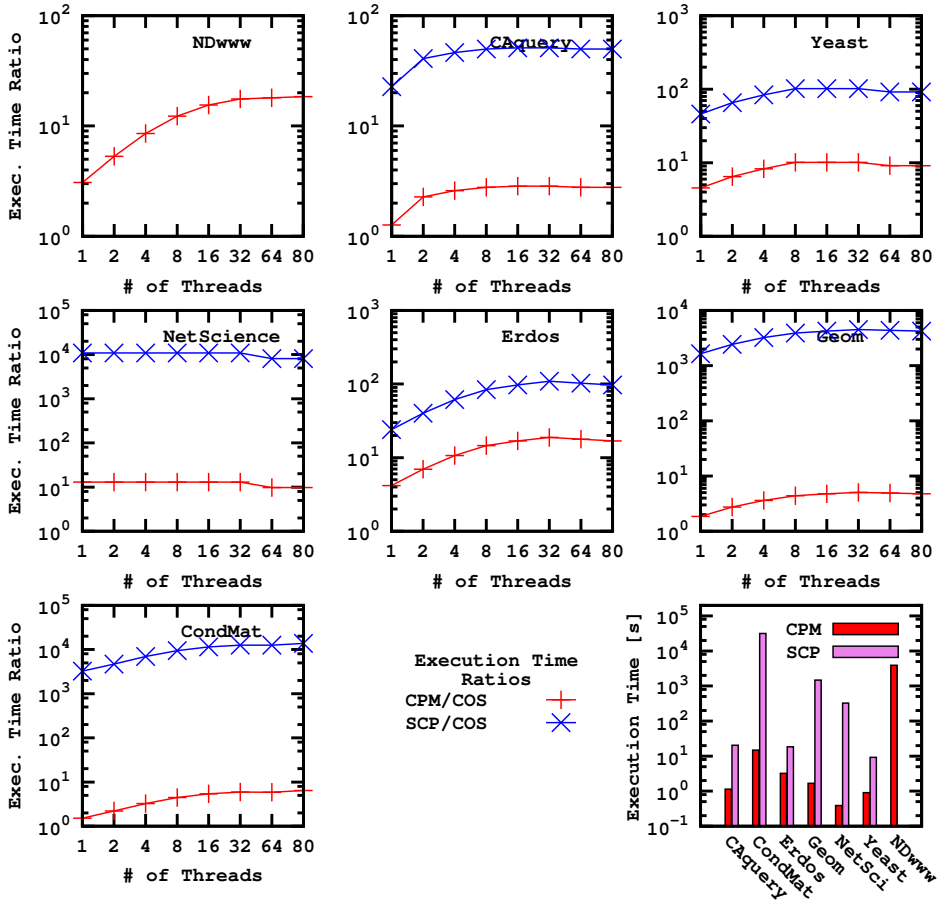


Fig. 3.12. CPM/COS (SCP/COS) execution time ratio and execution times of CPM and SCP.

the total time. Again, the exponential decrease of the execution time up to 32 threads provides evidence on the linear speedup of COS — that is, the best that one could hope for.

Comparsion of COS with The State-of-The-Art

In Fig. 3.12 we compare COS, CPM and SCP execution times. On our hardware, we were able to execute successfully CPM on NDwww, CAquery, Yeasy, NetSci, Erdos, Geom and CondMat. We were able to run also SCP on all these graphs except NDwww. Since SCP is designed to extract k-clique communities for a given k, we obtained its execution time by summing the times it takes to extract k-clique

communities for each possible value of k. Execution times of both CPM and SCP are given in the bottom right corner of Fig. 3.12. In the other plots of the same figure, we show the values of two execution time ratios, namely: CPM/COS (in red); and SCP/COS (in blue). For each number of threads in the x-axis, we computed the ratio CPM/COS (SCP/COS) by dividing the execution time of CPM (SCP) with that of COS, executed with the corresponding number of threads.

These ratios, which are always greater than 1, reveal that COS is always faster than both CPM and SCP. In particular, it is always more than 10 times faster than SCP on any input, even when run with 1 thread. By increasing the number of threads, we see that it becomes 100 to more than 1,000 times faster. Best performance is obtained for CondMat where COS terminate its 80-threaded execution in one 10,000th the time it takes SCP. Very good execution time reductions are experienced also with reference to CPM. In this case, COS is few times faster that CPM for single-threaded executions, but becomes 10 to 20 times faster when the number of threads is increased. Although these reductions are important in both cases, absolute execution times are too small (only NDwww takes more than 10 seconds) to enable the identification of a clear link between number of threads and COS execution time variations. Finally, with this experiment we can say that SCP, although designed to overcome its drawbacks, it is actually slower than CPM and differences in their execution time always exceed the order of magnitude. In the next experiment we compare COS with other state-of-the art algorithms, detecting communities different from the k-clique communities. For the comparison we carefully selected 6 of the best-performing algorithms available in the literature [106]. We observed that COS performance is as good as the fastest algorithms on some graphs, even when it is not executed in parallel. Performance degradations are observed on graphs with an extremely high number of maximal cliques with large sizes.

Comparison of COS with Other Community Detection Algorithms

Many community detection algorithms have been proposed so far in the literature. While they have already been subjected to strict tests with the aim of evaluating their performance [106], their efficiency has not yet been compared. Here we carry out a

comparative analysis of the efficiency of some state-of-the-art community detection algorithms and compare them with COS. All algorithms considered are thoroughly discussed in Sect. 2.3, and their main features are summarized in Tab. 2.1. Here, for the sake of readability, we briefly recall them. The first two algorithms chosen, *infomod* and *infomap*, were proposed by Rosvall and Bergstorm [164] [166]. The third algorithm *blondel* was proposed by Blondel et al. [20]. We also selected *kdense*, that is solely based on the topological properties of a graph (Saito et al. [169]). Finally, we chosen *oslom* and *gce*, that are able to detect *overlapping* community structure (Lancichinetti et al. [110] and Lee et al. [112]).

Since some of the algorithms considered are unacceptably slow, an upper bound in the execution time is necessary to complete the experiments in a reasonable amount of time. However, we avoided choosing fixed, arbitrary bounds. Rather, we carefully determined the maximum execution time to assign to each algorithm for each input graph. For the calculation of these timeouts we exploited the already measured execution times of COS. More precisely, we selected for each input the time COS took to complete is execution with 8 threads. In fact, this is the time COS should take to complete its execution on today's multi-core personal computers. We set the following timeouts: LINX (8,417.37 s), NDwww (320.16 s), SFwww (9,880.97 s), Amazon (7,648.42 s), AstroPh (11.97 s), HepTh (2,072.37 s) and EmlEnron (526.81 s). We did not set them for CAquery, CondMat, Erdos, Geom, NetSci and Yeast since all the algorithms completed quite fast on these small graphs.

Algorithms *infomod*, *infomap*, *blondel*, *oslom* and *gce* are non-deterministic. This means that both their outcome and their execution time varies unpredictably due to their intrinsic stochastic elements. For this reason we decided to plot average, maximum and minimum values across 10 independent runs. On the contrary, we plotted the exact values obtained from a single run of both COS and *kdense*, which are deterministic.

In Fig. 3.13 we show algorithms average, maximum and minimum execution time. Input graphs are those for which we did not set upper bounds on algorithms execution time. To ease the comparison we plotted COS execution time as an horizontal rule. This time is relative to the 1-threaded execution of COS, i.e. it is the worst-case execution time, achieved only when COS is *not executed* in parallel. We decided

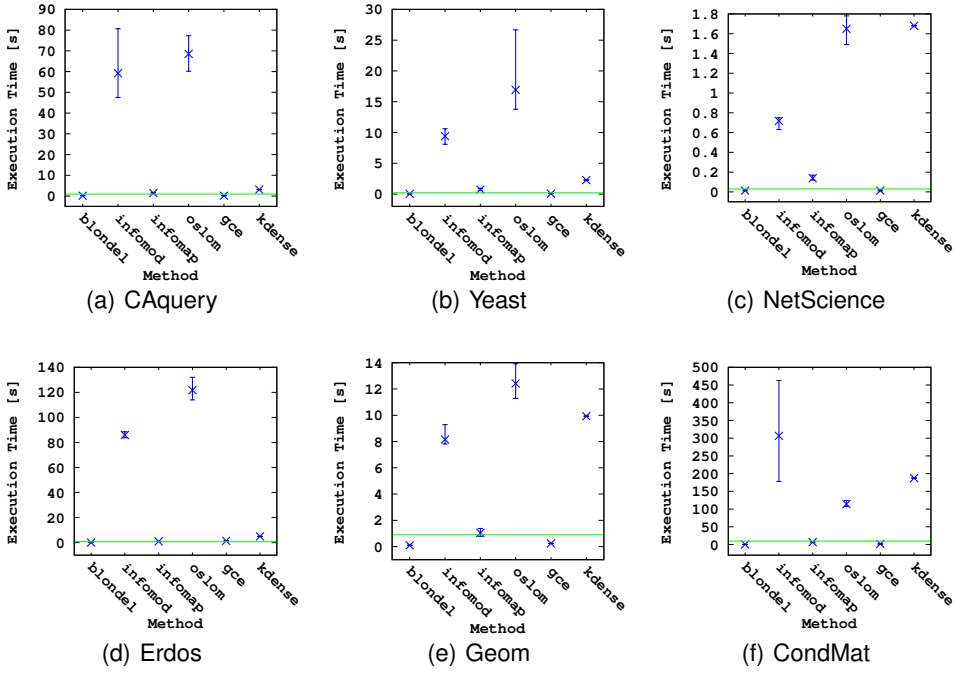


Fig. 3.13. Algorithms average, maximum and minimum execution time. Input graphs are those for which we did not set an execution timeout. COS execution time is plotted as an horizontal rule.

to plot these worst-case values to enable meaningful comparisons with the other algorithms, which are not parallel. By studying Fig. 3.13, we see that *infomod* and *oslom* have the worst performance. They are tens to hundred times slower that all the other algorithms. In addition, a great variability in their execution times was observed among different runs. Also *kdense* does not perform well and its performance is similar to that of *oslom* and *infomod* on NetScience, Geom and CondMat. All the other algorithms run in negligible time. They all terminate in less that 1 second on any input. By observing the horizontal rule indicating COS execution time, we see that it is definitely one of the best performing algorithms. In fact, COS performance is as good as the fastest state-of-the-art community detection algorithms on these graphs, *even if it is not executed in parallel*.

In Fig. 3.14 we show algorithms performance on inputs for which we set execution time upper bounds. Algorithms average, maximum and minimum values are plotted. COS best- and worst-case execution times experimented are plotted as horizontal

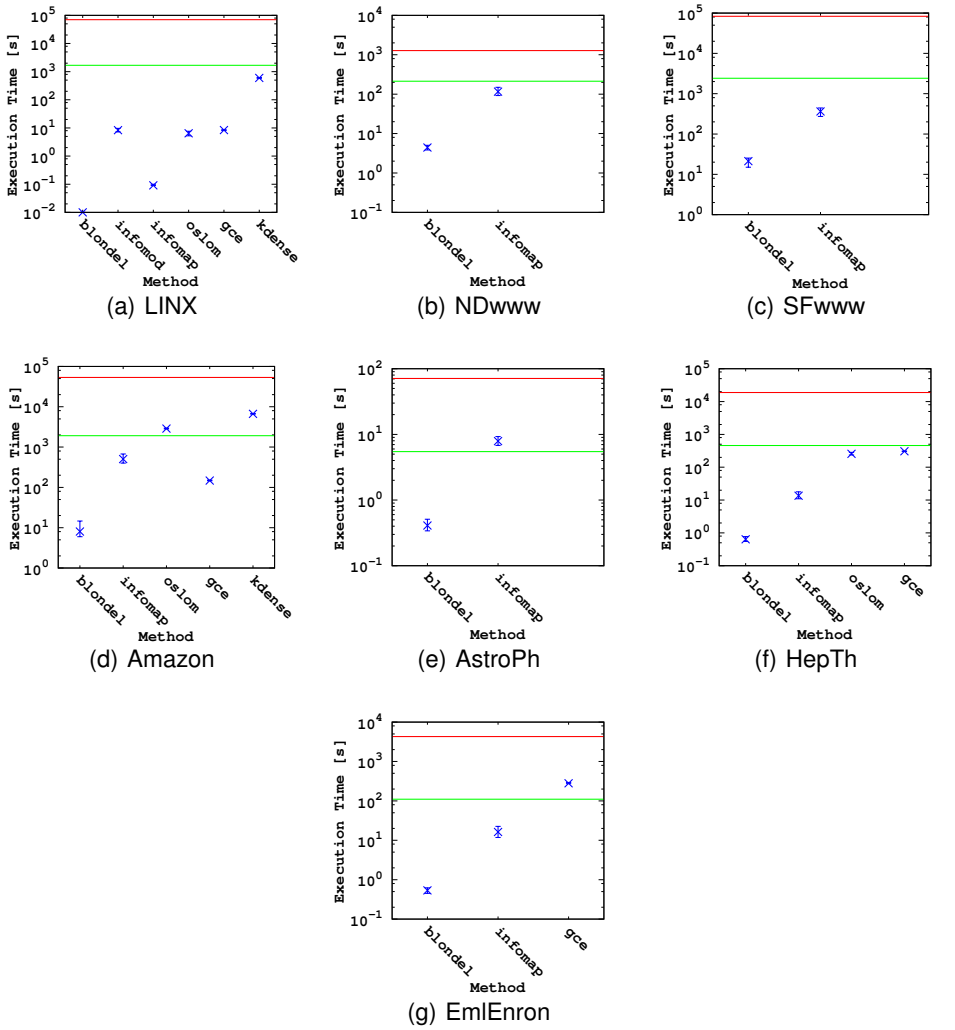


Fig. 3.14. Algorithms average, maximum and minimum execution time. Input graphs are those for which we set an execution timeout. For each graph, only algorithms faster than COS are shown. COS maximum and minimum execution times are plotted as horizontal rules.

rules. These values were obtained by running COS with 80 threads and 1 thread respectively. We want to emphasize values are shown only for algorithms faster than COS executed with 8 threads. This does not mean that they are always faster. In fact, many of them are *slower than* COS when its degree of parallelism is increased — this can be seen from the absence of slow algorithms from the x-axes in the figure.

Graph	*infomod*	*infomap*	*blondel*	*kdense*	*oslom*	*gce*
LINX	†	+	+	+	+	+
NDwww	†	+	+	-	-	-
SFwww	†	+	+	-	-	-
Amazon	†	+	+	+	+	+
AstroPh	-	+	+	-	-	-
HepTh	-	+	+	-	+	+
EmlEnron	-	+	+	-	-	+

†Execution terminated anomalously with exception `std::bad_alloc`

Table 3.3. Algorithms faster "+" (slower "-") than COS executed with 8 threads

blondel and *infomap* algorithms perform best with these additional graphs too (see Fig. 3.14). They are the only two algorithms able to terminate faster than COS (run with 8 threads on each input). In particular, *blondel* turns out to be always the fastest algorithm, achieving execution times which are orders of magnitude less than the others. It does not take more than 20 seconds to terminate on every input and it is the only algorithm which is always faster than COS. Conversely, *gce* does not exhibit very good performance as in Fig. 3.13 and it is always at least one order of magnitude slower than *blondel* and *infomap* on all the inputs except for Amazon. The algorithms which perform worst are still *infomod*, *oslom* and *kdense* in these cases too. For example, *infomod* was slower than COS for AstroPh, HepTh and EmlEnron. It was not even able to execute successfully on LINX, NDwww, SFwww and Amazon. As a summary, we report in Tab. 3.3, for each input, which algorithms executed faster (slower) than COS run with 8 threads.

Here we discuss the reasons why COS is significantly slower than every other algorithm on LINX. LINX is a small graph in terms of number of nodes and edges but has a huge number of maximal cliques. By observing Tab. 3.2 we see that this number is comparable with that of the largest graphs in terms of nodes/edges. Moreover, LINX has the highest average maximal clique size, which is more that twice the average size on each other graph. Differences in the execution time arise from the fact that COS is the only algorithm which has maximal cliques as the basis of its community discovery mechanism. Maximal cliques are so relevant in COS that we

expressed both its worst-case time and space complexities in terms of them. However, disadvantages arising from an higher execution time are well balanced by the quality of the communities extracted. Indeed, most of the other algorithms are not able to discover significant community structure on LINX since they end up in grouping all the nodes in a single (or, at most, in a few) community. On the contrary, k-clique communities extracted by COS better capture the extremely-overlapping and nested community structure of this graph.

We can conclude the discussion by observing that COS is not always able to achieve the lowest execution times on these inputs. However, it is important to emphasize that it is always able to execute successfully. While conducting the experiments, we also monitored runtime memory footprint of the algorithms. We observed that they do not require more than a few GB, except for the *kdense*, which uses approximately 25 GB while running on Amazon. However, its demand reduces significantly on the other inputs. In fact, if we exclude Amazon, it requires at most 1.8 GB for SFwww. Another memory-demanding algorithm is *infomod*, which requires approximately 3 GB for HepTh and 5 GB for EmlEnron. The other algorithms does not use an amount of memory exceeding significantly 1 GB for any graph.

3.9 Discussion and Conclusion

In this chapter we addressed the problem of extracting k-clique communities in parallel from real-world networks, such as the Internet. We theoretically analysed the existing Clique Percolation Method (CPM), highlighting its scalability issues. The identification of these scalability issues enabled us to design and develop CPM On Steroids (COS) algorithm. COS efficiently extracts k-clique communities, with low memory requirements and has an unbounded, user-configurable degree of parallelism. Analytic tight upper bounds on COS execution time and space requirements, providing strong evidence about its efficiency, are presented as well. A key role in COS is played by the innovative CONNECted ComponenTs MErging (CONNECT_ME) algorithm. With this technique we can obtain the connected components of a network, even if it has previously been split into and arbitrary number of subnetworks that could be processed in parallel. Through extensive experiments run on

real-world network data, we showed that COS has a linear speedup and constantly outperform all the other state-of-the-art k-clique community detection algorithms in terms of both space requirements and execution time. In our opinion, it should be the algorithm of choice for k-clique communities extraction aiming at very high performance and low resource requirements. As a future work we plan to extend the design of COS for a message-passing architecture and to investigate its performance on mega-scale networks such as Wikipedia, Facebook and Twitter.

4

Network-Based Methodologies to Study the Internet

4.1 Networks, Internet Companies, and Stock Markets: When Technology meets Business

4.1.1 Introduction

The nature of Autonomous Systems (AS) in the Internet is twofold. On the one hand, they are collections of switches and routers intra- and inter-connected via physical links and logical sessions. On the other hand, they are well-established companies that follow complex business strategies to be competitive within the same industry. Although these two natures could seem incommensurable at a first sight, they are actually closely related. Indeed, business strategies entail developing and implementing enterprise policies and plans. The implementation, in the Internet ecosystem, usually consists in operating routers and physical links or in establishing Border Gateway Protocol (BGP) sessions with providers or peers. BGP, the *de facto* standard for Internet traffic exchange, allows companies to finely tune their in- and out-bound traffic according to contracts signed with other companies. Therefore, we argue the existence of a strong mutual coupling between these natures. Strategic management determines changes in the physical links and logical sessions of switches and routers. In turn, the latter connectivity changes affect present and future business strategies. Here, we aim at taking a step forward by linking the two natures.

We investigate synchronous cross correlations between stock price variations and AS-level connectivity features such as the degree or the clustering coefficient.

The AS-level network is an abstract representation of the economically-driven interconnections between ASes, which need to cooperate in order to stay on the market. We focus on the AS-level network since we believe it best captures the dynamics underlying inter-AS economic relationships. As thoroughly discussed in the introductory chapter of this thesis, at this level all the routers and links operated by a single AS are collapsed into one single node and only inter-AS links are retained. These links, corresponding to BGP sessions, are always established according to some kind of economical agreement [139] [138].

Synchronous cross correlations are quantified by means of the Pearson correlation coefficient. A metric space is defined for the investigated stocks and AS-level features, ensuring that the stronger the correlation, the closer the elements in the space. A hierarchical organization in this space is detected through a clustering procedure able to extract an ultrametric space from it. We emphasize the hierarchical organization by means of a Minimum Spanning Tree (MST), which provides a meaningful topological arrangement of stocks and AS-level features. We show that this methodology allows to isolate groups which make sense from an economic point of view and provides valuable information on the factors behind the evolution of the Internet ecosystem.

Our contribution can be summarized as follows. We find that groups of companies homogeneous with reference to their service offering – e.g. transit providers – are positively correlated in the stock market. Similarly, we find that even geographically close companies are positively correlated, suggesting the existence of common economic factors driving geographically homogeneous companies. In addition, the topological arrangement obtained through the MST can be used to derive a meaningful taxonomy of the ASes. New evidence on the factors underlying the AS-level network time evolution is given by combining its properties with stock market data. We highlight the existence of factors, common to all ASes, able to drive the evolution of *global* features. Other factors, specific for each AS, determine strong correlations between *local* features. We also show that factors governing AS stock price variations are not the same as those synchronously driving the variations of AS-level features.

4.1.2 Related Work

Great interest and dedication has been shown so far in the analysis and modeling of the Internet AS-level network. An intense research activity has begun to emerge after the seminal works [56] and [2]. Analyses and models strongly depend on measurement data provided by projects such as IRL, CAIDA Ark and DIMES. Analyses (e.g. [125] [65] and references therein) rely on measurements to draw meaningful conclusions on the structural properties of the AS-level network. Models (see [194] for an accurate review an evolutionary comparison) rely on them for validation. Unfortunately, the ability to accurately map the AS-level network was shown to be fraught with difficulties and dangers [167]. Difficulties are encountered for example when detecting certain kinds of BGP sessions [40] [140] or when inferring the physical devices belonging to each AS [37]. Dangers are due to the "*as-is*" use of available measurement data as good proxies of the real underlying AS-level network [99].

To overcome these obstacles, researchers designed and deployed novel measurement infrastructures [55] [5], with the aim of providing an increasingly more accurate and detailed view of the network and its features. However, to the best of our knowledge, stock market data have never been used to augment or refine the knowledge we have of the AS-level network. Beginning with the pioneering work [128], such data has successfully been used to study and find *topological* arrangements of economically-principled networks and hence we believe it may provide valuable insights also into the AS-level network structure and evolution. Similarly, although (anti-)correlations have been observed among neighboring AS degrees [152] [151], to the best of our knowledge *cross* correlations between time-evolving AS-level connectivity features have never been studied before. We believe they may be relevant for a better understanding of the complex techno-socio-economic factors underlying the Internet. In addition, they may contribute significantly to the design of novel evolutionary or predictive models. Among the economically-principled models we mention the works [194] and [121]. In [194] the authors assume that AS wealth is the result of a multiplicative stochastic process and keep the degree of each AS proportional to its wealth. In the agent-based model proposed in [121] ASes optimize their cost-based fitness function according to provider or peering strategies.

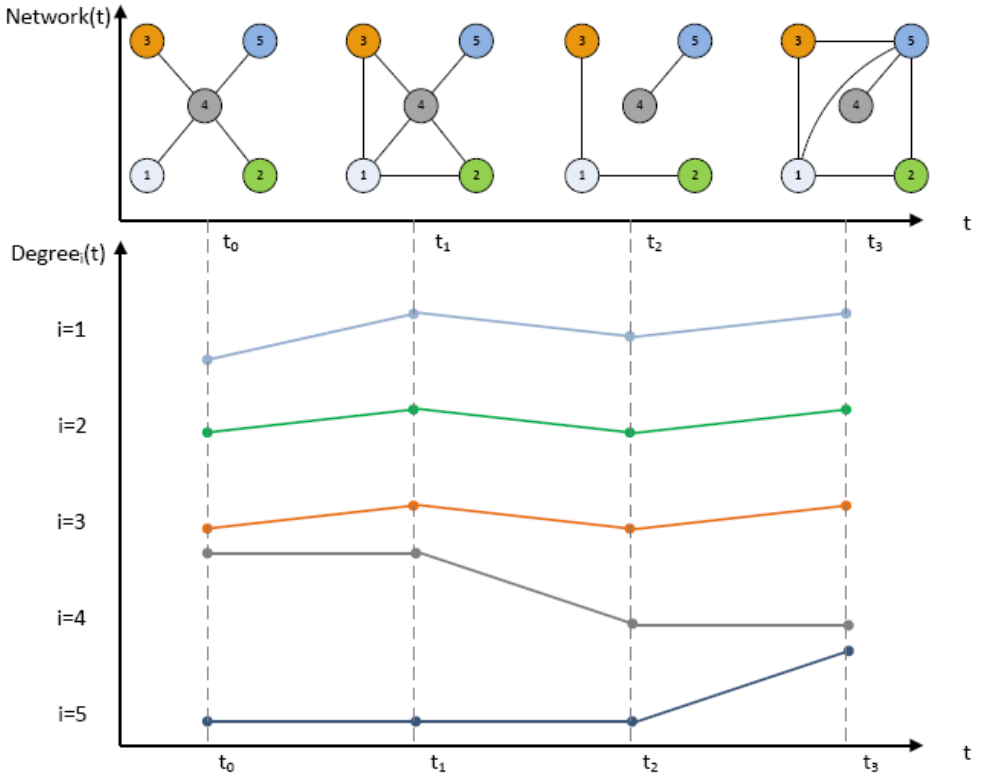


Fig. 4.1. A network sampled at four different times $t_0, \cdots, t_3$ (top) and time series associated to the degree of each node $i = 1, \cdots, 5$ (bottom). By looking at the shapes of such time series it is possible to qualitatively assess the correlations between nodes – the Pearson correlation coefficient is used for quantitative assessments. Indeed, the first three nodes undergo very similar variations and this will result in high values of the correlations coefficients.

4.1.3 Methodology

In general, given a feature P_i of node i (e.g. a connectivity feature such as the *degree*, as well as the stock price), we keep track of i's changes in time with reference to that feature via the time series

$$\hat{p}_i(t) = \ln P_i(t + \Delta t) - \ln P_i(t), \tag{4.1}$$

where Δt is a relative time. Then, we adjust the changes to a common scale as

$$p_i(t) = \frac{\hat{p}_i(t) - \hat{\mu}_i}{\hat{\sigma}_i}, \tag{4.2}$$

where $\hat{\mu}_i$ is an average over time and $\hat{\sigma}_i$ the standard deviation of $\hat{p}_i(t)$. We then determine correlations between adjusted changes using the bivariate Pearson correlation coefficient

$$c_{i,j} = \mathrm{E}\{p_i(t)p_j(t)\}, \tag{4.3}$$

which ranges from -1 (maximum anti-correlation) to 1 (maximum correlation) and is 0 when changes are uncorrelated. We arrange this correlation coefficients to form a correlation matrix $\mathbf{C} = [c_{i,j}]$. In Fig. 4.1 we give an example of how we characterize and keep track of nodes' degree changes in time. Intuitively, nodes 1, 2 and 3 are very similar in terms of their degree evolution in time and this will turn into high correlation coefficients between them.

Unfortunately, the correlation coefficient does not represent a distance function for any Euclidean space. Therefore, it cannot be used either to build up a hierarchy or to arrange features in a topological space. Hence, we use the distance function $d(i,j) = \sqrt{2(1-c_{ij})}$, which defines a metric space by fulfilling the three axioms of an Euclidean distance [127]. We obtain the hierarchical organization by extracting a ultrametric space [160] from the metric one. Practically, this is achieved via the single-linkage clustering procedure [173], which disposes features on the branches of a unique hierarchical tree. Single-linkage clustering is an ascending, bottom-up aggregation procedure. Initially, each node feature is in a different branch and, at each step, the two closest branches are aggregated into one larger branch. Distance between two branches is the minimum distance between any feature of one branch and any feature of the other. We also construct the MST connecting features in the metric space, in order to emphasize their hierarchical organization and to arrange them in a topological space. The MST – which alone contains all the information for carrying out single-linkage clustering [64] – gives an alternative way to highlight hierarchies among the investigated features.

4.1.4 Investigated Companies

In this thesis we focus our attention on a subset of large, publicly traded companies all over the world owning at least an AS. Typically, all these companies offer a rich

Company	Service	H.Q.	Ticker Symbol	Market
AT&T	*t1tp*	NA	T	NYSE
Verizon	*t1tp*	NA	VZ	NYSE
Sprint	*t1tp*	NA	S	NYSE
Inteliquent	*t1tp*	NA	IQNT	Nasdaq
CenturyLink	*t1tp*	NA	CTL	NYSE
Deutsche Telekom	*t1tp*	EU	DTE.DE	XETRA
Telecom Italia	*t1tp*	EU	TIT.MI	Milan
Telefonica	*t1tp*	EU	TEF.MC	Madrid
TeliaSonera	*t1tp*	EU	TLSN.ST	Stockholm
NTT	*t1tp*	A	NTT	NYSE
Level3	*t1tp*	NA	LVLT	NYSE
TATA Comm.	*t1tp*	A	TATACOMM.NS	Bombay
Cogent	*tp*	NA	CCOI	Nasdaq
TW Telecom	*tp*	NA	TWTC	Nasdaq
Akamai	*cdn*	NA	AKAM	Nasdaq
Limelight	*cdn*	NA	LLNW	Nasdaq
Rackspace	*cdn*	NA	RAX	NYSE
InterNAP	*cdn*	NA	INAP	Nasdaq
Equinix	*ixp*	NA	EQIX	Nasdaq

Table 4.1. Companies considered in the study

portfolio of Internet services. However, each one has a main service which can easily be recognized by looking at its history and activity. Hence, we based our selection on the main service offered and chosen: 14 large IP transit providers; 4 content delivery networks; and 1 internet exchange point. An IP transit provider (*tp*) carries IP traffic, enabling paying customer ASes to reach the whole Internet. If a *tp* has full, free-of-charge Internet reachability, then is termed Tier-1 (*t1tp*). A content delivery network (*cdn*) serves content (e.g. web and multimedia objects) to end-users with high availability and high performance. Content providers pay *cdn*s to better distribute their content among users. An Internet exchange point (*ixp*) is a physical facility that en-

ables Internet companies to directly exchange their traffic, without paying for transit. *ixp*s are mainly used by companies with the aim of reducing their costs by bypassing *t1tp*s. In Tab. 4.1 we list selected companies and indicate the geographical location of their headqurters – Europe (EU), North America (NA) or Asia (A). In addition, we report their main service offered, the ticker symbol identifying them in the stock market and the stock market where stocks are traded. Autonomous System Numbers (ASNs) are the following: AT&T (7018), Verizon (701), Sprint (1239), Inteliquent (3257), CenturyLink (209, 3561), Deutsche Telekom (3320), Telecom Italia (6762), Telefonica (12956), TeliaSonera (1299), NTT (2914), Level3 (3356, 3549, 1), TATA Communications (6453), Cogent (174), TW Telecom (4323), Akamai (20940), Limelight (22822), Rackspace (15395), InterNAP (11855) and Equinix ([many]). AS-level topologies are generated using the data available from the Internet Research Lab (IRL) website[1] – outliers in data are discarded using the Chauvenet's criterion [13].

To study companies under investigation, we consider both the stock price and 5 AS-level connectivity features. We are aware that the stock price is an aggregate economical indicator and that other indicators may be able to capture company's situation in more detail – e.g. revenue, sales and investments. Nevertheless, while such indicators are often difficult to obtain, the stock price is publicly available and to some extent condenses in a nutshell several aspects of a company. In the present study we focus on a time span from January 2008 to September 2012. However, we observed that different time spans do not lead to significant changes in the results. We retrieved historical stock closure prices data from Yahoo![2]. *Monthly* (rather than daily) synchronous cross correlations are considered when combining stock market data with AS-level connectivity features – closure prices are averaged on a monthly basis. Due to the incompleteness and the errors affecting AS-level topologies, a daily study of cross correlations would appear to have little meaning. AS-level connectivity features considered are the following:

- *Degree (de):* Is the number of BGP sessions an AS established with other ASes, i.e. with its *neighbors*.
- *Average neighbor degree (knn):* Is the average degree of the neighbors of an AS.

[1] `http://irl.cs.ucla.edu/topology/`
[2] `http://finance.yahoo.com/`

- *Clustering coefficient (cc):* Quantifies how close the neighbors of an AS are to being a clique, i.e. a complete graph.
- *Eigenvector centrality (ei):* Is a measure of the importance of an AS. It assigns relative scores to ASes based on the principle that connections to high-scoring ASes contribute more to the score of the AS in question than equal connections to low-scoring ASes.
- *Coreness (co):* A k-core is a maximal subgraph of the AS-level network in which each AS has at least degree k. If an AS belongs to the k-core but not to the $(k+1)$-core, then is said to have coreness k.

We refer the reader to Chapter 2 for a thorough description of the aforementioned features. Selected features are able to capture: direct AS connectivity – *de*; connectivity patterns in the neighborhood of the AS – *knn* and *cc*; and global connectivity features – *ei* and *co*. Indeed, while *de* simply accounts for the number of neighbors an AS has, *knn* and *cc* also tell relevant information on neighbors' BGP connectivity. Specifically, *knn* is an average indicator of the propensity of neighboring ASes to establish BGP sessions with the rest of the network. Similarly, *cc* gives information on the attitude of neighbors in establishing BGP sessions with each other. Since such features involve only neighboring ASes, we also included global features *ei* and *co* to quantify the role of each AS in the whole network. As further discussed in the next section, they both take into account network-wide BGP connectivity features.

4.1.5 Results

Autonomous Systems Stocks Hierarchical Organization

In Fig. 4.2 we show the MST highlighting the hierarchical organization of investigated stocks – the lower the distance between two companies, the higher and thicker the link connecting them. A first inspection of the MST suggests the existence of two geographically homogeneous groups: Europe (top-left with TEF) and North America (top-to-bottom-right with T). North American companies can be further divided into two smaller-but-stronger subgroups bridged through the link T-EQIX: *t1tp*s and *tp*s with T on the one hand, and *cdn*s with EQIX on the other. Apparently, geography

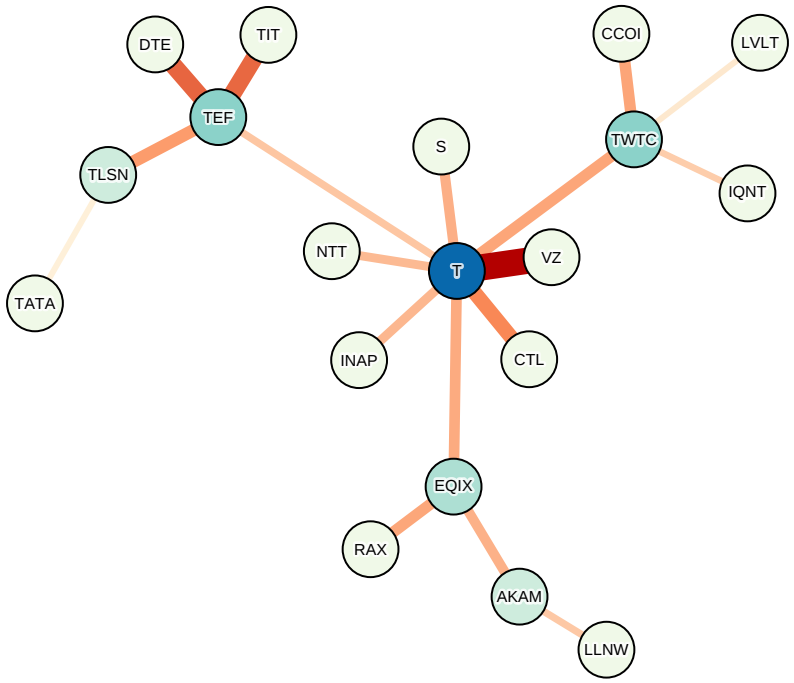


Fig. 4.2. Autonomous Systems stocks minimum spanning tree

seems not to play a significant role for Asiatic companies TATA and NTT, which connect to the EU and the NA groups, respectively. Actually, its relevance is once again confirmed by looking at their geographical location. Indeed, TATA is headquartered in India, which is much closer to EU rather than NA. Similarly, NTT, which is headquartered in Japan, is much closer to NA rather than EU. The absence of a well-defined Asiatic group may be due to the small number of Asiatic companies investigated. The hierarchical tree of the subdominant ultrametric associated to the MST is shown on the left side of Fig. 4.3. On the right side we visually represent correlation coefficients using colors by means of an heatmap. Mappings between colors and correlation values are reported in the top-left corner of the figure. In the same corner we also plot an histogram highlighting the correlation coefficients distribution. The lower the correlations a group of stocks has with others, the higher the distance at which the branching occurs in the hierarchical tree. For the sake of example we can consider

the branch involving TEF, DTE, TIT and TLSN. It departs early from the rest of the tree and in fact the heatmap highlights very low to no correlations with companies not belonging to the branch. Practically, a branch which departs from the tree at a high distance suggests that the involved companies are subject to *common* economic factors and that these factors do not affect companies outside the branch. To rephrase succinctly, it suggests the existence of economic factors which are *specific* only to companies in the branch. Likewise, when a branching occurs at low distance values – e.g. when VZ and T split into two distinct branches – companies involved are not only subject to *common* economic factors each other. They also have economic factors in *common* with other companies that departed earlier from the same branch – e.g. CTL.

A detailed inspection of the MST and of the branches of the associated hierarchical tree enable to identify two strongly correlated groups in the hierarchy:

- European Tier-1 transit providers (TEF, DTE, TIT, TLSN);
- North American Tier-1 transit providers (T, VZ, CTL).

Such groups correspond to dark diagonal blocks in the heatmap, which in turn map into strongly connected parts of the topological arrangement obtained through the MST. Discovered groups cluster together *t1tp* companies even if (first group) their stocks are traded in different markets. In addition we observe that any company in such groupings is also a telecommunications operator. Once again, we stress on the other striking feature of these groupings, i.e. their geographical homogeneity.

Two less strong groups correspond to:

- North American large, non-Tier-1 transit providers (TWTC, CCOI);
- North American *ixp*s and content delivery service providers (EQIX, RAX).

Companies in these smaller groups have less pronounced correlations. Nevertheless, they are topologically close to similar companies in terms of service offerings and headquarters location. Indeed, TWTC and CCOI are close to the north American *t1tp* in the topological arrangement obtained with the MST. This is reasonable if we look at the historical debates about their role as *t1tp*s or simply *tp*s in the Internet. Similarly, EQIX and RAX, are close to north American *cdn*s AKAM and LLNW. EQIX, which is a well-established *ixp*, has tens of datacenters all around the world enabling

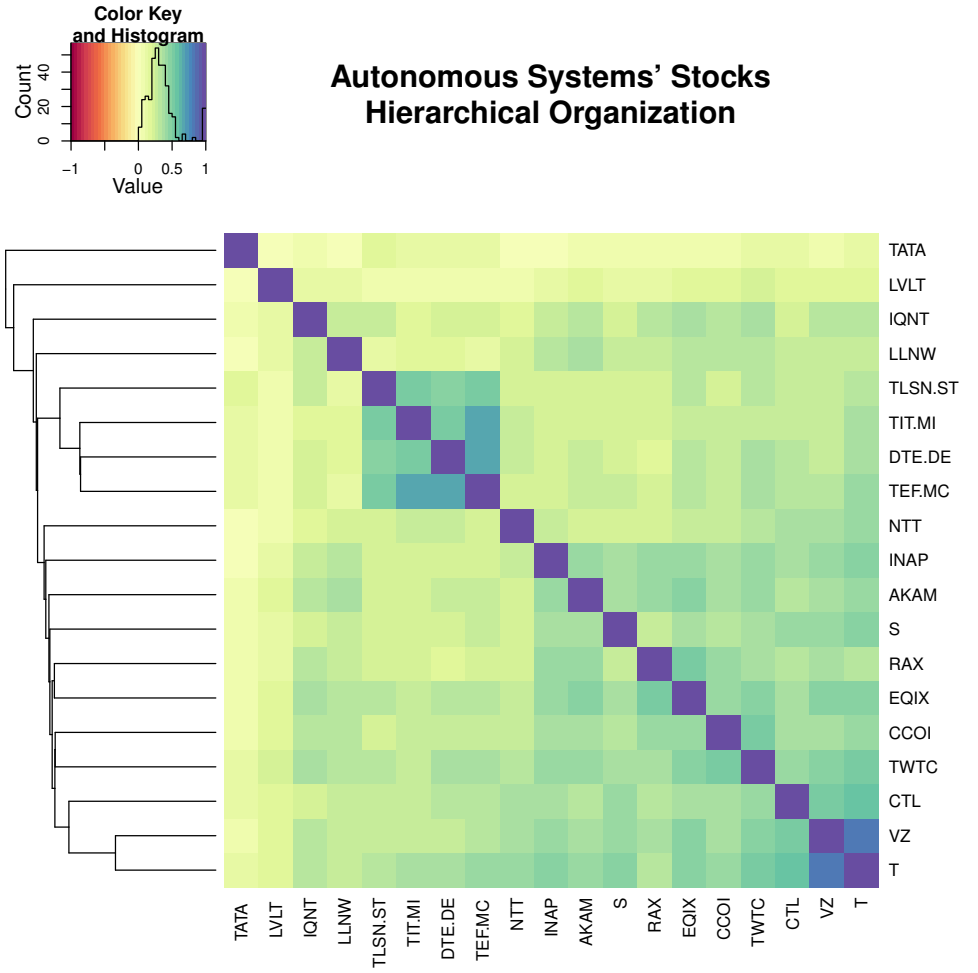


Fig. 4.3. Autonomous Systems stocks hierarchical tree

it to offer also *cdn* services. A similar explanation can be given for INAP, which is connected to *t1tp*s rather than *cdn*s. Indeed, as also pointed out in its services portfolio, it strongly relies on *t1tp*s to distribute contents in the Internet and also offers transit services on its own.

To sum up, the observed groups are meaningful from an economic standpoint since they are composed of companies homogeneous with respect to service offering and geographical location. This empirical evidence suggests the existence of common economic factors driving the synchronous time evolution of geographically homogeneous companies. Additionally, within the same location, companies offer-

ing similar services undergo to the same economic factors, which have a service-specific and service-exclusive nature. In contrast, very low to no correlation is found between geographically heterogeneous companies, suggesting that economic factors vary significantly among different countries. Finally, the ability of the MST and the hierarchical tree in isolating homogeneous groups suggests their use in deriving meaningful AS taxonomies.

Combining AS-level Topological Properties with Stock Prices

In Fig. 4.4 we show the MST obtained by combining stocks and connectivity features. Nodes are labeled with abbreviated, dot-separated company and property names. Space constraints do not allow us to show the hierarchical tree. An inspection of the MST highlights two distinct kinds of groups:

- Large groups, heterogeneous with respect to the company but strongly homogeneous with respect to the property;
- Small, single-company groups of heterogeneous features (clustering coefficient and average neighbor degree).

Large groups are three and emphasize the presence of synchronous cross correlations among heterogeneous companies with reference to their variations in: stock price (top-right yellow group); coreness (green group in the center); and eigenvector centrality (left star-like purple group). The latter three features are *global*, uncontrollable and almost completely independent of the single company. They depend on the whole Internet ecosystem. For example, stock prices are influenced by global market trends – their fluctuations do not depend only on the single company. Similarly, coreness and eigenvector centrality depend on the whole AS-level network, and not on the single AS or on its neighborhood. An AS cannot control its eigenvector centrality (coreness) since it strongly depend on the centrality (coreness) of its neighbors, which in turn depends on the centrality (coreness) of their neighbors, and so on. Therefore, empirical evidence suggest the existence of common, *ecosystem-wide factors*, that cause simultaneous similar variations of global features among all the Internet companies. For this reason, we argue that the very nature of these factors is embedded in the Internet ecosystem as a whole and not in smaller sub-parts of it. In

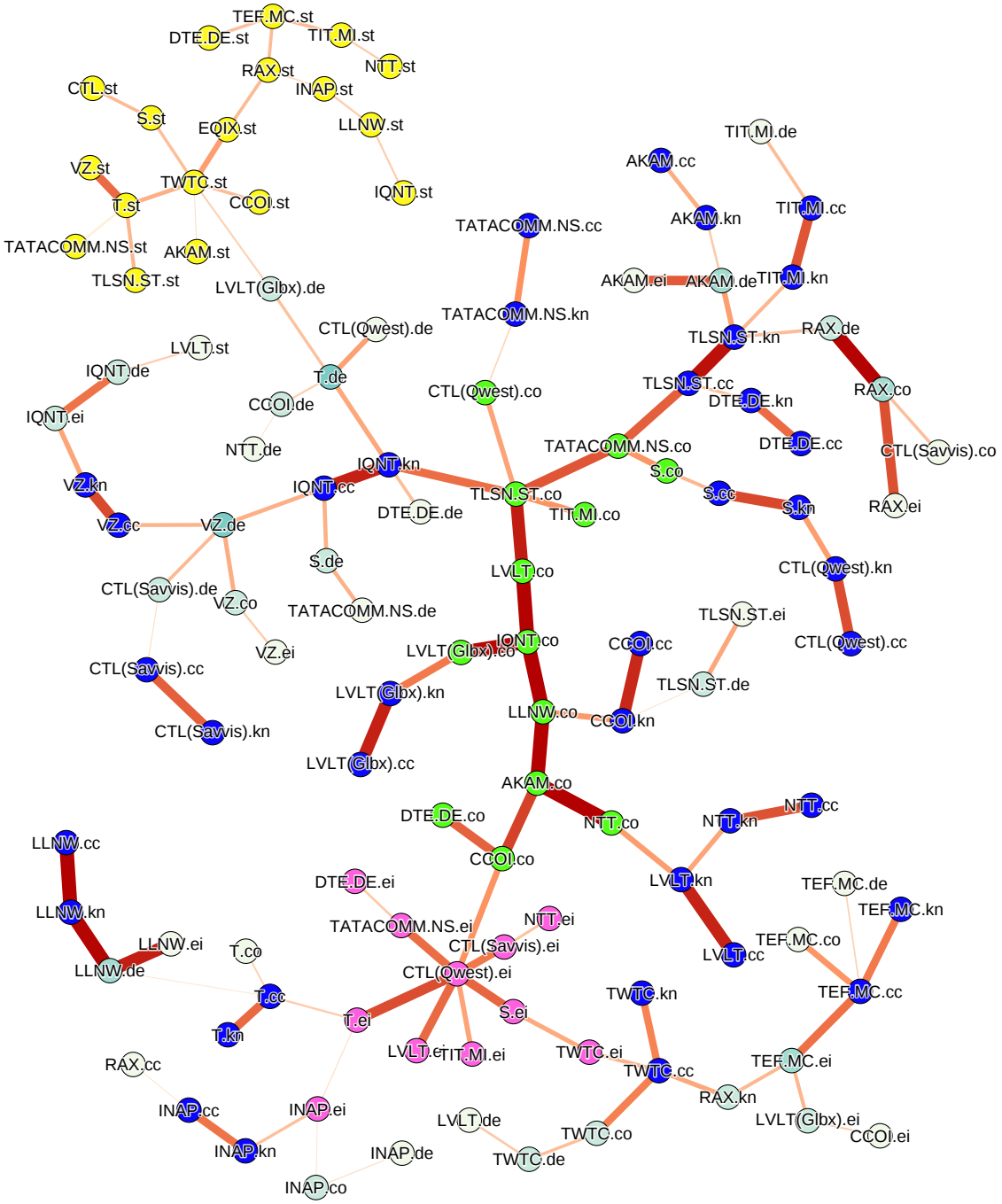


Fig. 4.4. Autonomous Systems stocks and connectivity features minimum spanning tree

addition, we observe that global factors underlying stock price variations are not the same as those governing neither the coreness nor the eigenvector centrality – no to very low correlation is measured.

Small, single-company groups (in dark blue) always capture strong correlations between clustering coefficient (*cc*) and average neighbor degree (*knn*) for *each* investigated AS, except for RAX. Strong correlations are highlighted by the thick, dark edges directly connecting *cc* and *knn*. Much lower correlations of these features are observed among different ASes. Therefore, empirical evidence supports the existence of *company-specific factors* determining per-AS *independent* neighbor selection processes. Indeed, there is a virtually zero chance that over time ASes choose (or are chosen by) either the same neighbors or neighbors with similar BGP connectivity features. If two ASes established BGP sessions with the same neighbors (or with neighbors having similar connectivity features), they would have same values for *cc* and *knn* and maximum positive cross correlation. It follows, therefore, that the aforementioned company-specific factors not only yield independent neighbor selection processes, they also cause each AS to establish BGP sessions with different ASes. In other words, there is a negligible chance that investigated ASes choose (or are chosen by) the same neighbors in the whole AS-level network.

4.1.6 Conclusion and Future Directions

In the present section we investigated synchronous cross correlations between stock market data and AS-level connectivity features. We found that groups of companies headquartered in the same location and offering similar services tend to be strongly correlated, suggesting that they are subject, in a statistical way, to the same economic factors. We also discussed on the existence and nature of common factors underlying the evolution in time of AS global and local connectivity features.

We believe our novel approach provides valuable insights for example for designing new predictive or evolutionary AS-level models, as well as for validating existing ones. A model may take our results into account in order to design mechanisms able to rewire/grow/shrink the AS-level network in a way that cross correlations are preserved where necessary. We observed that stock market data cannot be used to

infer *synchronous* variations in connectivity features. Nevertheless, our study paves the way for a fascinating list of new scientific questions, among which: “What if we consider cross correlations as functions of the time lag?”, “Current stock market data can predict future trends in AS-level connectivity features (or vice versa)?”, “Extending the set of publicly traded Internet companies may lead to new insights into market or AS-level dynamics?”, “What if we extend the set of ASes, selecting for example large content providers such as Google or Amazon?”, “What if we exploit aggregate indices such as the S&P500 or the gross domestic product of countries?”, “May other per-company indicators (e.g. revenue, sales) be used to gain further insights?”.

4.2 Network Models of the Internet DNS Traffic

4.2.1 Introduction and Related Work

The Domain Name System (DNS) is a an essential component of the Internet used to associate symbolic host names with numeric IP addresses. Internet service providers often perceive the DNS as a core system they must keep up and running as their customers rely on it, but being it a service that does not bring revenues, they do not usually invest much on it. The consequence is that ISP's DNS servers, also known as resolvers, are sometimes slow in responses [41], and this has opened the market to public DNS servers such as OpenDNS and Google Public DNS. Beside premium services, such public DNSes offer the service at no cost while making revenues through advertisements, web traffic redirection and mining of DNS data.

Although the DNS is perceived as a critical infrastructure [33], all publicly available DNS traffic monitoring tools [154] [54] focus only on aggregate values such as the type and number of queries received by a DNS server [144]. Research and academia have focused on DNS for the purpose of identifying malicious activities [9] [17] [57], managing large DNS infrastructures [36], understanding how DNS server selection and caching works in reality [199] [145], and modeling its infrastructure in order to predict how DNS traffic will change under specific conditions [201]. Unfortunately, there is a lack of specific models for DNS traffic [1] [30]. Such models, can be of fundamental importance for understanding patterns, or identifying malicious activities as well as trends and interests in the Internet.

Internet trends and interests are the focus of Periodic reports such as Google Zeitgeist [63] and Akamai State of the Internet [177]. They contain various types of information such as the number of Internet users, top queries on search engines, popular hashtags on social networks and percentage of spam emails per day. Although the DNS can potentially be a good source of data for understanding Internet usage [80], publicly available reports [176] [143] focus only on the number of registered domains per Top Level Domain (TLD), DNS servers performance, or aggregated query reports, without disclosing information about Internet usage and trends. Methods for scoring web pages [28] have been out for years, and are profitably used by search engines to return searches sorted according to web page ranking. Similar

methods recently appeared also for DNS [75] [180] [76] although to date there are no public DNS traffic reports based on such methods. Ranking Internet domains is needed to generate detailed traffic reports focusing on popular domains, and report about the trends and interests related to Internet domains.

Driven by the aforementioned motivations, in this section we present the following original contributions. Firstly, we propose novel methodologies to produce network-based models of the DNS traffic. Secondly, leveraging on these models, we develop novel ranking methodologies for Internet domains.

More precisely, we demonstrate, by means of a large-scale validation at ".it" country code Top Level Domain (ccTLD) DNS servers, that our contributions pave the way to:

1. Define domain rankings according to their popularity among resolvers and vice versa.
2. Identify the most popular resolvers so that it is possible to change traffic policies with the aim of providing these resolvers a lower response time. This can be achieved for example by minimizing the Round Trip Time (RTT) between the authoritative name servers and the resolvers using them.
3. Unveil domains inter-relationships. Are Internet domains fully independent or can they be clustered based on user interests or economical relationships? Groups of similar domains (e.g. e-commerce sites) can be used as a market indicator for speculating how a given market sector performs over time.
4. List resolvers that are likely to misbehave (e.g. do not obey to the Time To Live (TTL) specified for domains they are sending queries for) and that thus need to be monitored more closely as they might perform malicious activities.
5. Rank domains according to the traffic type (e.g. web and email), countries where resolvers are located, density of queries according to the time of the day (e.g. a domain that receives queries according to the Italian working hours is likely to identify a company/individual that is interesting only for domestic users and not a global player).
6. Identify resolvers that might be used by email spammers, and domains that are likely to be targets of email attacks.

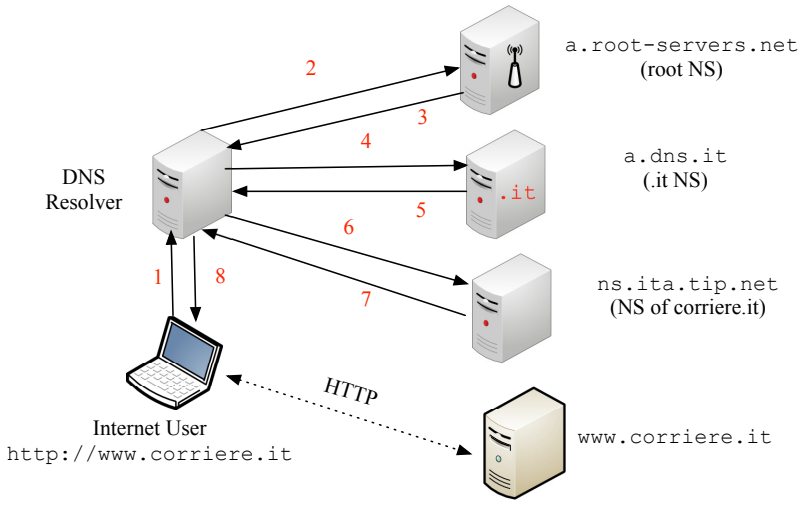


Fig. 4.5. Steps in DNS resolution of `www.corriere.it`

4.2.2 Understanding The DNS System

Iterative Domain Name Resolution

The domain name system is based on a hierarchical distributed architecture used to map domain names to resource records containing various types of data including numeric IP addresses (A record for IPv4 and AAAA for IPv6), names (NS record) and mail exchange servers (MX records) [8]. The DNS resolver is the client side of the DNS system, responsible for performing address resolution by starting from the top of the hierarchy (i.e., the *root zone*). The root zone is served by 13 root name servers, most of which with anycast addressing [87]. The address resolution process is iterative and involves contacting several name servers, each one responsible (i.e., *authoritative*) for a part of the domain name. In Fig. 4.5 we show an example. The process is triggered by an Internet user which, with the aim of establishing an HTTP session, queries a DNS resolver to obtain the IP address of `www.corriere.it`. The DNS resolver sequentially contact three name servers: the first (`a.root-servers.net`) replies indicating `a.dns.it` as authoritative name server for `.it` domain names; the second (`a.dns.it`) replies indicating that `ns.ita.tip.net` is authoritative for domain names ending with `corriere.it`; the third (`ns.ita.tip.net`), authoritative for `www.corriere.it` replies with the address of the host. Once the DNS resolver has

received the IP address from the last authoritative server, it send it to the user which eventually establishes an HTTP session with `www.corriere.it`.

DNS Records Caching

Each record has a Time To Live (TTL) [92], that can range from 0 (i.e., no cache) to days or weeks. It determines for how long the given response record can be kept in cache. The consequence of the DNS caching architecture is that DNS record updates do not propagate immediately in the network until cached records expire. Record caching is a pretty complex mechanism [92] as all DNS records used in the resolution process do not necessarily have uniform TTL values. Supposing that a DNS resolver starts with an empty cache the resolution of `www.corriere.it`, its cache at the end of the iterative process will be populated with several records – each one with its own TTL. For example, the A record of `www.corriere.it` has TTL equal to 600 seconds, shorter than the NS record of `corriere.it` (10,800 seconds) and shorter than the NS record of `.it` (172,800 seconds). So if `www.corriere.it` is requested after 700 seconds, the DNS resolver will no longer have its IP in cache as the A record expired in the meantime. However, it will still have in cache the NS records for `corriere.it` and `.it`. The resolver will contact again ".it" DNS servers only after the NS record for `corriere.it` has expired[3].

To make caching even more complex to understand, differences in DNS implementations must also be taken into account. In the above example, the NS record for `corriere.it` has a TTL of 10,800 seconds as it has been set by all the ".it" DNS servers, but as its TTL reported by the authoritative DNS of `corriere.it` is 600 seconds (i.e., `dig -t NS corriere.it ns.ita.tip.net`) some DNS implementations might override 10,800 with 600, making harder to predict the resolver cache contents.

Data caching must be taken into account when monitoring DNS traffic. Indeed, supposing that two domains are equally contacted during the day by a given resolver, the name servers for the domain with lower TTL will receive more queries than the

[3] It is worth to remark a name server can have different TTL values for NS records of domains it is authoritative for. Thus even within a single domain, TTLs might not be necessarily uniform.

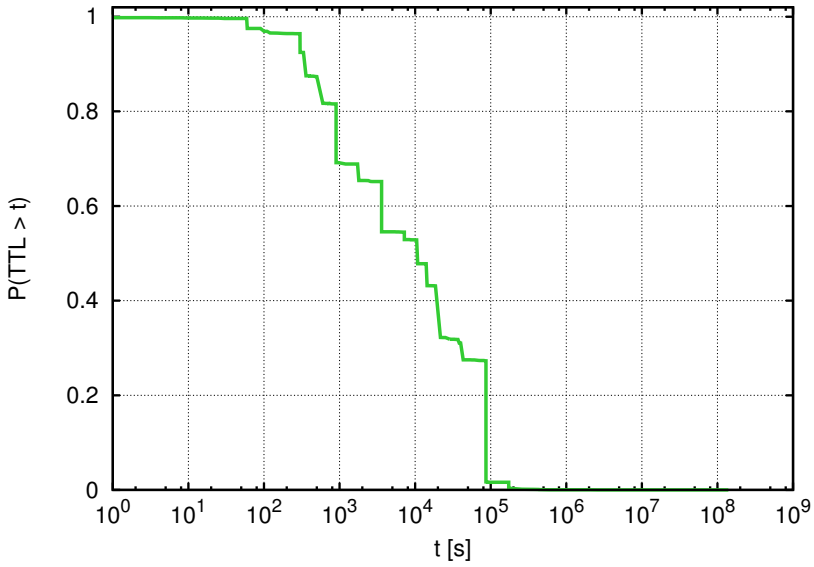


Fig. 4.6. TTL CCDF for `.it` Domains

name servers of the other domain, even though both domains have been contacted the same number of times by end-clients. Figure 4.6 shows the TTL complementary cumulative distribution function (CCDF) of `.it` domains. As shown in figure, it turns out that over 98% of `.it` domains have a TTL less than 86400 seconds, i.e. 1 day.

Monitoring DNS Traffic

As anticipated above, we are monitoring the DNS traffic at ".it" ccTLD DNS servers. This means that we only observe queries for `.it` (i.e., we do not observe queries for `.org` or `.net`) and only for those domains for which ".it" DNS servers are authoritative. Out of all the DNS traffic we observe, in the methodology described later in this thesis, we take into account only the AAAA, A, and MX records. In addition, for A and AAAA records, we ignore queries for both hosts that are known to be DNS servers and for which ".it" DNS servers are not authoritative (e.g. A record of `www.sub-domain.domain.it`). The reasons why we discard these queries are manyfold:

- Records other than A, AAAA, MX are used by the DNS infrastructure to resolve addresses (e.g. the NS record) or as ancillary records (e.g. PX records).

- A and AAAA record queries (as well NS records) for hosts that are DNS servers should not be taken into account because:
  - They have been requested due to the DNS caching mechanism where A records of DNS servers expire at different times than the corresponding NS record.
  - DNS servers are usually authoritative for many domains, and thus whenever we observe a A/AAAA record query for a DNS server, it is not possible to associate it with the domain name for which it has been requested.

In conclusion, we do not need to account all DNS queries but only those that indicate user activity such as an MX record query that indicate that an email will be sent, or A and AAAA records of hosts other than DNS servers (e.g. `www.nic.it`) that instead are used by the DNS system to resolve addresses. Please note that in theory, for a given resolver, once a record has been put in cache, no further query for the same record will be issued before the record expires as specified by the TTL. In practice this does not always hold, as most resolvers select authoritative name servers based on their response time. This explains why we often observe some extra queries used by resolvers to estimate the response time of all authoritative DNS servers for a given Internet domain. Thus beside these probing queries, we can identify resolvers that do not obey to the DNS specification when they perform queries that are well above the limit set by the TTL for the specified record. It is worth to remark, that resolvers identified using this method, cannot always be considered as malicious hosts. This is because sometimes network administrators periodically flush resolvers caches in order to reduce memory usage and thus extra queries are observed. For this reason we mark resolvers as malicious only whenever they significantly exceed the number of queries specified by the TTL.

4.2.3 DNS Modeling Methodologies

4.2.3.1 Normalizing Non-Uniform TTL Values

In order to model DNS traffic, we need to take into account the TTL and not just count DNS queries. In fact, if domain A has a TTL greater than domain B, a resolver that has to resolve both A and B addresses continuously throughout the day, will

issue fewer queries for A than B, as A records have a longer cache lifetime than B records. Since we need to deal with all the ".it" domains, our methodologies should enable domains with heterogeneous TTL values to be compared. This means that TTLs have to be normalized to the maximum TTL value among all the observed TTL values for NS records. As we measured that less than 2% of – about 2.5 million at the date of writing – ".it" domains use a TTL greater than 86,400 sec (1 day), we decided to use 1 day as baseline for our graph theoretical DNS models.

4.2.3.2 Bipartite Network Models of the DNS

A very effective way of modeling resolvers, domains and their interactions is through an *undirected bipartite graph*[4] $G = (V, E)$, such that $V = R \cup D$ and $R \cap D = \emptyset$. We take as R the set of resolvers and as D the set of domains. We shall recall that the bipartite graph G can be represented with an adjacency matrix of the form

$$\mathbf{A} = \begin{pmatrix} \mathbf{0} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{0} \end{pmatrix},$$

where $\mathbf{B} = [b_{r,d}]$ is a matrix with $|R|$ rows and $|D|$ columns, that uniquely identifies G.

In general, a bipartite graph of the DNS can be built by placing edges between a resolver $r \in R$ and a domain $d \in D$ whenever a certain condition is met. In this thesis we generate the following bipartite graphs:

- G_{ALL}: we place an edge between r and d iff r issued at least one DNS query for d in the observation period for A, AAA and MX records.
- G_{WEB}: we place an edge between r and d iff r issued at least one DNS query for d in the observation period for A and AAAA records and specify a name which is: the domain name with no host specified (e.g. `nic.it`); or the domain name preceded by either `www` or `web` (e.g. `www.nic.it`). In essence, we consider only those DNS queries that should be originated uniquely by web traffic[5].

[4] The reader may refer to Chapter 2 for definitions and terminology.

[5] We are aware that using this approach, some web traffic might not be accounted. This happens whenever a resolver 1) issues queries for hosts that are not marked as web although

- G_{MX}: we place an edge between r and d iff r issued at least one DNS query for d in the observation period for MX records. In essence we consider only `.it` Internet domains email traffic[6].

In all the previous cases, once the condition is met, the degree of a resolver $r \in R$ equals the number of domains it issues queries for. Similarly, the degree of a domain $d \in D$ represents the number of resolvers that query a given domain name. Since we excluded isolated nodes, i.e. resolvers and domains with zero degree, we have that degrees are always not less than one.

4.2.3.3 Common-Neighbors Network Models of the DNS

Internet domains and DNS resolvers can also be modelled in a way that the concept of adjacency becomes associated to the number of their common neighbors. In the case of two domains d_1 and d_2, *common neighbors* represent DNS resolvers which issue queries for both d_1 and d_2. This value is obtained by multiplying element-by-element columns of $\mathbf{B}$ with indices d_1 and d_2 and summing the result, i.e. $\sum_{r=1}^{|R|} b_{r,d_1} b_{r,d_2}$. Equivalently, the number of *common resolvers* can be directly computed for any pair of domains by taking the matrix product $\mathbf{B}^T\mathbf{B}$. Assuming the latter product is an adjacency matrix of a weighted graph $G_D = (D, E_D)$, then it turns out that G_D has D as its set of nodes. In addition, any of its edges $(d_1, d_2) \in E_D$ is associated with a positive weight corresponding to the number of resolvers shared by d_1 and d_2. Similarly, the number of *common domains* for each pair of resolvers can be obtained by computing $\mathbf{B}\mathbf{B}^T$, which in turn can be taken as the adjacency matrix of a weighted graph $G_R = (R, E_R)$ having R as its set of nodes. Each edge $(r_1, r_2) \in E_R$ is associated with a positive weight corresponding to the number of domains shared by r_1 and r_2.

---

they are in practice a web site (e.g. `video.mysite.it`) or 2) has issued a query for a record other than `www` (e.g. `mx.nic.it`) prior to issue a query for `www`. In this case, since the resolver cache was already filled-up, the query for `www` could not be observed. We estimate these are few cases as the probability that a web user makes non-web activity with a domain prior to access the web site is small.

[6] It is worth to remark, that in case a domain name has no MX record defined, email senders query the A record of the domain name. This means that queries for the exact domain name can either be due to emails or web traffic. In general as most domains have the MX record defined, we account queries for exact domain name into web traffic.

4.2.4 DNS Ranking Methodologies

As we have introduced DNS traffic models, we can now focus on how to suitably assess the relevance of resolvers and domains. In order to do that, we need to assign domains/resolvers a score that can will be used as baseline for their ranking. In particular, we define two rankings by sorting resolvers and domains in a decreasing order of *node degree* and *eigenvector centrality*. We refer the reader to Chapter 2 for a thorough discussion of these network features.

Node Degree

The *degree* of a node i is the number of nodes adjacent with i. Hence, in the case of a domain $d \in D$ in any of the bipartite graphs described above, its degree counts the number of resolvers issuing queries for it. Similarly, the degree of a resolver $r \in R$ gives the number of `.it` domains it issues queries for.

Eigenvector Centrality

We choose the relevance of a domain in a way that it is directly proportional to the sum of the relevance of resolvers issuing queries for it. Similarly, the relevance of a resolver is chosen to be directly proportional to the sum of the relevance of domains it issues queries for. Formally, the relevance x_i of resolver (domain) i is measured as $x_i = \lambda^{-1} \sum_{j=1}^{n} a_{i,j} x_j$. This measure can be written in matrix form as the eigenvector equation $\mathbf{Ax} = \lambda \mathbf{x}$ [23]. In general, there are many eigenvalues for which an eigenvector exists. However, with the additional requirement that components x_i of $\mathbf{x}$ be non-negative, then the Perron-Frobenius theorem ensures that λ is the largest (in absolute value) eigenvalue and $\mathbf{x}$ the corresponding eigenvector. As future work we plan to evaluate additional methods of graph theory for defining new ranks, such as strength, coreness, closeness, and betweeness.

4.2.5 Results and Validation

The ".it" zone has seven administrative DNS servers, three of which with anycast addresses. The ".it" DNS monitoring system [46] we used for validating this work monitors four authoritative name servers. Two name servers have anycast

Property	G_{ALL}	G_{WEB}	G_{MX}
no. of edges	33,047,646	17,896,183	2,099,030
no. of domains	1,779,249	1,693,180	323,956
no. of resolvers	511,039	294,091	99,016
avg. d. degree	18.75	10.57	6.48
avg. r. degree	64.67	60.86	21.20

Table 4.2. Structural properties of bipartite DNS networks

Structural properties of bipartite DNS graphs

address (`a.dns.it` located in Rome and Milan) and two have unicast address (`dns.nic.it` and `nameserver.nic.it`, both located in Pisa). Every ".it" DNS server node serves about 40 million requests/day, and we passively collect DNS traffic using a home-grown open-source NetFlow probe [61] featuring a plugin for dissecting DNS query/responses. This solution allows us to be independent from the DNS implementation being used, and thus be general enough to use it on different contexts. The system is producing daily domain ranking since Jan 2013. In this section we present the monitoring results we observed on Jan 4th, 2013 while monitoring `dns.nic.it`. Every night we consolidate the DNS traffic logs produced by the probe, and produce:

- The G_{ALL}, G_{WEB}, G_{MX} graphs.
- The domain and resolver rankings based on score we described on this thesis.
- The consolidated list of suspicious DNS activities carried on by resolvers that we use to spot potential issues.

Structural Properties of DNS Graphs

Structural properties of bipartite graphs G_{ALL}, G_{WEB}, G_{MX} are reported in Tab. 4.2. For each graph we indicate: the number of edges $|E|$; the number of domains $|D|$ and resolvers $|R|$ with degree greater than zero; the average domain degree $\overline{d_D} = |D|^{-1} \sum_{d \in D} d_d$; and the average resolver degree $\overline{d_R} = |R|^{-1} \sum_{r \in R} d_r$.

We observe that G_{WEB} has half the edges of G_{ALL}. In addition, while the number of domains is slightly smaller, resolvers reduce significantly (from more than 500,000

to less than 300,000), highlighting that a large part of them is not interested in resolving names for web resources – although web traffic may be under-estimated as previously described in Sect. 4.2.3.2. On average, resolvers query the same number of domains in G_{ALL} and G_{WEB} as highlighted by values of average resolver degree which are pretty close in both cases. Each domain is resolved for its web resources only one half of the times as the average domain degree halves from G_{ALL} to G_{WEB}. Much more significant reductions are observed for structural properties of graph G_{MX}, suggesting that only the 20% of resolvers issue queries uniquely for mail. Although this fact deserves further investigation, it may indicate that one resolver out of five has mail servers among its users.

We now deepen the analysis by further studying node degrees. In Fig. 4.7 we show Log-Log plots of frequency f_d versus degree d for domains and resolvers. *Frequency* f_d is the number of domains (resolvers) having degree d. These plots span approximately 5 orders of magnitude, indicating great variability and skewness in the degree of domains and resolvers. We observe that a single resolver can issue queries for more that 100k different domains, while the maximum number of resolvers querying a domain never exceed 50k. Nevertheless, these are statistically rare events. Indeed, we observe that domains with a low number of queries are the most frequent. Similarly, resolvers querying a low number of domains are the most common. Hence, the *"interest"* shown for Internet domains from DNS resolvers is far from being arbitrary. This "interest" can be quantified to some extent by observing that, for some ranges of d, plots are approximately linear in the Log-Log plots of Fig. 4.7. Linearity in the log-log scale mean that node degree follows a *power law*. Mathematically, a function $f(x)$ follows power law if it varies proportionally to a power γ of x, i.e., $f(x) \propto x^{\gamma}$. In our case, power law means that domain (resolver) i has a chance of having degree d_i which is proportional to the degree raised to a constant power γ. We estimated γ values using the method described in [39], which also provides values d_{min} such that power laws are obeyed only for $d \geq d_{min}$. Power laws fits with best γ values are plotted in Fig. 4.7 as dark lines, starting from d_{min} for each degree frequency. Tails of domain degree frequency in both G_{ALL} and G_{WEB} follow very well power laws with strikingly close exponents. In G_{MX} the power law is obeyed for a much lower d_{min}. Similarly, power laws are observed for resolver degree fre-

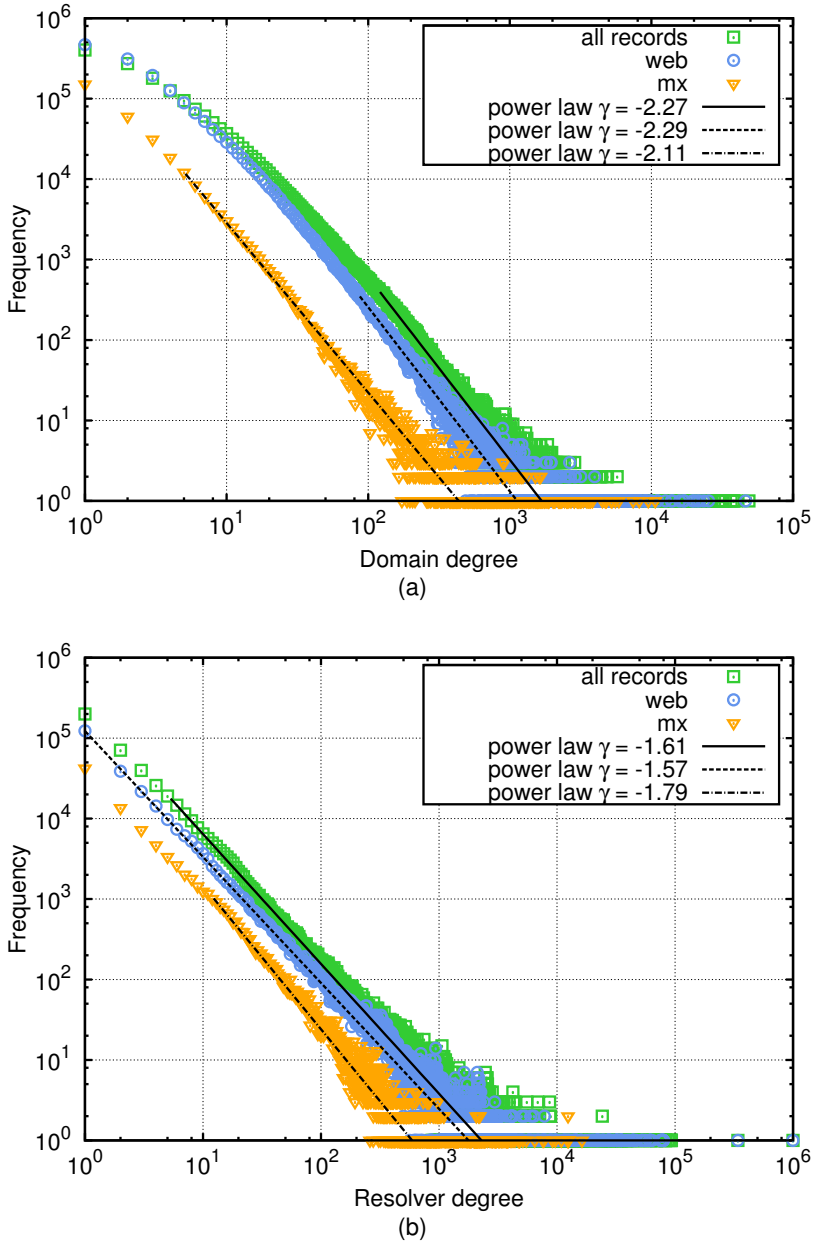


Fig. 4.7. Log-Log plots of frequency f_d versus degree d.

quencies but with lower exponents, indicating slower decays. Since power laws with exponent less that -2 have infinite mean and variance [48], we stress on the extreme skewness of resolver degree. Nevertheless, also resolvers in G_{ALL} and G_{MX}

Property	G_D	G_R
no. of edges	124,717	124,750
no. of nodes	500	500
avg. edge weight	1253.72	1972.97
avg. node intensity	625,442.89	984,516.46

Table 4.3. Structural properties of Common-Neighbors DNS graphs

degree obey power laws with very close exponents. We also compared our power law estimations with other distributions through *likelihood ratio tests* [39]. Such tests suggested that frequencies may also be well approximated by stretched exponentials and truncated power laws, but they definitely excluded gamma, log-normal and exponential distributions. Further investigations are left as future work. Nevertheless, we can conclude that a few domains (resolvers) are responsible for most of the DNS activity and power laws well describe this activity.

We have also analysed the common-neighbors DNS graphs. We generated weighted graphs G_D and G_R by selecting the top 500 highest-degree domains (resolvers) and all their adjacent resolvers (domains) from G_{ALL}. In both cases, the weight of each edge equals the number of neighbors in common. Structural properties of common-neighbors graphs G_D and G_R are reported in Tab. 4.3. Both graphs have a number of edges which approaches the maximum, hence any pair of domains (resolvers) have at least one neighbor in common. By observing the average edge weight, it turns out that pairs of domains on average are requested by more than 1200 common resolvers. Similarly, pairs of resolvers on average query approximately 2000 domains. In addition, average node intensities (i.e., the average of the sum of edge weights for each node) tell that a large number of resolvers (domains) is shared among top 500 highest-degree domains (resolvers).

In Fig. 4.8 we show the Complementary Cumulative Distribution Function (CCDF) $p_{E_D}(h) = P(w_{d_1,d_2} > h|(d_1, d_2) \in E_D)$ giving the probability that any pair of domains $(d_1, d_2) \in E_D$ has a number w_{d_1,d_2} of resolvers in common greater that h as function of h. Similarly, we show the CCDF $p_{E_R}(h) = P(w_{r_1,r_2} > h|(r_1, r_2) \in E_R)$ which gives the chance that any pair of resolvers $(r_1, r_2) \in E_R$ shares a number of domains

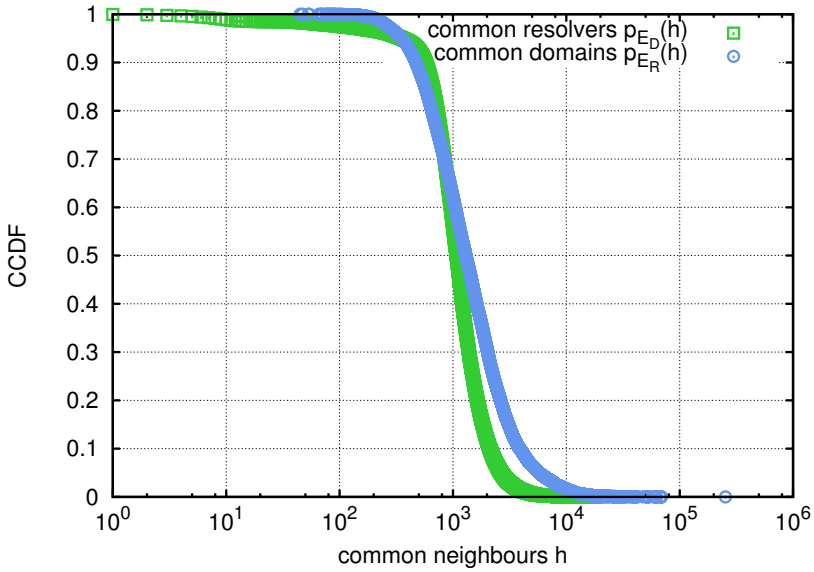


Fig. 4.8. G_D and G_R common neighbors CCDF $p_{E_D}(h)$ for domains and $p_{E_R}(h)$ for resolvers.

Domain Name	`dns.nic.it`	`a.nic.it`	**Difference**
amazon.it	1	1	=
telecomitalia.it	2	4	+2
virgilio.it	3	3	=
fastwebnet.it	4	2	-2
corriere.it	5	5	=
tiscali.it	6	6	=
aruba.it	7	5	-2
google.it	8	11	+3
vodafone.it	9	12	+3
gazzetta.it	10	8	-2

Table 4.4. ".it" domains ranking comparisons

Domain Score Comparison

w_{r_1,r_2} greater that h. Both distributions are almost identical up to 1000, whereas the resolvers start decaying much faster than domains.

We have also compared the degree of top 100 domains on two monitored name servers (`dns.nic.it` and `m.dns.it`), assigning a score from 1 to 100 based on the domain degree. We have found that over 93% of domains are present on both servers, and that the domain scores are pretty close but not alike as shown in Tab. 4.4. Resolvers score have instead a different behaviour:

- Resolver score are very different across the two name servers servers as only 52% of top 100 resolvers contacted both servers.
- For each name server, the list of top resolvers across various days reports only minor differences.

As the Round Trip Time (RTT) is the metric used to choose between authoritative servers for the same zone[3] [172], not all name servers are alike for resolvers as they prefer those that reply with lower RTT[7]. This is also justified by a probing activity that we have identified in our traffic traces, where several resolvers issue the same query to all monitored ".it" authoritative name servers twice in about 10 seconds, in order to figure out which one reports the lowest RTT; this will be the selected name server to be used for future queries on that domain name.

Domain rankings are discussed in greater detail in the following.

Internet Domains Rankings

In this section we present domain ranking results obtained from an observation of Jan 4th, 2013 while monitoring `dns.nic.it`. The monitoring system is in place since late 2011 thus we have access to historical data series. We omit the results we have obtained on the other three monitoring sites as they are pretty similar to what we will present later on this section. The only differences we observe is that resolvers select an authoritative name server based on its RTT. Hence, for each monitored site the resolvers distribution is different in terms of queries but not in terms of edges, confirming that resolvers randomly select authoritative servers and that they probe

[7] BGP (Border Gateway Protocol) peering allows RTT to be reduced for those resolvers belonging to ASs (Autonomous Systems) for which there is a peering relations, thus affecting the resolvers distribution across monitored name servers. This is because depending on the site where the name server is located, there are non-uniform peering relationships in place.

servers for selecting the one with lower RTT. To the best of our knowledge we have not found in literature works similar to the one we presented nor we consider sites such as `alexa.com` which have poor information for `.it` domains; thus we are unable to compare our results with others. For goals listed in Sect. 4.2.1, we use the following approaches.

Rank	Degree	Eig. Cent.	Degree	Eig. Cent.
1.	amazon	corriere	amazon	gazzetta
2.	fastwebnet	rcs	google	corriere
3.	virgilio	gazzetta	corriere	gazzettaobjects
4.	telecomitalia	aruba	excite	corrieredellosport
5.	corriere	virgilio	imdb	softonic
6.	aruba	excite	softonic	tripadvisor
7.	tiscali	gazzettaobjects	gazzetta	vogue
8.	gazzetta	softonic	tripadvisor	agi
9.	rcs	corriereobjects	virgilio	tuttosullavoro
10.	rcsadv	groupon	ebay	virginradioitaly

Table 4.5. Bipartite Graphs: top ".it" degree and eigenvector centrality domain ranking

Domain and Resolver Ranking

We rank domains and resolvers according to their node degree and eigenvector centrality. Node degree ranks domains in terms of their degree without considering neighboring resolvers degree. Eigenvector centrality instead takes into account both domains and neighboring resolvers degree. In Tab. 4.5 we compare the results for top `.it` domains when considering all or only web traffic as defined in Sect. 4.2.3. Both rankings are similar but not alike. When considering the domain degree we count just the number of resolvers that contacted the domain, without distinguishing across resolvers degree — i.e. a resolver that queried a limited number of domains has the same weight of a resolver that queried many more domains in the same observation period. When using the eigenvector centrality, domains queried by resolvers with higher scores are pushed higher in the ranking. We believe that both ranking criteria are good, but the eigenvector centrality is probably the best as it takes into account

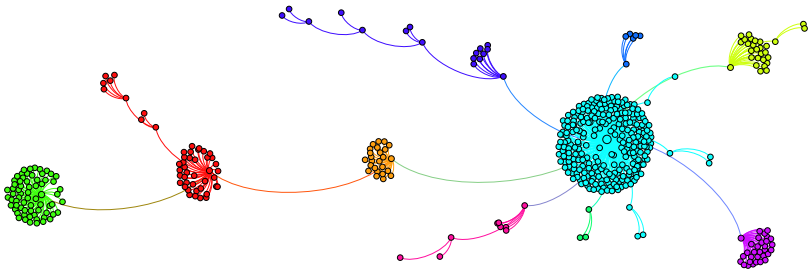


Fig. 4.9. Common-Neighbors Maximum Spanning Tree

neighboring resolvers degree that give an indication of the size of the population behind such resolver. This is in the assumption that resolvers with higher degree are likely to serve a larger client population than those with a smaller degree.

Inter-domain Relationships

We build the common-neighbors graph of `.it` Internet domains. Specifically, we have selected the top 500 domains according to their eigenvector centrality score from G_{ALL} and created a their common-neighbors graph as described in Sect. 4.2.3. Then, we extracted the maximum spanning tree [157] from the latter graph. The maximum spanning tree is shown in Fig. 4.9. If domains d_1 and d_2 are connected in the minimum spanning tree, there is no domain d_3 such that: *i)* d_1 has a number of common neighbors with d_3 greater than the number it has with d_2; and *ii)* d_2 has a number of common neighbors with d_3 greater than the number it has with d_1. Formally, $cn(d_1, d_2) > cn(d_1, d_3) + cn(d_2, d_3)$, where $cn(d_i, d_j)$ is the number of neighbors between i and j.

In Fig. 4.10 we zoomed a region of Fig. 4.9 to spot the links of a large Italian content provider. Although our approach is based uniquely on the domains degree with no knowledge of the nature of information hosted by domains web sites, our algorithm has been able to place on the same cluster additional domains of domestic ISPs and Internet content providers. The same behaviour can be found on the

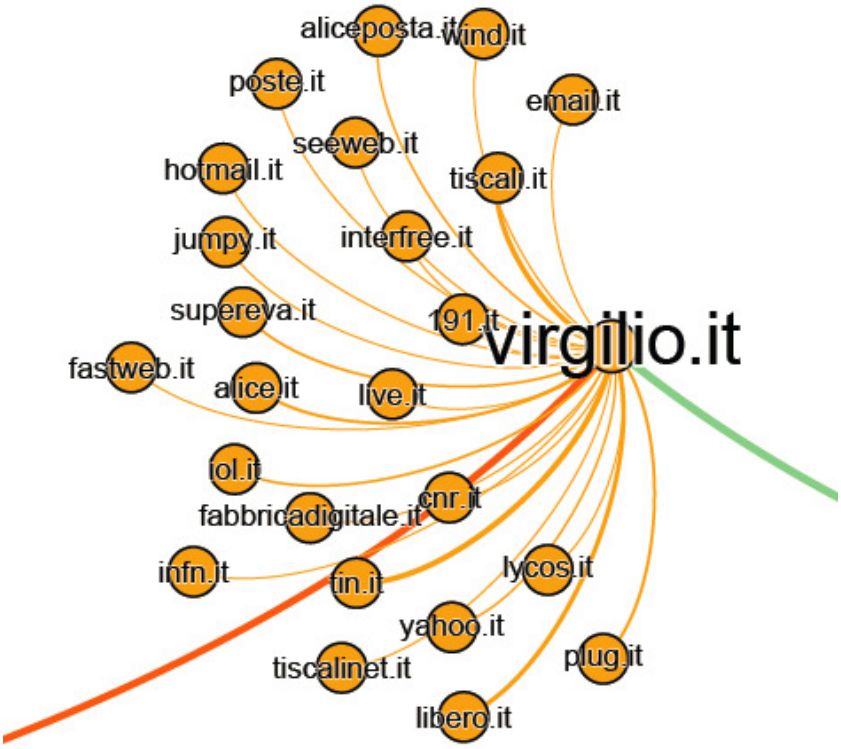


Fig. 4.10. Common-Neighbors Maximum Spanning Tree around a Large Italian Content Provider (`virgilio.it`)

additional clusters of the Fig. 4.9. According to our knowledge, domain clustering happens when:

- Domains have economical relationships. For instance domains such as `fiat.it`, `alfaromeo.it` and `abarth.it` belong to the same cluster as their web sites contain cross links to all these domains that belong to the Fiat group.
- Domains are similar in content and nature as shown in Fig. 4.10.
- Domains have some “side relationships”. For instance `amazon.it` has several edges in common with peer-to-peer and torrent tracker sites. This is because such sites use Amazon to show multimedia artwork of shared files, or perhaps people first search on Amazon the stuff they are interested in, and the access such sites for (illegally) downloading it.

Misbehaving Resolvers

We use a combined approach.

- The modeling methodologes take TTL into account. For each tuple `<resolver, queried domain, TTL query response>` we should not observe at each monitoring point a query more frequent than the TTL specified. If this property is not respected, then the resolver is likely to use a faulty software or to be a scanner. In both cases it is worth to be analysed more in detail (see Sect. 4.2.6).
- For each resolver we keep the ratio of positive versus negative replies, and we group it according to the autonomous system (AS) such resolver belongs to. This is in order to also take into account other resolvers (e.g. secondary DNSs) belonging to the same administrative domain. If the ratio exceeds a certain threshold we mark this activity as suspicious. In fact, in case of negative DNS replies (e.g. NXDOMAIN), the resolver must also cache these responses and avoid repeating negative queries similar to what happens with positive replies. Furthermore high negative responses ratio, often identify scanners attempting to guess the registered domain names, given that such list is not publicly available.

Rank Domains According to Traffic Type

Tab. 4.5 shows different types of ranking based on the nature of traffic. As previously explained, data caching in DNS prevent us from analysing data at a granularity lower than a day, and thus just compute a daily ranking. Nevertheless, this does not prevent us from periodically accounting the number of observed domain queries. This has enabled us to:

- Highlight periodicity in traffic, such as identify domains that are mostly accessed during the day including webmail portals and (many but not all) news sites written in Italian.
- Spot hosts used for activities that happen during the whole day such as torrent tracker sites.
- Highlight hosts that receive very periodic contacts from specific resolvers, that might be due to remote monitoring activities.

Identify Email Spam and Attacks

We use the G_{MX} bipartite graph described in Sect. 4.2.3 for focusing on email traffic. Currently, we are able to use this information for emitting alerts only if the daily degree of domains changes suddenly with respect to previous days. As future work we plan to characterize domains, and thus create a more advanced alerting system. For instance domains of ISPs or large institutions can have a higher alert threshold than domains of smaller institutions. The ratio between G_{MX} and G_{ALL} can also be used to spot sites that mostly perform mail activity, and also that might be worth to further investigation.

In summary, the DNS traffic model we have defined has enabled us to reach our project goals. Currently, we are currently evaluating additional methods for assigning scores to domains and resolvers, in order to create additional rankings.

4.2.6 Future Work Items

The described contributions are a good starting point for several follow up activities, including:

- Relationships highlighting in web traffic via common-neighbors graphs models of DNS traffic. For instance the number of resolvers in common between two domains can be used to:
  - Evaluate the effectiveness of advertisements (e.g. restricted to web traffic, counting the number of resolvers in common between `X.it` and `companyname.it` allows to figure out on which web sites `companyname.it` places its advertisements).
  - Understand the chain of interests of people (e.g. Internet users who access news site `X` will likely also access news site `Y`).
- RTT minimization. In order to have an efficient DNS infrastructure, the goal is to minimize the RTT for resolvers. Using our models, we can start optimizing traffic routing so top scoring resolvers can be reached by `.it` name servers with a low TTL thus resulting on a more efficient service.
- Misbehaving resolver identification. Refinement of the algorithm used to identify resolvers that do not obey to the TTL, so that we can distinguish between misbe-

having or misconfigured resolvers worth to track as they may conduct malicious activities [190], and resolvers running malfunctioning DNS implementations that instead should not be taken into account.

4.2.7 Conclusion

Original contributions presented in this section are novel methodologies for modeling DNS traffic by means of graphs. During the validation on the ".it" ccTLD authoritative name servers, we found that DNS resolvers degrees are highly-skewed and obey power laws. This fact also holds for Internet domains when considering all traffic or just a portion of it such as web or email. These findings give new insight into the scale-free nature of the DNS system, where a few resolvers and a few domains are responsible for most of the DNS activity.

Other original contributions are novel methodologies for ranking Internet domains using DNS traffic. The main advantage of our approaches is that the monitored traffic to be used for creating rankings is very limited with respect to other protocols such as HTTP or social networks analysis. The validation phase has demonstrated that using the proposed methodologies enable to: successfully rank resolvers and Internet domains according to different criteria; automatically cluster domains containing similar information and interests; and discover malicious activities using the DNS traffic otherwise difficult to identify by other means.

5

Conclusions and Future Directions

Recently, networks have become fundamental tools to study real-world systems. Their unique ability to model complex relationships has enabled knowledge extraction from modern systems such as financial markets [25] [141], technological infrastructures [179] [21] and societies, to name a few key examples. Efficient network algorithms are necessary to extract new knowledge, especially from large-scale systems that are available today, in the era of *big-data*. Among them, community detection algorithms are largely used and widely acknowledged as being essential to gain new insights into the structure and function of systems [59, 108, 156, 158, 164, 166, 20, 110, 112]. Community detection algorithms are the first main area of focus of this thesis. The second area is the application of networks to the study of the Internet, one of the most prominent modern technological systems. Being able to understand, develop and protect it is of fundamental importance and has many direct worldwide implications [105, 104, 179, 21, 105, 159, 150].

Our findings and original contributions are presented below. For the sake of clarity, network research areas of focus are discussed separately.

5.1 Parallel Network Community Detection Algorithms

As eluded to above, the fist area of focus of this thesis has been the development of network algorithms, allowing researchers to study the underlying clustered structure of networks. We have focused on k-clique community detection algorithms, since cliques are responsible for the structural properties of networks [74]. Existing algo-

rithms for k-clique community detection are extremely demanding in terms of computational resources. However, to the best of our knowledge, no formal worst-case analyses have been conducted, leaving the existence of bottlenecks and scalability issues unexplored. In this thesis we carry out a theoretical worst-case complexity analysis of existing k-clique community detection algorithms, highlighting their scalability issues. We analytically demonstrate that worst-case space and time complexities have a quadratic dependence on the number of maximal cliques in the network. Then, we develop algorithms and parallel algorithms that dramatically reduce the aforementioned bounds on complexity. Specifically, we:

- Reduce worst-case space complexity from a quadratic to a linear dependence on the number of maximal cliques in the network.
- Reduce worst-case time complexity that becomes inversely proportional to the number of processing units available for the computation.

In order to lessen space complexity, we design an innovative algorithm to obtain the connected components of networks. This algorithm enables large-scale networks to be decomposed into an arbitrary number of smaller subnetworks, without any constraint on their structure. The connected components of the initial, non-decomposed, network are obtained using only the connected components of the subnetworks. The correctness of the algorithm is demonstrated in a theorem. Time complexity is alleviated by opportunely balancing the workload on the processing units available. Operations are finely distributed among the units available before starting the computation, thus avoiding runtime load balance overheads.

Extensive experiments have been carried out to study the algorithms proposed, to validate them against theoretical worst-case bounds, and to compare them with other community detection methods. Experimental results have demonstrated that:

- Existing k-clique community detection algorithms fail to mine communities from networks larger than a handful of nodes.
- Contributed algorithms have enabled k-clique communities to be detected from networks of a scale never reached before, even on commodity hardware.
- Proposed algorithms are typically at least one order of magnitude faster than other state-of-the-art k-clique community detection methods.

- The amount of resources demanded by our algorithms is dramatically reduced. For example, we have measured a runtime memory footprint reduction of more than 30 GB on the same network.
- Algorithms performances are almost ideal on parallel architectures. Indeed, we have measured a linear speedup, i.e., algorithms halve their execution time when the number of processing units is doubled.

5.2 Network-Based Methodologies to Study the Internet

The second main area of this work has been on the application of networks to the study of the Internet, one of the most prominent modern technological systems. We have studied the Internet from two alternative perspectives, namely from:

- The level of abstraction of Autonomous Systems (ASes), to shed light into the hidden factors underlying complex, business-driven interconnections that enable the Internet to be a worldwide pervasive infrastructure. Stock market data has been used in this study as well, to leverage on the twofold nature of Internet operators that, on the one hand, are physical collections of electronic devices and, on the other hand, are large revenue-generating companies.
- The Domain Name System (DNS), to identify malfunctions and malicious activities, as well as to rank Internet domains and uncover patterns and trends in users' interests. New insights into the DNS have been gained by means of a large-scale analysis and validation, that includes data of the whole set of ".it" domains, of which, at the time of writing, there are more that 2.5 million.

Internet Companies, ASes, and Stock Markets

To study the Internet from this perspective, we have designed a novel general methodology to investigate cross correlations in evolving networks. Patterns and similarities in the dynamics of network nodes connectivity can be identified using the proposed methodology.

We have found that groups of Internet companies homogeneous with reference to their service offering – e.g. transit providers – are positively correlated in the stock

market. Similarly, we have found that even geographically close companies are positively correlated, suggesting the existence of common economic factors driving geographically homogeneous companies. We have also given empirical evidence on the existence of factors, common to all companies, which result in collective evolutionary behaviours in the Internet connectivity. Finally, we have also shown that factors governing stock price variations are not the same as those synchronously driving the variations of the Internet connectivity.

The proposed methodology to identify cross correlations is general enough to be applied to any kind of network. At the present time, we are using the presented methodology to study financial [16] as well as climate networks [73]. We are also extending our investigations to include the Random Matrix Theory [50]. We believe this will provide stronger support to the existence of collective behaviours in the evolution of networks. Similarly, it will provide a rigorous mathematical tool to filter out noise from genuine correlations.

The Domain Name System (DNS)

To study the Internet from this standpoint, we have designed and validated network-based models of DNS traffic. The DNS is one of the core protocols on which the Internet is built upon. Hidden behind higher-level protocols such as email and web, it carries valuable information that we have exploited, for example, to identify malicious activities and to understand trends and preferences of the Internet community.

We have investigated the structural properties of DNS traffic networks. We have rigorously verified that the DNS ecosystem is scale-free [12], with a handful of domains responsible for a large part of user activities. We have also obtained meaningful rankings of Internet domains, using very limited amounts of traffic. With our approaches, we have been able to automatically cluster domains containing similar information and interests. We have given empirical evidence that the sole information contained in the DNS traffic, if opportunely modeled into networks, is enough to cluster together similar sites – e.g., content providers, banking and finance, and automotive. Similarly, we have discovered malicious activities, which otherwise would have been difficult to identify.

As future works, we are planning to use the DNS traffic to evaluate the effectiveness of advertisements. This can be accomplished by monitoring the DNS queries that originate from a given source. Similarly, we are planning to refine the identification of malicious activities that may lead to failures in the DNS.

A

Proofs of Our Theorems

A.1 Proof of Correctness of the Connected Components Merging Algorithm CONNECT_ME

Theorem 1. *If F_1 and F_2 are collections of sets corresponding to the connected components of graphs $H_1 = (V_1, E_1)$ and $H_2 = (V_2, E_2)$, $V_2 \subseteq V_1$, then $CONNECT\_ME(F_1, F_2)$ ensures F_1 contains sets corresponding to the connected components of $H_1 \cup H_2$.*

Proof. Since $find_{F_2}(u)$ in line 3 returns the canonical element U of the set in F_2 containing u, a path between U and u always exists in H_2 because F_2 contains disjoint sets corresponding to the connected components of H_2.

Therefore, except for the trivial case when $U = u$, it must be ensured that U and u are in the same disjoint set in F_1 because the existence of the path between U and u in H_2 implies the existence of the same path in $H_2 \cup H_1$. This is guaranteed by $MERGE\_SETS$ in line 5 which finds in F_1 two sets containing U and u respectively, and joins them if they were not already the same. The only paths that can exist in H_2 and which are not generated in line 3, are those involving nodes p and q in the same connected component, for which $find_F(p) = find_F(q) = U$ with $U \neq p \neq q$. In any case, after line 5, U and p are in the same set when $u = p$ as well as U and q when $u = q$. Indeed, if the loop in line 2 executes first with $u = p$ and then with $u = q$, a set, already containing U and p, is joined with another set containing q. Similarly, if $u = q$ precedes $u = p$ in the loop, a set, already containing U and q, is joined with another set containing p. Hence, after the loop has executed with $u = p$ and $u = q$, both p

and q are in the same set. Therefore, for each couple of nodes p, q, if there exists a path between them in H_2, they end up in the same set in F_1.

If instead there exists a path between them in H_1, since they were in the same set of the initial collection and only union operations are performed by $MERGE\_SETS$, they continue to be in the same set after the loop has executed.

Finally, p and q are in the same set in F_1 also if there not exists a path between them neither in H_2 nor in H_1 but there exists in $H_1 \cup H_2$. Suppose, by contradiction, that they are in two different sets in F_1 after $CONNECT\_ME$ has executed. This implies they were in different sets at the beginning and hence in two different connected components in H_1. This implies also that all the paths of H_2 together are not enough to allow the creation of a new path between them, otherwise two disjoint sets containing p and q, respectively, would have been joined by $MERGE\_SETS$. Therefore, since the path does not exist in H_1 and all the all paths in H_2 do not permit the creation of a path between them, p and q, are in two separate connected components in $H_1 \cup H_2$.

A.2 A Parallel Algorithm to Detect k-Clique Communities

A.2.1 Proof of Worst-Case Time Complexity

Theorem 2. *If operations on collections of disjoint sets are in $O(1)$, perfect load balancing is achieved and overlap is calculated through binary searches, then $COSpoc(c_0, \cdots, c_{l-1})$ worst-case time complexity is in:*

$$O\left(\frac{l^2}{p} k_{max} \log_2 k_{max}\right). \tag{A.1}$$

Proof. Total worst-case time complexity is sum of the following:

- T_{in} required to initialize collections of disjoint sets.
- $T_{p,\mathrm{ov}}$ required to compute the overlap between pairs of maximal cliques.
- $T_{p,\mathrm{ds}}$ required to operate on collections of disjoint sets.
- T_{jo} required to merge partial results.

$T_{\text{in}} \in O(l \cdot k_{max})$ since each processor initializes in parallel $(k_{max} - 1)$ collections of disjoint sets with at most l elements.

$T_{p,\text{ov}} \in O((1/p) \cdot l^2 k_{max} \log_2 k_{max})$. The overlap is calculated through binary searches, which are $O(\log_2 k)$ on k elements. In the worst-case where all the maximal cliques are k_{max} in size, k_{max} binary searches are required for each of the $\binom{l}{2} \in O(l^2)$ possible pairs of maximal cliques. Therefore the total cost is in $O(l^2 k_{max} \log_2 k_{max})$. This total cost can be divided by p due to the perfect load balancing, obtaining the bound specified for $T_{p,\text{ov}}$.

$T_{p,\text{ds}} \in O((1/p) \cdot l^2 k_{max})$. In the worst-case where all the maximal cliques are k_{max} in size and have $(k_{max} - 1)$ nodes in common with each other maximal clique, MERGE_SETS is called $(k_{max} - 1)$ times for each of the possible $\binom{l}{2} \in O(l^2)$ pairs of maximal cliques. Therefore the total cost is in $O(l^2 k_{max})$ since operations on collections of disjoint sets are in $O(1)$.

$T_{\text{jo}} \in O(p \cdot l \cdot k_{max})$. CONNECT_ME processes at most l elements each time is called. Hence its cost is in $O(l)$, because operations on collections of disjoint sets are in $O(1)$. The bound specified for T_{jo} is obtained by observing that CONNECT_ME is called $(p - 1)$ times for each $k \in [2, k_{max}]$.

Only $T_{p,\text{ov}}$ and $T_{p,\text{ds}}$ have a quadratic dependence on l and their sum is in $O((1/p) \cdot l^2 k_{max}(\log_2 k_{max} + 1))$. The (A.1) is derived from the latter sum by disregarding the constant 1.

A.2.2 Proof of Worst-Case Space Complexity

Theorem 3. $COSpoc(c_0, \cdots, c_{l-1})$ *worst-case space complexity is in:*

$$O(p \cdot l \cdot k_{max}).$$

Proof. Each of the p processors has $(k_{max} - 1)$ collections $F_{q,2}, F_{q,3}, ..., F_{q,k_{max}}$ and each collection $F_{q,k}$ has at most l elements.

A.3 A Parallel Algorithm to Detect k-Clique Communities (on Steroids)

A.3.1 Proof of Worst-Case Time Complexity

Theorem 4. *With conditions as in Theorem 2, if the overall complexity of SLIDE, READ, WRITE, F_q initialization and CONNECT_ME is in* (A.1)*, then $COS(c_0, \cdots, c_{l-1})$ worst-case time complexity is in* (A.1)*.*

Proof. If SLIDE, READ, WRITE, F_q initialization and CONNECT_ME overall wost-case time complexity is in (A.1), the proof is mutatis mutandis the same as that of Theorem 2.

A.3.2 Proof of Worst-Case Space Complexity

Theorem 5. $COS(c_0, \cdots, c_{l-1})$ *worst-case space complexity is in:*

$$O\left(\, l \cdot (p + k_{max}) + W \,\right).$$

Proof. Each processor has a collection F_q with size l. Hence the total memory used by p processors is $p \cdot l$. As regards global collections $F_{\mathrm{global},k}$, their the total size reaches its maximum $l \cdot (k_{max} - 1)$ in a worst-case scenario where all the maximal cliques are k_{max} in size. Therefore, total space for collections of disjoint sets is in $O\left(\, l \cdot (p + k_{max}) \,\right)$. The $O(W)$ is for the sliding window.

References

[1] J. M. Agosta et al. “Mixture Models of Endhost Network Traffic”. In: *ArXiv e-prints* (2012).

[2] R. Albert, H. Jeong, and A.L. Barabási. “Error and attack tolerance of complex networks”. In: *Nature* 406.6794 (2000).

[3] Paul Albitz and Cricket Liu. *DNS and Bind, 4th Edition*. 4th ed. O’Reilly, 2001. ISBN: 0-596-00158-4.

[4] Nikolaos Alexiou et al. “Formal analysis of the kaminsky DNS cache-poisoning attack using probabilistic model checking”. In: *High-Assurance Systems Engineering (HASE), 2010 IEEE 12th International Symposium on*. IEEE. 2010, pp. 94–103.

[5] M. Allalouf, E. Kaplan, and Y. Shavitt. “On the feasibility of a large scale distributed testbed for measuring quality of path characteristics in the Internet”. In: *Testbeds and Research Infrastructures for the Development of Networks & Communities and Workshops, 2009. TridentCom 2009. 5th International Conference on*. 2009.

[6] Franklin Allen and Ana Babus. “Networks in finance”. In: *Wharton Financial Institutions Center Working Paper* (2008).

[7] J Ignacio Alvarez-Hamelin et al. “Large scale networks fingerprinting and visualization using the k-core decomposition”. In: *Advances in neural information processing systems*. 2005, pp. 41–50.

[8] Hussein A Alzoubi et al. “Anycast cdns revisited”. In: *Proceedings of the 17th international conference on World Wide Web*. ACM. 2008, pp. 277–286.

[9] M. Antonakakis et al. “Building a Dynamic Reputation System for DNS”. In: *9th Usenix Security Symposium* (2010).

[10] Hitesh Ballani and Paul Francis. “Mitigating DNS dos attacks”. In: *Proceedings of the 15th ACM conference on Computer and communications security*. ACM. 2008, pp. 189–198.

[11] Hitesh Ballani, Paul Francis, and Xinyang Zhang. “A study of prefix hijacking and interception in the Internet”. In: *ACM SIGCOMM Computer Communication Review*. Vol. 37. 4. ACM. 2007, pp. 265–276.

[12] Albert-László Barabási and Réka Albert. “Emergence of scaling in random networks”. In: *science* 286.5439 (1999), pp. 509–512.

[13] V. Barnett and T. Lewis. “Outliers in statistical data”. In: *Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics, Chichester: Wiley, 1984, 2nd ed.* 1 (1984).

[14] Vladimir Batagelj and Andrej Mrvar. *Pajek datasets*. `http://vlado.fmf.uni-lj.si/pub/networks/data/`.

[15] Vladimir Batagelj and Matjaz Zaversnik. “An O(m) Algorithm for Cores Decomposition of Networks”. In: *CoRR* cs.DS/0310049 (2003).

[16] Stefano Battiston et al. “The structure of financial networks”. In: *Network Science*. Springer, 2010, pp. 131–163.

[17] Andreas Berger and Eduard Natale. “Assessing the real-world dynamics of DNS”. In: *Proceedings of the 4th international conference on Traffic Monitoring and Analysis*. TMA’12. Berlin, Heidelberg: Springer-Verlag, 2012.

[18] Sun Bin, Wen Qiaoyan, and Liang Xiaoying. “A DNS based anti-phishing approach”. In: *Networks Security Wireless Communications and Trusted Computing (NSWCTC), 2010 Second International Conference on*. Vol. 2. IEEE. 2010, pp. 262–265.

[19] Vincent D Blondel et al. *Fast unfolding of communities in large networks*. URL: `https://sites.google.com/site/findcommunities/`.

[20] Vincent D Blondel et al. “Fast unfolding of communities in large networks”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2008.10 (2008).

[21] Marián Boguñá, Fragkiskos Papadopoulos, and Dmitri Krioukov. “Sustaining the internet with hyperbolic mapping”. In: *Nature Communications* 1 (2010), p. 62.

[22] Immanuel M. Bomze et al. “The Maximum Clique Problem”. In: *Handbook of Combinatorial Optimization*. Kluwer Academic Publishers, 1999, pp. 1–74.

[23] Phillip Bonacich. “Power and Centrality: A Family of Measures”. In: *American Journal of Sociology* 92.5 (1987).

[24] Phillip Bonacich. “Some unique properties of eigenvector centrality”. In: *Social Networks* 29.4 (2007), pp. 555–564.

[25] Giovanni Bonanno et al. “Networks of equities in financial markets”. In: *The European Physical Journal B-Condensed Matter and Complex Systems* 38.2 (2004), pp. 363–371.

[26] Giovanni Bonanno et al. “Topology of correlation-based minimal spanning trees in real and model markets”. In: *Physical Review E* 68.4 (2003).

[27] Stephen P Borgatti. “Centrality and network flow”. In: *Social networks* 27.1 (2005), pp. 55–71.

[28] S. Brin and L. Page. “The Anatomy of a Large-Scale Hypertextual Web Search Engine”. In: *Seventh International World-Wide Web Conference*. 1998.

[29] Coen Bron and Joep Kerbosch. “Algorithm 457: finding all cliques of an undirected graph”. In: *Commun. ACM* 16.9 (1973), pp. 575–577. ISSN: 0001-0782.

[30] N. Brownlee and K.C. Claffy. “Understanding Internet traffic streams: dragonflies and tortoises”. In: *Communications Magazine, IEEE* 40.10 (2002).

[31] Dongbo Bu et al. “Topological structure analysis of the protein-protein interaction network in budding yeast”. In: *Nucl. Acids Res.* 31.9 (2003).

[32] Charles W Calomiris. “The subprime turmoil: What’s old, what’s new, and what’s next”. In: *Maintaining Stability in a Changing Financial System* (2008), pp. 21–23.

[33] Emiliano Casalicchio and Igor Nai Favino. *The 1st Workshop on DNS Health and Security*. 2012. URL: `http://www.gcsec.org/sites/default/files/files/dnseasy2011.pdf`.

[34] Shu-Yan Chan, Pan Hui, and Kuang Xu. “Community detection of time-varying mobile social networks”. In: *Complex Sciences*. Springer, 2009, pp. 1154–1159.

[35] Nikolaos Chatzis et al. “On the importance of Internet eXchange Points for today’s Internet ecosystem”. In: *arXiv preprint arXiv:1307.5264* (2013).

[36] Chang-Sheng Chen, Shian-Shyong Tseng, and Chien-Liang Liu. “A unifying framework for intelligent DNS management”. In: *International Journal of Human-Computer Studies* 58.4 (2003).

[37] K. Chen et al. “Where the sidewalk ends: Extending the Internet AS graph using traceroutes from P2P users”. In: *Proceedings of the 5th international conference on Emerging networking experiments and technologies*. 2009.

[38] P Chen and S Redner. “Community Structure of the Physical Review Citation Network”. In: *Journal of Informetrics* 4.3 (2009).

[39] Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman. “Power-Law Distributions in Empirical Data”. In: *SIAM Rev.* 51.4 (Nov. 2009).

[40] R. Cohen and D. Raz. “The Internet dark matter-on the missing links in the AS connectivity map”. In: *Proc. IEEE Infocom*. Vol. 6. 2006.

[41] Gibson Research Corporation. *Domain Name Speed Benchmark*. 2012. URL: `https://www.grc.com/dns/benchmark.htm`.

[42] Gabor Csardi and Tamas Nepusz. *The igraph software package for complex network research*. `http://igraph.sf.net`.

[43] Alberto Dainotti et al. “Analysis of country-wide internet outages caused by censorship”. In: *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM. 2011, pp. 1–18.

[44] OrientDB Graph-Document NoSQL dbms. URL: `http://www.orientdb.org/`.

[45] James Demmel et al. *Templates for the solution of algebraic eigenvalue problems: a practical guide*. Ed. by Zhaojun Bai. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2000.

[46] L. Deri et al. “Unveiling Interests and Trends Using the DNS”. In: *IADIS Conference on Internet Technologies*. 2012.

[47] Xenofontas Dimitropoulos et al. “AS relationships: Inference and validation”. In: *ACM SIGCOMM Computer Communication Review* 37.1 (2007), pp. 29–40.

[48] S. N. Dorogovtsev and J. F. F. Mendes. *Evolution of Networks: From Biological Nets to the Internet and WWW*. New York, NY, USA: Oxford University Press, 2003.

[49] Nan Du et al. “A Parallel Algorithm for Enumerating All Maximal Cliques in Complex Network”. In: (2006), pp. 320 –324.

[50] Alan Edelman and N Raj Rao. “Random matrix theory”. In: *Acta Numerica* 14.1 (2005), pp. 233–297.

[51] David Eppstein, Zvi Galil, and Giuseppe F. Italiano. “Dynamic Graph Algorithms”. In: *Algorithms and Theory of Computation Handbook*. Ed. by Mikhail J. Atallah and Susan Fox. 1st. CRC Press, Inc., 1998.

[52] Martin G Everett and Stephen P Borgatti. “Analyzing Clique Overlap”. In: *Connections* 21.1 (1998), pp. 49–61.

[53] Facebook. URL: http://www.facebook.com/.

[54] The Measurement Factory. *dnstop and dsc tools*. 2006. URL: http://dns.measurement-factory.com/tools/.

[55] A. Faggiani et al. “On the Feasibility of Measuring the Internet Through Smartphone-based Crowdsourcing”. In: *International Workshop on Wireless Network Measurements (WinMee)*. 2012.

[56] M. Faloutsos, P. Faloutsos, and C. Faloutsos. “On power-law relationships of the internet topology”. In: *ACM SIGCOMM Computer Communication Review*. Vol. 29. 4. 1999.

[57] M. Feily, A. Shahrestani, and S. Ramadass. “A Survey of Botnet and Botnet Detection”. In: *Emerging Security Information, Systems and Technologies, 2009. SECURWARE ’09. Third International Conference on*. 2009.

[58] Dima Feldman and Yuval Shavitt. “Automatic large scale generation of internet pop level maps”. In: *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*. IEEE. 2008, pp. 1–6.

[59] Santo Fortunato. “Community detection in graphs”. In: *Physics Reports* 486.3 (2010), pp. 75–174.

[60] Santo Fortunato. “Community detection in graphs”. In: *Physics Reports* 486.3 - 5 (2010), pp. 75–174.

[61] Francesco Fusco and Luca Deri. “High speed network traffic analysis with commodity multi-core systems”. In: *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. IMC ’10. New York, NY, USA: ACM, 2010.

[62] Michelle Girvan and M.E.J. Newman. “Community structure in social and biological networks”. In: *PNAS* 99.12 (2002).

[63] *Google Zeitgeist 2012*. Tech. rep. Google Inc, 2012.

[64] J.C. Gower and GJS Ross. “Minimum spanning trees and single linkage cluster analysis”. In: *Applied statistics* (1969).

[65] E. Gregori, L. Lenzini, and C. Orsini. “k-dense Communities in the Internet AS-Level Topology”. In: *Communication Systems and Networks (COMSNETS), 2011 Third International Conference on*. 2011.

[66] Enrico Gregori, Luciano Lenzini, and Simone Mainardi. *Parallel k-Clique Community Detection on Large-Scale Networks*. `http://cosparallel.sourceforge.net/`.

[67] Enrico Gregori, Luciano Lenzini, and Chiara Orsini. *k-clique Communities in the Internet AS-level Topology Graph*. Tech. rep. 2010. URL: `http://puma.isti.cnr.it/`.

[68] Enrico Gregori, Luciano Lenzini, and Chiara Orsini. “k-clique Communities in the Internet AS-Level Topology Graph”. In: *SIMPLEX*. Minneapolis, MN, USA, 2011.

[69] Enrico Gregori et al. “BGP and inter-AS economic relationships”. In: *NETWORKING 2011*. Springer, 2011, pp. 54–67.

[70] Enrico Gregori et al. “FLIP-CPM: A Parallel Community Detection Method”. In: *ISCIS*. 2011, pp. 249–255.

[71] Enrico Gregori et al. “The impact of IXPs on the AS-level topology structure of the Internet”. In: *Computer Communications* 34.1 (2011), pp. 68 –82.

[72] Peter D. Grnwald, In Jae Myung, and Mark A. Pitt. *Advances in Minimum Description Length: Theory and Applications (Neural Information Processing)*. The MIT Press, 2005. ISBN: 0262072629.

[73] Oded Guez et al. “Global climate network evolves with North Atlantic Oscillation phases: Coupling to Southern Pacific Ocean”. In: *EPL (Europhysics Letters)* 103.6 (2013).

[74] Jean-Loup Guillaume and Matthieu Latapy. “Bipartite graphs as models of complex networks”. In: *Physica A: Statistical Mechanics and its Applications* 371.2 (2006), pp. 795 –813.

[75] A. Holmes et al. “Domain Traffic Ranking”. European Patent 2012/EP241753. 2012.

[76] A. Holmes et al. “Existent Domain Name DNS Traffic Capture and Analysis”. US Patent 2010/0257266. 2010.

[77] Sungpack Hong et al. “Accelerating CUDA graph algorithms at maximum warp”. In: *Proceedings of the 16th ACM symposium on Principles and practice of parallel programming*. ACM. 2011, pp. 267–276.

[78] Bruce Hoppe and Claire Reinelt. “Social network analysis and the evaluation of leadership networks”. In: *The Leadership Quarterly* 21.4 (2010), pp. 600–619.

[79] Chengchen Hu et al. “Evaluating potential routing diversity for internet failure recovery”. In: *INFOCOM, 2010 Proceedings IEEE*. IEEE. 2010, pp. 1–5.

[80] Cheng Huang et al. “Public DNS system and Global Traffic Management”. In: *INFOCOM, 2011 Proceedings IEEE*. 2011.

[81] Pan Hui and Jon Crowcroft. “Human mobility models and opportunistic communications system design”. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 366.1872 (2008), pp. 2005–2016.

[82] Pan Hui, Jon Crowcroft, and Eiko Yoneki. “Bubble rap: social-based forwarding in delay tolerant networks”. In: *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*. 2008, pp. 241–250.

[83] Pan Hui et al. “Distributed community detection in delay tolerant networks”. In: *Proceedings of 2nd ACM/IEEE international workshop on Mobility in the evolving internet architecture*. 2007.

[84] ICANN. *Root server attack on 6 February 2007*. Tech. rep. 2007.

[85] Joshua Introne and Sean Goggins. “Tracing Knowledge Evolution in Online Forums”. In: *ACM Web Science Conference: Words and Networks Workshop*. 2012.

[86] Joshua E Introne and Marcus Drescher. “Analyzing the flow of knowledge in computer mediated teams”. In: *Proceedings of the 2013 conference on Computer supported cooperative work*. ACM. 2013, pp. 341–356.

[87] J. Abley and K.Lindqvist. “Operation of Anycast Services”. In: *Internet RFC 4786* (2006).

[88] Adam Jacobs. “The pathologies of big data”. In: *Communications of the ACM* 52.8 (2009), pp. 36–44.

[89] Divya James and Mintu Philip. “A Novel Anti phishing framework based on visual cryptography”. In: *Power, Signals, Controls and Computation (EPSCICON), 2012 International Conference on*. IEEE. 2012, pp. 1–5.

[90] Vitor Jesus, Rui L Aguiar, and Peter Steenkiste. “Topological Implications of Cascading Interdomain Bilateral Traffic Agreements”. In: *Selected Areas in Communications, IEEE Journal on* 29.9 (2011), pp. 1848–1862.

[91] B Jones. *Computational Geometry Database*. `ftp://ftp.cs.usask.ca/pub/geometry/`.

[92] Jaeyeon Jung et al. “DNS performance and the effectiveness of caching”. In: *Networking, IEEE/ACM Transactions on* 10.5 (2002).

[93] S. Kamvar et al. *Exploiting the block structure of the web for computing PageRank*. Tech. rep. Stanford University, 2003.

[94] Anuj Karnik et al. “A Fragile Internet: Non-Technical Issues Leading to Internet Blackouts”. In: *Capstone paper submitted for the degree of Masters in Interdisciplinary Telecommunications at the University of Colorado* (2011).

[95] David Kempe, Jon Kleinberg, and Éva Tardos. “Maximizing the spread of influence through a social network”. In: *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2003, pp. 137–146.

[96] Maksim Kitsak et al. “Identification of influential spreaders in complex networks”. In: *Nature Physics* 6.11 (2010), pp. 888–893.

[97] Jon M. Kleinberg. “Authoritative sources in a hyperlinked environment”. In: *J. ACM* 46 (5 1999), pp. 604–632.

[98] Balachander Krishnamurthy, Walter Willinger, and Brice Augustin. *Reverse Engineering Peering At Internet Exchange Points*. US Patent App. 13/873,316. 2013.

[99] Balachander Krishnamurthy et al. “A Socratic method for validation of measurement-based networking research”. In: *Comput. Commun.* 34.1 (2011).

[100] L Kullmann, J Kertesz, and RN Mantegna. “Identification of clusters of companies in stock indices via Potts super-paramagnetic transitions”. In: *Physica A: Statistical Mechanics and its Applications* 287.3 (2000), pp. 412–419.

[101] Ravi Kumar et al. “Trawling the Web for emerging cyber-communities”. In: *Computer Networks* 31.11-16 (1999), pp. 1481 –1493.

[102] Jussi M. Kumpula et al. “Sequential algorithm for fast clique percolation”. In: *Phys. Rev. E* 78.2 (2008).

[103] Haewoon Kwak et al. “What is Twitter, a social network or a news media?” In: *Proceedings of the 19th international conference on World wide web*. ACM. 2010, pp. 591–600.

[104] Mohit Lad et al. “PHAS: A prefix hijack alert system”. In: *Proc. USENIX Security Symposium*. Vol. 2. 2006, pp. 153–166.

[105] Mohit Lad et al. “Understanding resiliency of internet topology against prefix hijack attacks”. In: *Dependable Systems and Networks, 2007. DSN’07. 37th Annual IEEE/IFIP International Conference on*. IEEE. 2007, pp. 368–377.

[106] Andrea Lancichinetti and Santo Fortunato. “Community detection algorithms: A comparative analysis”. In: *Phys. Rev. E* 80 (5 2009).

[107] Andrea Lancichinetti, Santo Fortunato, and János Kertész. “Detecting the overlapping and hierarchical community structure in complex networks”. In: *New Journal of Physics* 11.3 (2009).

[108] Andrea Lancichinetti et al. “Characterizing the community structure of complex networks”. In: *PloS one* 5.8 (2010).

[109] Andrea Lancichinetti et al. *Finding Statistically Significant Communities in Networks*. URL: http://oslom.org/.

[110] Andrea Lancichinetti et al. “Finding Statistically Significant Communities in Networks”. In: *PLoS ONE* 6.4 (2011).

[111] Conrad Lee et al. *Detecting highly overlapping community structure by greedy clique expansion*. URL: https://sites.google.com/site/greedycliqueexpansion/.

[112] Conrad Lee et al. “Detecting highly overlapping community structure by greedy clique expansion”. In: *Workshop on Social Network Mining and Analysis*. 2010.

[113] Jure Leskovec. *Stanford Large Network Dataset Collection*. `http://snap.stanford.edu/data/`.

[114] Jure Leskovec, Lada A. Adamic, and Bernardo A. Huberman. “The dynamics of viral marketing”. In: *ACM Trans. Web* 1 (1 2007).

[115] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. “Graph evolution: Densification and shrinking diameters”. In: *ACM Trans. Knowl. Discov. Data* 1 (1 2007).

[116] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. “Graphs over time: densification laws, shrinking diameters and possible explanations”. In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. 2005, pp. 177–187.

[117] Kevin Lewis et al. “Tastes, ties, and time: A new social network dataset using Facebook. com”. In: *Social networks* 30.4 (2008), pp. 330–342.

[118] Lun Li et al. “A first-principles approach to understanding the internet’s router-level topology”. In: *ACM SIGCOMM Computer Communication Review*. Vol. 34. 4. ACM. 2004, pp. 3–14.

[119] David Liben-Nowell and Jon Kleinberg. “The link-prediction problem for social networks”. In: *Journal of the American society for information science and technology* 58.7 (2007), pp. 1019–1031.

[120] Cricket Liu and Paul Albitz. *DNS and Bind*. O’Reilly Media, Inc., 2009.

[121] A. Lodhi, A. Dhamdhere, and C. Dovrolis. “GENESIS: An agent-based model of interdomain network formation, traffic flow and economics”. In: *INFOCOM, 2012 Proceedings IEEE*. 2012.

[122] Steve Lohr. “The age of big data”. In: *New York Times* 11 (2012).

[123] Richard TB Ma et al. “Internet Economics: The use of Shapley value for ISP settlement”. In: *IEEE/ACM Transactions on Networking (TON)* 18.3 (2010), pp. 775–787.

[124] Kamesh Madduri et al. “A faster parallel algorithm and efficient multithreaded implementations for evaluating betweenness centrality on massive datasets”. In: *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*. IEEE. 2009, pp. 1–8.

[125] P. Mahadevan et al. “Lessons from three views of the Internet topology”. In: *arXiv preprint cs/0508033* (2005).

[126] Grzegorz Malewicz et al. “Pregel: a system for large-scale graph processing”. In: *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM. 2010, pp. 135–146.

[127] R.N. Mantegna and H.E. Stanley. *Introduction to econophysics: correlations and complexity in finance*. Cambridge University Press, 1999.

[128] Rosario N Mantegna. “Hierarchical structure in financial markets”. In: *The European Physical Journal B-Condensed Matter and Complex Systems* 11.1 (1999), pp. 193–197.

[129] Steve McCanne. *Content distribution system for operation over an internetwork including content peering arrangements*. US Patent 6,785,704. 2004.

[130] Miller McPherson, Lynn Smith-Lovin, and James M Cook. “Birds of a feather: Homophily in social networks”. In: *Annual review of sociology* (2001), pp. 415–444.

[131] Elinor Mills. “Puerto Rico sites redirected in DNS attack”. In: *CNET, San Francisco, California* (2009).

[132] J. Moon and L. Moser. “On cliques in graphs”. In: *Israel Journal of Mathematics* 3 (1 1965), pp. 23–28. ISSN: 0021-2172.

[133] Neo4j. URL: `http://www.neo4j.org/`.

[134] M. E. J. Newman. “Analysis of weighted networks”. In: *Phys. Rev. E* 70 (5 2004).

[135] M. E. J. Newman. “Finding community structure in networks using the eigenvectors of matrices”. In: *PHYS.REV.E* 74 (2006).

[136] Mark EJ Newman. “Mixing patterns in networks”. In: *Physical Review E* 67.2 (2003), p. 026126.

[137] Mark EJ Newman. “The structure and function of complex networks”. In: *SIAM review* 45.2 (2003), pp. 167–256.

[138] W.B. Norton. “The evolution of the US Internet peering ecosystem”. In: *Equinix white papers* (2004).

[139] William B Norton. “Internet service providers and peering”. In: *Proceedings of NANOG*. Vol. 19. 2001, pp. 1–17.

[140] R.V. Oliveira et al. “In search of the elusive ground truth: the internet’s as-level connectivity structure”. In: *ACM SIGMETRICS Performance Evaluation Review*. Vol. 36. 1. 2008.

[141] J-P Onnela, Kimmo Kaski, and Janos Kertész. “Clustering and information in correlation based financial networks”. In: *The European Physical Journal B-Condensed Matter and Complex Systems* 38.2 (2004), pp. 353–362.

[142] J. P. Onnela et al. “Structure and tie strengths in mobile communication networks”. In: *PNAS* 104.18 (2007), pp. 7332–7336.

[143] *OpenDNS Stats*. Tech. rep. OpenDNS Inc, 2013.

[144] Eric Osterweil et al. “Behavior of DNS’ top talkers, a .com/.net view”. In: *Proceedings of the 13th international conference on Passive and Active Measurement*. Berlin, Heidelberg: Springer-Verlag, 2012.

[145] John S. Otto et al. “Content delivery and the natural evolution of DNS: remote dns trends, performance issues and alternative solutions”. In: *Proceedings of the 2012 ACM conference on Internet measurement conference*. IMC ’12. New York, NY, USA: ACM, 2012.

[146] G. Palla, A. Barabasi, and T. Vicsek. “Quantifying social group evolution”. In: *Nature* 446 (Apr. 2007), pp. 664–667.

[147] Gergely Palla et al. “Social group dynamics in networks”. In: *Adaptive Networks*. Springer, 2009, pp. 11–38.

[148] Gergely Palla et al. “Uncovering the overlapping community structure of complex networks in nature and society”. In: *Nature* 435.7043 (2005), pp. 814–818.

[149] Gergely Palla et al. “Uncovering the overlapping community structure of complex networks in nature and society - Supplementary Material”. In: *Nature* 435.7043 (2005), pp. 814–818.

[150] Christopher R Palmer et al. “The connectivity and fault-tolerance of the Internet topology”. In: *Workshop on Network-Related Data Management (NRDM)*. 2001.

[151] J. Park and M.E.J. Newman. “Origin of degree correlations in the Internet and other networks”. In: *Physical Review E* 68.2 (2003).

[152] R. Pastor-Satorras, A. Vázquez, and A. Vespignani. “Dynamical and correlation properties of the Internet”. In: *Physical review letters* 87.25 (2001).

[153] Dan Pei, Lixia Zhang, and Dan Massey. “A framework for resilient Internet routing protocols”. In: *Network, IEEE* 18.2 (2004), pp. 5–12.

[154] David Plonka and Paul Barford. “Context-aware clustering of DNS query traffic”. In: *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*. IMC ’08. New York, NY, USA: ACM, 2008.

[155] Alin C Popescu, Brian J Premore, and Todd Underwood. *Anatomy of a leak: AS9121*. Tech. rep. NANOG, 2005.

[156] Mason A Porter, Jukka-Pekka Onnela, and Peter J Mucha. “Communities in networks”. In: *Notices of the AMS* 56.9 (2009), pp. 1082–1097.

[157] Abraham P. Punnen. “A linear time algorithm for the maximum capacity path problem”. In: *European Journal of Operational Research* 53.3 (1991).

[158] Filippo Radicchi et al. “Defining and identifying communities in networks”. In: *Proceedings of the National Academy of Sciences of the United States of America* 101.9 (2004), pp. 2658–2663.

[159] Smita Rai, Biswanath Mukherjee, and Omkar Deshpande. “IP resilience within an autonomous system: Current approaches, challenges, and future directions”. In: *Communications Magazine, IEEE* 43.10 (2005), pp. 142–149.

[160] R. Rammal, G. Toulouse, and M.A. Virasoro. “Ultrametricity for physicists”. In: *Reviews of Modern Physics* 58.3 (1986).

[161] J. Rissanen. “Modeling by shortest data description”. In: *Automatica* 14.5 (1978).

[162] Ian Robinson, J Webber, and E Eifrem. *Graph Databases*. O’Reilly Media, 2013.

[163] Martin Rosvall and Carl T. Bergstrom. *An information-theoretic framework for resolving community structure in complex networks*. URL: `http://www.tp.umu.se/~rosvall/code.html`.

[164] Martin Rosvall and Carl T. Bergstrom. “An information-theoretic framework for resolving community structure in complex networks”. In: *Proceedings of the National Academy of Sciences* 104.18 (2007).

[165] Martin Rosvall and Carl T. Bergstrom. *Maps of information flow reveal community structure in complex networks*. URL: http://www.tp.umu.se/~rosvall/code.html.

[166] Martin Rosvall and Carl T. Bergstrom. "Maps of random walks on complex networks reveal community structure". In: *Proceedings of the National Academy of Sciences* 105.4 (2008).

[167] M. Roughan et al. "10 Lessons from 10 Years of Measuring and Modeling the Internet's Autonomous Systems". In: *Selected Areas in Communications, IEEE Journal on* 29.9 (2011).

[168] Sercan Sadi, Sule Gündüz Ögüdücü, and A. Sima Etaner-Uyar. "An efficient community detection method using parallel clique-finding ants". In: *IEEE Congress on Evolutionary Computation*. 2010, pp. 1–7.

[169] Kazumi Saito, Takeshi Yamada, and Kazuhiro Kazama. "Extracting Communities from Complex Networks by the k-Dense Method". In: *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* E91-A.11 (2008), pp. 3304–3311.

[170] Matthew C. Schmidt et al. "A scalable, parallel algorithm for maximal clique enumeration". In: *J. Parallel Distrib. Comput.* 69 (4 2009), pp. 417–428.

[171] Hossain Shahriar and Mohammad Zulkernine. "Information Source-based Classification of Automatic Phishing Website Detectors". In: *Applications and the Internet (SAINT), 2011 IEEE/IPSJ 11th International Symposium on*. IEEE. 2011, pp. 190–195.

[172] A. Shaikh, R. Tewari, and M. Agrawal. "On the effectiveness of DNS-based server selection". In: *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*. Vol. 3. 2001.

[173] R. Sibson. "SLINK: an optimally efficient algorithm for the single-link cluster method". In: *The Computer Journal* 16.1 (1973).

[174] Oliver Spatscheck et al. *Multi-Autonomous System Anycast Content Delivery Network*. US Patent App. 12/645,000. 2009.

[175] Isabelle Stanton and Gabriel Kliot. "Streaming graph partitioning for large distributed graphs". In: *Proceedings of the 18th ACM SIGKDD international con-*

ference on Knowledge discovery and data mining. ACM. 2012, pp. 1222–1230.

[176] *State of the Domain*. Tech. rep. Verisign Inc, 2012.

[177] *State of the Internet: Q4 2012 Report*. Tech. rep. Q4-2012. Akamai Technologies, 2012.

[178] James PG Sterbenz et al. “Evaluation of network resilience, survivability, and disruption tolerance: analysis, topology generation, simulation, and experimentation”. In: *Telecommunication Systems* (2011), pp. 1–32.

[179] William Timothy Strayer et al. *Method and system for detecting attack path connections in a computer network using state-space correlation*. US Patent 8,125,898. 2012.

[180] Alan T. Sullivan. “Methods and systems for node ranking based on DNS session data”. US Patent 2012/8090726. 2012.

[181] Hung-Min Sun et al. “DepenDNS: Dependable mechanism against DNS cache poisoning”. In: *Cryptology and Network Security*. Springer, 2009, pp. 174–188.

[182] Robert E. Tarjan and Jan van Leeuwen. “Worst-case Analysis of Set Union Algorithms”. In: *J. ACM* 31 (2 1984), pp. 245–281.

[183] Robert Endre Tarjan. “Efficiency of a Good But Not Linear Set Union Algorithm”. In: *J. ACM* 22 (2 1975), pp. 215–225.

[184] Claudio Tesoriero. *Getting Started with OrientDB*. Packt Publishing Ltd, 2013.

[185] Maksim Tsvetovat and Alexander Kouznetsov. *Social Network Analysis for Startups: Finding connections on the social web*. O’Reilly Media, Inc., 2011.

[186] Twitter. URL: http://www.twitter.com/.

[187] Johan Ugander et al. “The anatomy of the facebook social graph”. In: *arXiv preprint arXiv:1111.4503* (2011).

[188] Thomas W Valente. *Social networks and health*. Oxford University Press Oxford, 2010.

[189] Jacobus Van Der Merwe et al. *Intelligent computer network routing using logically centralized, physically distributed servers distinct from network routers*. US Patent 7,904,589. 2011.

[190] Paul Vixie. “What DNS Is Not”. In: *Queue* 7.10 (2009).

[191] Paul Vixie, Gerry Sneeringer, and Mark Schleifer. *Events of 21-Oct-2002*. Tech. rep. ISC/UMD/Cogent, 2002.

[192] Jian Wang et al. “Model for router-level Internet topology based on attribute evolution”. In: *Communications Letters, IEEE* 13.6 (2009), pp. 447–449.

[193] Jian-Wei Wang and Li-Li Rong. “Cascade-based attack vulnerability on the US power grid”. In: *Safety Science* 47.10 (2009), pp. 1332–1336.

[194] X. Wang and D. Loguinov. “Understanding and modeling the Internet topology: economics and evolution perspective”. In: *IEEE/ACM Transactions on Networking (TON)* 18.1 (2010).

[195] Xiaoming Wang and Dmitri Loguinov. “Wealth-Based Evolution Model for the Internet AS-Level Topology.” In: *INFOCOM*. 2006.

[196] Kanyapat Watcharasitthiwat and Paramote Wardkein. “Reliability optimization of topology communication network design using an improved ant colony optimization”. In: *Computers & Electrical Engineering* 35.5 (2009), pp. 730–747.

[197] Duncan J Watts and Steven H Strogatz. “Collective dynamics of ‘small-world’networks”. In: *nature* 393.6684 (1998), pp. 440–442.

[198] Jim Webber. “A programmatic introduction to Neo4j”. In: *Proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity*. ACM. 2012, pp. 217–218.

[199] Duane Wessels et al. “Measurements and Laboratory Simulations of the Upper DNS Hierarchy”. In: *Passive and Active Network Measurement*. Ed. by Chadi Barakat and Ian Pratt. Vol. 3015. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004.

[200] Jian Wu et al. “Internet routing resilience to failures: analysis and implications”. In: *Proceedings of the 2007 ACM CoNEXT conference*. ACM. 2007, p. 25.

[201] Almerima Jamakovic Yakup Koç and Bart Gijsen. “A Global Reference Model of the DNS”. In: *Proceedings of DNS EASY 2011 Workshop* (2012).

[202] Boru Yan et al. “Detection and defence of DNS spoofing attack.” In: *Jisuanji Gongcheng/ Computer Engineering* 32.21 (2006), pp. 130–132.

[203] Chao Zhang et al. “Modeling Crime diffusion and crime suppression on transportation networks: An initial report”. In: *The AAAI Fall Symposium 2013 on Social Networks and Social Contagion (SNSC)*. 2013.

[204] Guo-Qing Zhang, Di Wang, and Guo-Jie Li. “Enhancing the transmission efficiency by edge deletion in scale-free networks”. In: *Physical Review E* 76.1 (2007).

[205] Shihua Zhang, Xuemei Ning, and Xiang-Sun Zhang. “Identification of functional modules in a PPI network by clique percolation clustering”. In: *Computational Biology and Chemistry* 30.6 (2006), pp. 445–451.

[206] Yun Zhang et al. “Genome-Scale Computational Approaches to Memory-Intensive Applications in Systems Biology”. In: *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*. 2005.

[207] Yuzhou Zhang et al. “Parallel community detection on large networks with propinquity dynamics”. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2009, pp. 997–1006.

[208] Paul Zikopoulos, Chris Eaton, et al. *Understanding big data: Analytics for enterprise class hadoop and streaming data*. McGraw-Hill Osborne Media, 2011.