UNIVERSITÀ DEGLI STUDI DI PARMA

Dottorato di Ricerca in Tecnologie dell'Informazione

XXV Ciclo

Advanced Agent-Based Modeling for Social Networks

Coordinatore:

Chiar.mo Prof. Marco Locatelli

Tutor:

Chiar.mo Prof. Agostino Poggi

Dottorando: *Enrico Franchi*

January 2013

Contents

List of Figures

List of Tables

Chapter 1

Introduction

If we should select because of its outstanding social impact just a single innovation among the ones of the past decade, that should be the diffusion of online social networks. While some social networking services were already active in the nineties, the capillary diffusion and the sheer number of people involved has transformed online social networking in an unprecedented revolution. Online social networks have altered how people interact, allowing them to stay in touch with their acquaintances, reconnect with old friends, and establish new relationships with other people based on hobbies, interests, and friendship circles.

From a technological perspective, online social networks are mostly based on sets of web-based services that allow people to present themselves through a profile, to establish connections with other users in the system and to publish resources, which are typically available to the users with whom they established a connection. Moreover, these systems use common interests and the natural transitivity of some human relationships to suggest new contacts with whom to establish a connection. Although some of these aspects already appeared in other systems, what transformed online social networks in an unprecedented cultural phenomenon is the unceasing flow of information users pour in such systems and their overwhelming intent to increase the number of

their virtual friends and acquaintances.

For our purposes, an Online Social Network (OSN) is defined as a software system allowing users to have a profile and managing a contact network, i.e., an OSN allows users to: (*i*) construct a profile which represents them in the system; (*ii*) create a list of users with whom they share a connection and (*iii*) navigate their list of connections and that of their friends [29]. Although many software systems allowed either to create a profile or to manage a list of contacts, according to Boyd and Ellison, the first website that presented both aspects together was SixDegrees.com, created in 1997 [29].

The first "modern" OSNs is Friendster, created in 2002 and spectacularly failed few years later. In the following years, several of today top OSN were also started: LinkedIn (2003), MySpace (2003), Last.fm (2003), Flickr (2004), Facebook (2004, for college students, and 2006, for everyone), and YouTube (2005).

All the sites together created what can be called the "social network revolution" that changed the way people mean the web, attracting users with very weak interest in technology, including people that before the social networking revolution were not even regular users of other popular Internet services and computers in general [175]. The phenomenon is so widespread that many people started using social networking systems to ask questions to other people instead of querying search engines [134] and in place of regular email.

Although social network analysis is a mature research area by all means, the diffusion of these software systems among people of all ages, gender, instruction and nationality allows for the unprecedented possibility to study quantitatively large scale social systems. Moreover, the sheer amount of data available may require novel techniques of analysis.

The virtual nature of the social networks based by the social networking sites does not facilitate traditional experiments, neither from the ethical nor from the practical point of view. On the other hand, agent-based modeling and simulation has been already successfully applied to problems coming from the social sciences [11, 13, 53–55, 108, 123], and could be profitably used also

for the online social networks.

However, the tools presently available for agent-based modeling do not offer specific support for social network models. Moreover, most agent-based models have been developed in an ad-hoc manner to solve specific problems. On the other hand, many social network models are not formulated with concurrent and ultimately agent-based execution in mind.

Therefore, in the present work, we present a unified conceptual framework to express both novel agent-based and traditional social network models. This conceptual framework is essentially a meta-model that acts as a template for other models.

In addition, we develop a domain-specific language to express the models in an executable way, so that simulations can be performed effortlessly. The language aims at being expressive and powerful for those skilled in the art of computing, and yet simple and easy to learn for those with different backgrounds.

We also develop a software framework that can execute such models in an agent-oriented context, providing effective support for large networks. Moreover, the framework hides most of the complexity of running the simulations on remote server-class machines.

Eventually, we validate out approach translating several traditional models in our meta-model, verifying that the expected features of the models are maintained.

In Chapter 2 we introduce the basic techniques of social network analysis, the metrics used to study the networks and some network formation models. Moreover, we present a review of works where those metrics have been measured on real online social networks, and compare the models with the empirical results.

In Chapter 3 we present multi-agent systems as software engineering tools and discuss how they can be used to support social networking systems. On the other hand, in Chapter 4 we focus on multi-agent systems as modeling tools, thus introducing the discipline of agent-based modeling. We also discuss

the epistemological foundations of the generative approach to science that was pioneered by agent-based modeling.

In Chapter 5 we discuss several issues regarding the different notion of time and concurrency of traditional social network models and agent-based models. We also present some experimental results that show how, under reasonable assumptions, the traditional models can be easily executed in concurrent scenario. We also introduce our meta-model for expressing traditional and agent-based models and show how some traditional models presented in Chapter 2 can be expressed in the meta-model.

Our own domain-specific language for agent-based modeling and the agent-based simulation framework are the subjects of Chapter 6. We discuss the underlying implementation of agents, the runtime system and the structure of the simulation, based on the meta-model. We also show the implementation of a couple on traditional social network models.

In Chapter 7 we apply our tool to the open research problem of creating a distributed social networking system, in order to better investigate (*i*) the necessary assumptions under which such system could be successful, and (*ii*) some metrics to express quantitatively the notion of "successful".

Finally, some conclusions are drawn in Chapter 8.

Chapter 2

Social Network Analysis

Social Network Analysis is a discipline that studies topological and structural features of *social networks*. A social network is typically defined as a finite set or sets of *actors* and the *relation* or relations defined on them [185]. The presence of relational information is critical and defining of social networks.

In this context, an actor is essentially any social entity, such as a discrete individual, a corporate, or collective social unit. The use of the term "actor" does not imply that the entities have the volition or the ability to act. Moreover, it is not related with the concept of actor in the *actor model* [3].

The relationships among the actors can be any kind of social tie and establish a linkage between a pair of actors. Let R be a relationship and x and y two generic actors. If xRy does not entail yRx, R is *directed*. If xRy entails that yRx, then R is *directed* A network with only directed relationships is called a *directed network*; a network with only undirected relationships is called an *undirected network*. Typical examples of relationships are (*i*) evaluations, such as friendhip, respect or trust, (*ii*) transfers of material or immaterial resources, such as information, money or diseases, (*iii*) behavioral interaction, e.g., sending messages, (*iv*) formal or biological relationships, such as marriage, employment or kinship.

In this section we present: (*i*) the introduction with a brief résumé of the

historical background of Social Network Analysis (SNA); (*ii*) a few metrics and measures used to characterize and study social networks (Section 2.2); (*iii*) a selection of models for the simulation of social networks among the models stemming from modern complex network theory (Section 2.3); (*iv*) a summary of metrics measured on real-world social networks (Section 2.4); (*v*) a comparison of the results of the analysis of real-world and simulated social networks, that shows how the simple models do not fully grasp the complexity of real-world social networks (Section 2.5).

2.1 History of Social Network Analysis

As for the historical background, we note that while the history of online social networks begins somewhere between the last years of the past century and the first years of the new millennium, the origins of methods for social network analysis date a century earlier.

The first clear contribution comes from Gestalt Theory through the work of social psychologists such as Moreno [133] and Lewin [118, 119]. Gestalt theory focuses on interpreting patterns as systems with properties distinct from those of their components and that, moreover, determine the nature of such parts. The natural interpretation of Gestalt Theory in the context of psychology and sociology is that of focusing on how social environments determine the choices of individuals.

Moreno was especially interested in the understanding of how psychological well-being is related to structural features that he named "social configurations" [133]. Such configurations are essentially patterns of friendship and repulsion among individuals, that as a whole, constitute large-scale "social aggregates", such as the whole economy of a country.

Although Moreno's methods, which pioneered the use of psychotherapy to understand choices of friendship, questionnaires, controlled observation and experiments, have an everlasting importance in the general context of sociology and psychology, for our purposes his most interesting contribution is that of the

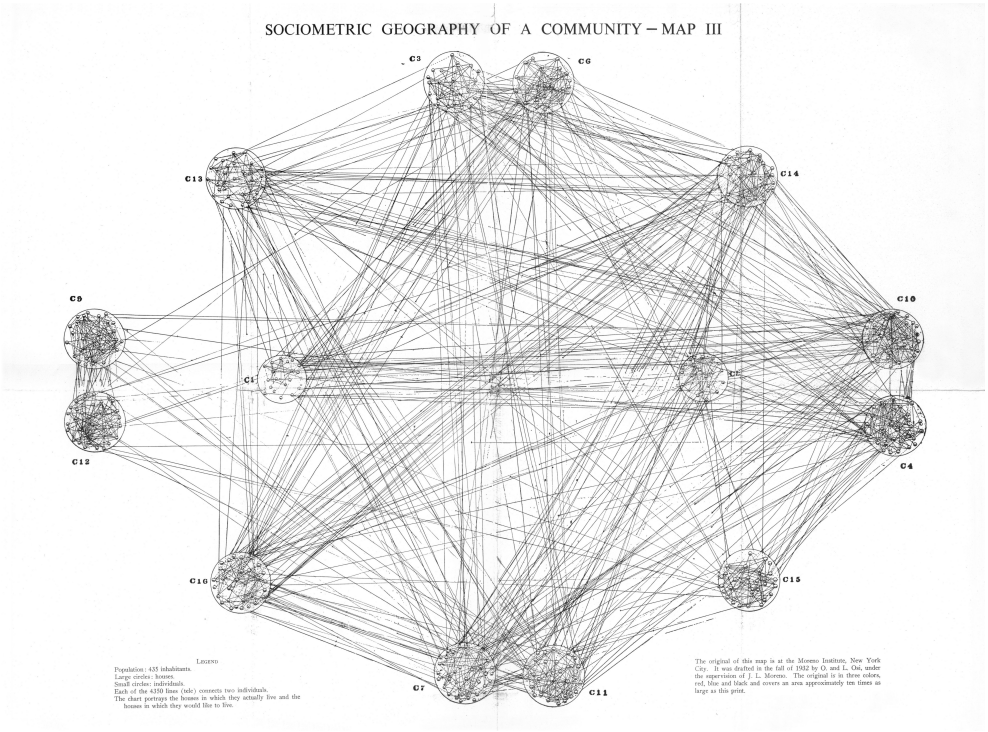

Figure 2.1: Reproduction of Moreno's original sociogram.

sociogram. The sociogram is related to the main body of Moreno's research as a way of representing the formal properties of social configurations; however, a sociogram is a general concept applicable outside his framework and, in fact, outside social psychology altogether. A sociogram is essentially a simple diagram where the points are the social entities and the lines that connect the points represent social relationships.

Nowadays, it is natural to interpret a sociogram as an instance of the graph that can be studied by graph theory. However, at that time it was a relatively novel field of research. The first studies on the subject are due to L. Euler [58] with his solution of the Seven Bridges of Königsberg problem

(1735), but the term graph was coined by Sylvester [176] several years after. The first comprehensive textbook on graph theory, which helped spreading the discipline, was published only in 1936 [110].

Graph theory was gained greater recognition and several important lines of research were conducted, such as *extremal graph theory* and *random graph theory*. Extremal graph theory originated from the study of the old four color problem [191], in which the problem is to determine whether it is true that any map drawn in the plane may have regions colored with one of four colors, in such a way that any two regions having a common border have different colors. The problem, besides having fascinated generations of mathematicians, has a very clear graph theoretic formulation and rose to greater popularity for being one of the first formal problems in which computers played a key role in the proof.

On the other hand, random graph theory started from the early works of Erdős and Rényi [57] and of Gilbert [81], which were the first to considered ensembles of graphs instead of individual graphs. Up to that moment, graphs were individually studied, as is the case of the Königsberg graph. In other words, they concentrated not on any single graphs but on distribution probabilities on spaces of graphs with some specific properties. Eventually, they also played an important role in attracting back to social network analysis the attention of physicists and computer scientists interested in network science [140].

However, back to the early years of social network analysis, its mathematical foundation is essentially due to Cartwright and Haray [89, 90], motivated by the pioneering topology inspired work of Lewin [118]. Haray is also the author of an important textbook on graph theory [88] that was instrumental in spreading the discipline among different areas of research, including social sciences. Consequently, most of the successive social network analysis uses concepts from graph theory, that, as intuited by Moreno, are extremely suitable to express social relationships in a quantitative rather than qualitative way.

Social network analysis developed during the following years, and several

important research directions were undertaken. Some researchers were mainly interested in positions and roles, i.e., they developed the notion of structural equivalence and substitutability of individuals within social positions. Such approaches, usually called "blockmodels", are essentially clustering techniques that organize the networks into hierarchical positions. For a detailed discussion of these ideas, we refer to Chap. 7 of [167].

Another important research area was that of centrality measures. While blockmodels establish the *role* of an individual node in a social network, centrality metrics try to assess the *importance* of the node. Centrality metrics essentially employ the structure of the network and not the semantic meaning of the nodes; in fact, most disciplines dealing with graphs have developed their own notions of centrality. Consider for example:

- centrality in network of citations, authors and journals, which essentially led to the definition of the various indexes used in bibliometrics; or
- Google PageRank [148] or Kleinberg's HITS [104] in the context of ranking web-pages according to their link structure.

One of the most important works that dealt with centrality is due to Bearden et al. and regards the power and influence of american banks [18]. In fact, while some researchers were still mainly concerned with networks of individuals [84,85], that focused on employment patterns, other turned their attention towards network of corporations [165,166,168,174]: by the time social network analysis was a completely different research area from the social psychology that originated it.

Eventually, the third important area of research developed inside social network analysis is the study of community structure inside social networks. Several definitions of "community" have been invented and applied to different networks. Among the most influential works in this direction, we mention the ones of Fisher [61] and Wellman [188,189]. For a complete description of these results, we refer to [185].

Independently from social network analysis, other kinds of networks were studied in the context of Complex Network Theory (CNT). A complex net-

work can be tentatively defined as a network with non-trivial topological features, i.e., features that are not present in regular lattices or simpler network structures such as Erdős-Rényi graphs. Among these features, some typical examples are heavy-tailed degree distributions, high transitivity or, more in general, unusual structural patterns. The definition and discussion of those features is presented in Section 2.2.

The definition of complex network is extremely broad because it must encompass the variety of studies conducted in the area: the networks studied were originally from extremely different areas such as biology, telecommunication engineering and computer science, and complex network analysis provided a unified tool for their interpretation. The discipline draw a lot of attention in the late nineties, because among the networks studied there were samples from the web graph, which was an extremely popular subjects at the time, and it did not take long before network scientists directed their attention towards social networks as well.

Among the most cited papers in the field, there are the seminal works by Watts and Strogatz [186] and Barabási and Albert [16]. Watts and Strogatz studied both social and non-social networks and provided a unifying model for the so-called "small-worlds", i.e., networks where no pair or nodes is really "distant" and where triangle-shaped paths are frequent. On the other hand, Barábasi and Albert's paper introduced a model for scale-free networks, i.e., networks with power-law degree distribution, starting the investigation from a network of web links and a citation graph.

The above mentioned paper, as well as later works created essentially a parallel social network analysis, because social networks increasingly became a primary study for network scientists. Eventually, a few years later, the diffusion of online social networks produced a whole new class of subjects to study that renewed the interest for complex network theory and, as a consequence, many book oriented towards a technical, but typically not academical readership, were published [94, 159–161, 181].

Contrarily to expectations, the traditional social network analysis and the

CNT-inspired social network analysis have only recently started to form a unique discipline: for years the two groups had surprisingly little contact or cross-fertilization [77]. Nowadays cross-citations are becoming more frequent and the work of the early sociologists and social network scientists is increasingly known in the complex network circles. Conversely, ideas grown in the latter field, such as the study of dynamic networks, are starting to attract more attention from the social network analysts.

2.2 Networks Metrics and Measures

As we discussed at the beginning of the chapter, social network analysis is essentially about studying the topological structure of a network. Since networks are essentially identified with graphs, to the point we will use the two terms rather interchangeably, all the algorithms and properties that are studied in the context of graph theory also make sense in the context of social network analysis. However, some of these properties have specific social network analytic meaning in addition to the purely descriptive value that the property has in the context of graph theory. We generally refer as "metric" to any property of a social network or graph that is quantitative in nature.

In this section we briefly review some classic metrics which give insight on the network structure. We consider here only metrics that are among the most widely studied in network analysis and which can be used to characterize the structure of a network; we also outline some correlation in the chosen metrics. The discussion of further advanced analysis techniques such as community or cluster detection, is beyond the scope of this dissertation.

In this Chapter, the properties taken into account are: (*i*) average shortest path length/diameter; (*ii*) clustering coefficient; (*iii*) degree distribution; (*iv*) assortativity coefficient; (*v*) navigability.

Before introducing the metrics, we introduce the notation we use. Let $G = (V, E)$ be a network. With $A(G)$ we refer to the adjacency matrix of the analyzed network; we will omit the G every time it is clear from the context,

and will simply write A.

If u is a node in a directed network:

- the in-degree $k_u^{\text{in}} = \sum_v A_{vu}$ is the number of incoming edges;
- the out-degree $k_u^{\text{out}} = \sum_v A_{uv}$ is the number of outgoing edges;
- k_u is the sum of the in-degree and the out-degree. For undirected networks, the degree k_u is the total number of edges of u.

With $\langle \cdot \rangle$ we refer to the expected value of a quantity. We usually omit to indicate the elements participating in the sum, when this is clear from the context. For example we simply write $\langle k \rangle$ instead of $\langle k_i \rangle_{i \in V}$ to refer to the average degree of the nodes in the network.

In order to compare directed and undirected networks, we ensure that the directed networks are highly symmetrical. The measure of how symmetric is an undirected network is called *reciprocity* (or simply *symmetry*). If m is the number of edges in the network and A is the adjacency matrix of the network, then the reciprocity coefficient is $\frac{1}{m} \sum_{uv} A_{uv} A_{vu}$. The coefficient is trivially 1 for undirected networks. For a directed network, having a high reciprocity coefficient is important from a sociological point of view, because:

- it tells much on the kind of interactions between the actors in the network and
- it allows to soundly compare directed and undirected network with regard to other metrics.

Classic metrics in network analysis are the Average Shortest Path Length (ASPL), the Characteristic Path Length (CPL) and the *diameter*. Let v and v' be two vertices in the network, then $L(v, v')$ is the length of the shortest path connecting v to v' (also called *geodesic path*). The closeness L_i of a node i is the mean of the geodesic distance between i and all the vertices reachable from it, that is to say: $L_i = \langle L(i, j) \rangle_j$. The shortest path length and the characteristic path length are the mean and the median value of all the L_i respectively. The diameter is the longest geodesic path.

In the context of network analysis the diameter, the CPL and the ASPL are said to be *short* if they depend in a logarithmical way on the number of nodes

in the network. Similarily, a link $e = (u, v)$ is a *shortcut* (or *long-range* link) in the network $G = (V, E)$ if $L_{G'}(u, v) \gg 1$ where $G' = (V, E \backslash \{e\})$; otherwise it is a *local* or *short-range link*.

Another very important metric in the context of social networks is the *clustering coefficient* C, which is the mean of all the *local clustering coefficients* C_i, where C_i is the fraction of pairs of neighbors of i which are also connected [187]. A different and non equivalent definition is given by Newman [138], where the clustering coefficient is defined as the fraction of paths (u, v, w) of length two in a network $G = (V, E)$ for which $\{(u, v), (v, w), (w, u)\} \subseteq E$ holds.

The *degree distribution* of a network is simply the frequency distribution of vertex degrees. p_k is the fraction of vertices in the network with degree k. If the network is undirected, then there are two different degree distributions: the in-degree distribution and the out-degree distribution. Although in principle they can be very different, in practice in the examined contexts they are very similar (because the analyzed networks are highly symmetric) and consequently we simply refer to the degree distribution.

A particularly important distribution in the context of social network analysis is the power-law distribution. A power-law distribution has the general form:

$$p_k \propto k^{-\gamma} \tag{2.1}$$

γ is the *exponent* or scaling exponent of the distribution.

Depending on the context, a power-law distribution can be also called a Pareto distribution or a Zipf-law distribution. These distributions have only minor variations in the way the parameters are expressed.

The more distinguishing features of the power-law distribution are (*i*) its left-skewedness, (*ii*) the fat-tail and (*iii*) the straight line form in a log-log plot.

Power-laws are used in many different areas, such as those (*i*) of genus of flowering plants [202], (*ii*) of sizes of cities [171], (*iii*) of citations, scientific productivity and journal use [128, 156], (*iv*) and, eventually, of web links [16], and are believed to denote that a "non trivial" process is involved. As a con-

sequence, many datasets that looked almost linear on a log-log plot have been fitted by power-laws, and, even worse, their exponent has been evaluated by linearly fitting the log-log data. Such procedure is very imprecise [37] and leads to wrong results, especially considering that other distributions could have fit the data better.

Other distributions with heavy tail that may be of interest for us, are the power-law with cutoff or the log-normal distribution (Equations (2.2) and (2.3) respectively).

$$f(x) = x^{-\gamma} \cdot e^{-x\gamma} \tag{2.2}$$

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma x} \exp\left(-\frac{1}{2}\left(\frac{\log(x/\mu)}{\sigma}\right)^2\right) \tag{2.3}$$

However, some processes actually generate a power-law distribution, and some networks actually have power-law degree distributions. They usually are called scale-free, because of the scale invariance property of power-law, i.e., if f is a power-law with exponent γ, $f(c \cdot x) = a(cx)^\gamma = c^\gamma \cdot ax^\gamma = c^\gamma f(x) \propto f(x)$.

Nonetheless, scale-free networks are of particular interest to us because: (*i*) of their historical importance, and especially of the fact that many complex network actually have power-law degree distributions (*ii*) Cohen proved that a wide category of scale-free networks have short diameter *because* they are scale free [41]. To be more precise, it has been proved that if the network has a power-law degree distribution of exponent γ, the diameter d is:

$$d = \begin{cases} \log\log N & \gamma \in (2,3) \\ \log N / \log\log N & \gamma = 3 \\ \log N & \gamma > 3 \end{cases} \tag{2.4}$$

Another very important property related to the degree distribution is the *assortativity coefficient* r, which is basically the Pearson product-moment correlation coefficient of degree between pairs of linked nodes. A network is *as-*

sortative if it has positive assortativity coefficient. The definition of r is:

$$r = \frac{\sum_{ij \in E} \left(A_{ij} - \frac{k_i k_j}{2m} \right) k_i k_j}{\sum_{ij \in E} \left(k_i \delta_{ij} - \frac{k_i k_j}{2m} \right) k_i k_j} \tag{2.5}$$

The last property of social networks we take into account is the *navigability*. We say that a network is navigable if it exists a simple decentralized algorithm that is able to deliver a message to any node, starting from any node, in poly-logarithmic number of steps. With "simple" we mean that each node passes the message to a single neighbor using some ranking function to decide which one. The ranking function must not encompass global knowledge of the long-range links. The *delivery time* of an algorithm is the expected number of steps required to reach the target, randomly choosing the start and the end node.

2.3 Models for Simulated Networks

In this section we consider some models to generate random graphs which have been proposed to simulate social networks. Creating models of social network formation is necessary for improving the understanding of real networks. Creating models allows for easier study of how individual processes concur in the formation of a generic social network; considering that in many cases it is usually not possible to make experiments regarding the formation and evolution of social networks, the only way to assess the basic hypotheses concerning the network formation is by using simulated models.

The first and still most studied model of random graphs is the *Erdős-Rényi model* (ER) [57, 139]. $G(n, p)$ is a probability distribution over the set of all graphs with n nodes. The p parameter indicates that an edge is placed between any given pair of nodes with probability p. Consequently:

- each individual graph is chosen with probability $p^m(1-p)^{\binom{n}{2}-m}$;
- the expected value of the number of edges is $\langle m \rangle = \binom{n}{2} p$;
- the expected mean degree is $\langle k \rangle = (n-1)p$;
- the expected diameter is $\log n$;

– the degree distribution tends to a Poisson distribution for large n;
– the clustering coefficient is given by $C = \langle k \rangle / (n-1)$.

Another very important model is the *Strogatz-Watts model* (SW) [187]. The model starts with a closed linear structure where each node is connected with κ neighbours and then the local connections are rewired to remote nodes with probability p. The rewired connections are usually *shortcuts*. p is a parameter governing the transition from the very regular lattice ($p = 0$, no rewiring) to $G(n, \hat{p})$, where $\hat{p} = n\kappa/2\binom{n}{2}$. In this model, the mean degree is exactly κ. The other metrics are rather hard to derive for this model, however, a minor variant of this model has been analyzed analytically [27, 137, 142]. In this variant the shortcuts are added without removing the local connections. To be more precise, for each link in the lattice a shortcut is added with probability p. Consequently the average number σ of long-range links each node gains is $p\kappa$ according to the distribution:

$$p_\sigma = e^{-p\kappa} \frac{(p\kappa)^\sigma}{\sigma!} \tag{2.6}$$

The average shortest path length is logarithmic with the size of the network, at least for large networks and the clustering coefficient is:

$$C = \frac{3(\kappa - 2)}{4(\kappa - 1) + 8\kappa p + 4\kappa p^2} \tag{2.7}$$

The third model considered here is due to Kleinberg [105, 106] who used ideas somewhat similar to the previous ones, although starting from a different regular structure. In the paper the starting structure is a $n \times n$ 2D grid where each node is connected to some neighbors and shortcuts are also added. All the links are directed. In this model any node u has a position in the grid, which determines its two coordinates u_x and u_y. Let $d(u, v) = |u_x - v_x| + |u_y - v_y|$ is the Manhattan distance between nodes u and v in the grid. There are three parameters $p \geqslant 1$, $q \geqslant 0$ and $\alpha \geqslant 0$ that essentially govern the number of short and long range links of every node: each node u is: (*i*) connected with every other node $v \neq u$ such that $d(u, v) \leqslant p$ and (*ii*) has q other long range

links. A long range link starting from u has endpoint v with a probability proportional to $d(u,v)^{-\alpha}$.

The coefficient α determines the functional dependency of the diameter from the size of the network [41, 143]:

- if $\alpha \in [0,k]$ then the diameter is $\Theta(\log n)$;
- if $\alpha \in (k,2k)$ then the diameter is poly-logarithmic;
- if $\alpha > 2k$ then the diameter is polynomial;
- for $\alpha = 2k$ the diameter length is still an open problem.

The clustering coefficient is naturally quite high (coming from a very regular structure).

However, the most interesting property of this model is that the generated networks are *navigable* and that the delivery time T of any decentralized simple algorithm in the 2D grid based model is:

$$T = \begin{cases} \Omega\left(n^{(2-\alpha)/3}\right) & \text{if } 0 \leqslant \alpha < 2 \\ \Theta\left(\log^2 n\right) & \text{if } \alpha = 2 \\ \Omega\left(n^{(\alpha-2)/(\alpha-1)}\right) & \text{if } \alpha > 2 \end{cases} \tag{2.8}$$

These results have been proven by Kleinberg [105]. This model has been extended to use a k-dimensional mesh as a starting structure. Similar results have been given for the k-dimensional grid models, where $\alpha = 2$ is substituted by $\alpha = k$.

The group model [107] is not a generative one, but it is meant to be used to make any network navigable adding some shortcuts and can also be used to attempt a formal proof of the navigability of a given network. The process starts creating a finite family $\mathcal{S}$ over the set of nodes V satisfying the following conditions for some $\lambda \in (0,1)$ and $\beta > 1$: (*i*) $V \in \mathcal{S}$; (*ii*) $S_i \in \mathcal{S}$ and $|S_i| \geqslant 2$ such that $v \in S_i$, then there exists $S_j \in \mathcal{S}$ such that $S_j \subset S_i$ and $|S_j| \geqslant \min(\lambda g, g-1)$; (*iii*) if *a*) $S_i, S_j, S_k \ldots$ are in $\mathcal{S}$, *b*) have size at most q and *c*) v is in their intersection, then their union has size at most $q\beta$.

The sets in $\mathcal{S}$ are called groups. These conditions hold taking as the groups the balls determined by the Manhattan distance on the k-dimensional grid, for

example. Let $q(u, v)$ be the size of the minimum group in $\mathcal{S}$ containing both u and v. A *group-based model with structure $\mathcal{S}$, exponent α and out-degree m* is a network where for each node u a shortcut to v has been added with probability proportional to $f(q(u, v))$ and $f(x) \asymp x^{\alpha}$ and the process is repeated for m times.

An additional result proved is that for a network (V, E), given an arbitrary finite family $\mathcal{S}$ of sets over V satisfying properties (*i*), (*ii*) and (*iii*), there is a decentralised algorithm with poly-logarithmic delivery time in the group-based model with structure $\mathcal{S}$, exponent $\alpha = 1$ and out-degree $m = c \log^2 n$ for a sufficiently large constant c [107]. Kleinberg also gave negative results for the existence of such algorithm for both $\alpha < 1$ and $\alpha > 1$. Most other metrics depends on the underlying network structure.

Popular models to generate scale-free networks are the ones based on preferential attachment (PA), where links are added more often to nodes with higher degree. In this family of methods the network is generated through multiple steps. At each step some edges and links are added or removed according to some rules that vary from model to model. A popular model of this family is the *Barabási-Albert model* (BA) [16]. The BA model starts with n_0 nodes and no edges. At each step a new node with m random links is added. The m links are directed towards node with probability proportional to their degree. The BA model generates only networks whose degree distribution is a power-law with exponent 3, on the other hand other preferential-attachment models yield scale-free networks with any exponent.

Duchon and Hanusse [50] proved that being scale-free with a degree $\gamma > 2$ implies having a short (poly-logarithmic) diameter. Considering that the diameter is an upper bound of the geodesic paths in the network, the results also bounds the characteristic path length. No such results are available on navigability, it is however reasonable to use the meta-models to add such a property.

The basic PA process or the BA model do not generate networks with high clustering coefficient. For example, it has been empirically found that for a BA

graph $C \sim n^{-0.75}$. No analytical method to compute C for the BA model [6] is available.

A method to increase the clustering coefficient is mingling PA steps with triadic closure (TC) steps. During the TC step, if a link between u and v was added in the PA step, then it is added also a link between u and a random neighbour w of v. This model yields networks with high clustering coefficient and has been extensively studied [93, 177].

Another model in the family of PA models is the *biased preferential attachment* [112]. The set of nodes V is partitioned in three sets P, I and L. These sets represent the different macro-behavioral categories of users the authors have found in existing social networking system:

- P stands for *passive* and are the kind of users that enter the system when invited but usually do not invite new users themselves, nor actively seek their acquaintances in the system;
- I stands for *inviters*, the kind of users who actively invite new users in the system and tend to be at the center of a small cluster;
- L stands for *linkers* and are the kind of users that mainly seek their real life friends in the system and link to them.

At each new step (i) a new node is added to the network and is assigned to one of the three sets according to a distribution of probability p; (ii) $\epsilon > 0$ edges are added to the network. Essentially both p and ϵ are parameters that can be tuned; there is also a third parameter γ. D^β is a probability distribution such that for each node u:

$$D_u^\beta \propto \begin{cases} (\beta+1)\cdot(k_u+1) & u \in L \\ k_u + 1 & u \in I \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

The ϵ edges are added according the following rule: for each edge (u, v), u is chosen with distribution D^0 and (i) if $u \in I$, v is a new node and is assigned to P; (ii) if $u \in L$, v is chosen according to D^γ. No analytical results about the network metrics are presently known. However, the procedure reproduces parameters measured in two real social networks (Yahoo360 and

Flickr) [112]. Consequently we expect that, at least for some choice of parameters, the method yields a network with high clustering coefficient, short diameter and power-law degree distribution.

The last model we review is called *transitive linking* [43]. The model is somewhat similar to the PA model with the addition of the TC step. However, the model also accounts for the possibility that nodes leave the network. In every step of the method two things occur: (*i*) a random node is chosen, and it introduces two other nodes that are linked to it, resulting in a new link (this is the transitive linking, in short TL); (*ii*) with probability p a node is chosen and removed from the network and its edges are removed as well and replaced with another node with one random edge. If the node chosen in (*i*) does not have two edges, then it introduces himself to another random node. The parameter p dictates how often someone is removed from the social network and is assumed to be much smaller than 1.

When $p \ll 1$ the TL dominates the process and the degree distribution is essentially a power-law with a cutoff for larger k, as nodes have finite lifetime. For larger values of p the two different processes concur to form an exponential degree distribution, while for $p \approx 1$ the degree distribution is essentially Poisson distribution. For $p \ll 1$ the clustering coefficient is rather large and can be determined with the relation $1 - C = p(\langle k \rangle - 1)$; as p decreases $\langle k \rangle$ grows. For example, for $p = 0.01$, $\langle k \rangle = 49.1$ and $C = 0.52$. The authors also calculated that:

$$\text{ASPL} \approx \frac{\log(n/\langle k \rangle)}{\log\left(\frac{\langle k^2 \rangle - \langle k \rangle}{\langle k \rangle}\right)} + 1. \tag{2.10}$$

2.4 Analysis of Real-World Social Networks

In the early studies on social networks, the first step was the long manual gathering of data regarding the social network itself, using interviews or other ad-hoc methods. Consequently the social networks were relatively small and biases could be introduced by the sampling methods.

Table 2.1: Basic metrics for a selection of online social networks. The table is organized in subtables where the data are separated depending on the extent of the sampling of each OSN. When in the original paper a datum is missing, we placed "n.a." (not available) in the table. In the table we used some symbols ($\diamond, \blacklozenge, \dagger, \ddagger, \star, \circ$) to indicate that some datum needed further commenting. The symbols are explained hereafter: ($\dagger$) the degree distribution has two different regions, one resembling a power-law with cutoff and exponent $\gamma = 4$ and one with exponent $\gamma = 1$; the crossover occurs between $k = 10^3$ and $k = 10^4$; ($\ddagger$) for nodes with degree less than 300, the degree distribution is approximated by a power-law of exponent $\gamma_{<300} = 1.32$ while that of nodes with degree greater than 300 is approximated by a power-law of exponent $\gamma_{\geqslant 300} = 3.38$; ($\star$) average between the in-degree and the out-degree distribution power-law exponent; ($\diamond$) the out-degree distribution is a power-law with exponent 2.276 up to nodes whose degree is less than 10000, the nodes with higher out-degree are slightly more frequent than what a power-law would predict; ($\blacklozenge$) the degree distributions have a flatter head than power-laws and sharp cut in the tail; ($\circ$) the authors report degree distributions only as graphs.

OSN	Refs.	Users	Links	$\langle k \rangle$	C	CPL	d	γ
Full or nearly full sample								
Club Nexus	[1]	2.5 K	10 K	8.2	0.17	4	13	*n.a.*
Cyworld	[4]	12 M	191 M	31.6	0.16	3.2	16	†
Cyworld T	[4]	92 K	0.7 M	15.3	0.32	7.2	*n.a.*	*n.a.*
LiveJournal	[130]	5 M	77 M	17	0.33	5.9	20	$1.62^\star$
Sample more than 20% of the whole network								
Flickr	[130]	1.8 M	22 M	12.2	0.31	5.7	27	$1.76^\star$
Twitter	[113]	41 M	1.7 B	*n.a.*	*n.a.*	4	4.12	$2.27^\diamond$
Sample around 10% of the whole network								
Orkut	[130]	3 M	223 M	106	0.17	4.3	9	1.50
Sample < 1% of the whole network								
Orkut	[4]	100 K	1.5 M	30.2	0.3	3.8	*n.a.*	3.7
Sample % n.a.								
Youtube	[130]	1.1 M	5 M	4.29	0.14	5.1	21	$1.81^\star$
Facebook	[82]	1 M	*n.a.*	*n.a.*	0.16	*n.a.*	*n.a.*	‡
FB H	[136]	51 K	116 K	*n.a.*	0.41	*n.a.*	29	$^\circ$
FB GL	[136]	277 K	600 K	*n.a.*	0.31	*n.a.*	45	$^\circ$
BrightKite	[164]	54 K	213 K	7.88	0.18	4.7	*n.a.*	$^\blacklozenge$
FourSquare	[164]	58 K	351 K	12	0.26	4.6	*n.a.*	$^\blacklozenge$
LiveJournal	[164]	993 K	29.6 M	29.9	0.18	4.9	*n.a.*	$^\blacklozenge$
Twitter	[100]	87 K	829 K	18.9	0.11	*n.a.*	6	$2.4^\star$
Twitter	[164]	409 K	183 M	447	0.20	2.8	*n.a.*	$^\blacklozenge$

With the widespread use of *social networking systems* by huge amount of people, it became possible to study social networks of unprecedented size. In this section we review a number of papers analyzing different *online social networks* (OSN) and in Table 2.1 we have gathered metrics from some papers [1,2,4,82,100,113,130,136,164] where existing social networks have been measured [23]. The main results that can be deduced from the inspection of Table 2.1 are commented below. Comparing the metrics of such networks with the ones predicted by the models presented in Section 2.3 may give insight on which processes are actually present in online social networks. Such analysis is the subject of the next Section.

While some researchers had access to the full body of data from the analyzed online social network, most of them had still to resort to sampling techniques and consequently we reported the percentage of the sampled social network. Therefore, the table is organized in subtables where the data are separated depending of the extent of the sampling of each social network. For further details on the sampling techniques we refer to the original papers.

In principle, sampling can introduce the same biases typical of earlier studies, and probably some of the studies we reviewed suffered this problem. The biases may be the reason why different studies on the same OSN yield different values on the same metrics. However, some general trends are confirmed by most of the studies we reviewed, and the general structure of online social networks appears not dissimilar to that of off-line social networks.

In fact, sociologists have known since a long time that social networks are highly clustered and OSNs show high clustering as well, see, for example [84]. Like off-line social networks, OSNs have a relatively high clustering coefficient C, orders of magnitude higher than that of random graphs. The actual value of the coefficient exhibits a large variability. Moreover it appears that this coefficient varies much for the same network depending on the sample used and how it was gathered. For example, Ahn et al. measured that the clustering coefficient for Orkut is 0.31 [4], while according to Mislove et al. [130] it is 0.17. Moreover, clustering coefficient is not uniform for nodes of different degree: in

Cyworld the nodes with degree $k < 500$ have a high local clustering coefficient, while friends of nodes with higher degree are not tightly clustered [4]. This contributes to the relatively small global clustering coefficient.

We also argue that different kinds of social networks may well have different clustering coefficients; e.g., it is entirely possible that a real life based social network such as Facebook or Cyworld may have a lower clustering coefficient than an information-based social network such as Twitters. In fact, the Twitter network may be too atypical to be considered a "social" network [113]. In fact, we have a broader definition of social network, essentially encompassing all the networks stemming from relations between humans or human-created entities, such as companies; thus, in any case we consider the Twitter network a social network, even with different topological qualities.

Social networks have short CPL and online social networks present the same trend, confirming the intuition that they actually are small-worlds; almost all measured values of CPL vary between 4 and 6, which is consistent with expected values for small-world networks of comparable size [1, 130]. In the case of Twitter, the CPL is exceptionally small, especially considering the size of the sample analyzed [113]. It is also interesting that studies taking into account the evolution of such social networks point out that the CPL varies in time: typically there is a period when the distance between users increases (which occurs when many new users join) and then when the network becomes more dense the CPL and diameter fall [4, 112, 115, 116].

Some of the analyzed OSNs clearly present a power-law degree distribution; however, the coefficient differ greatly and some OSNs have the coefficient γ of the power-law smaller than 2 [1, 4, 112, 130].

Moreover, we may well say that most of the reviewed networks do not clearly present a power-law degree distribution. For example, Cyworld [4] and Facebook [82] have power-law like behavior on limited degree ranges. Alas, the degree distribution is not scale free and is not a power-law. Twitter [113] more closely resembles a power-law. However, it has many more nodes in the region $k > 10^5$; the authors claim it may be due to a large number of

celebrities participating in the social networking system. But this is exactly the kind of event that model designers should accurately consider and be able to reproduce, even if it contrasts with the uniformity of nodes we implicitly assumed.

The samples collected by Scellato et al. [164] show a rather flat head and a sharp cut in the tail. The authors claim that the effect can be a cause of the samples being obtained with snowball sampling, which is known to under-represent low degree nodes, while the sharp cut may indicate that the degree distribution is a power-law with cutoff.

2.5 Analysis of Simulated Social Network Models

In this Section we compare the models described in Section 2.3 with the real online social networks described in Section 2.4. The models fail to catch some aspect of real world social networks, especially considering how different is the structure of the various OSNs we reviewed in Section 2.4. However, the results are encouraging in the sense that the simplified models miss some process that is actually present in social networks, not that the processes that are part of the models are not present themselves. Probably the most striking result is that the networks presented do not have a power-law degree distribution and, consequently, the "choice" of friends cannot follow preferential attachment alone. The degree distribution is nonetheless "fat-tailed" and some prefererial-attachment-like process is likely involved.

It is not surprising that the *Erdős-Rényi model* fails to describe social networks: from a sociological point of view, it would be awkward to think people establishing relationships purely by chance, regardless of affinity and geographical distance. The expected degree distribution of generated networks also deviates from that of the networks analyzed in Section 2.4: they have heavy tailed distributions instead of a Poisson distribution. Moreover, the average number of connections a person has in a social process does not depend on the size of the network [51]. However, if we used the ER model with a

constant average degree, $C \to 0$ as $n \to \infty$.

The SW model was introduced to cope with the shortcomings of the *Erdős-Rényi model*, in particular the very low clustering coefficient. In the SW model for a very large interval of p values the resulting graph is both highly clustered (like the starting lattice) and shows short characteristic path length like ER random graphs. However, the degree distribution is Poisson and is very different from the distributions found in real social networks, which usually have right skewed degree distributions [141].

Classic preferential attachment models fail to yield highly clustered graphs in most their variants and are consequently unsuitable to model a social network. Indeed, the models have been developed to explain the metrics of citation networks. However, many variations on the original model which have a rather high clustering coefficient have been proposed in order to explicitly model social networks.

For example, the *biased preferential attachment* model [112] was built from the ground to reproduce some metrics of real world social networks. In fact, the model does not only yield a static social network with realistic properties: the entire generation process reproduces the formation of the real network.

The transitive linking model also looks very promising. People is far more likely to make friends with friend of friends [84]. The importance of adding explicit triadic closure steps has also been proved by Leskovec et al. [115], where the authors showed that regular PA, without steps adding local links between friends of friends failed to model real social networks. Moreover, node death (or inactivity, for the less dramatically inclined readers) moves the degree distribution from a pure power-law to a power-law with cutoff. The other metrics are also compatible with the experimental data of Section 2.4.

In fact, we have chosen few metrics in order to simplify the analysis. However, at the present time, these simple metrics are enough to rule out many network generation models and to question the remaining ones, as well as giving insight on which parts may need adjustment. Moreover, from studies such as [4, 112, 115, 117] it is clear that OSN are constituted by distinct areas with

very different topological structure and we believe that this should be taken into account.

In the present study we are only concerned with the metrics of the final network, with little focus on the metrics along the process itself, while [112] and [115, 117] deal with processes that reproduce OSN metrics during the whole formation process. We decided to consider only the static analysis of the resulting network because: (*i*) many models are not meant as proper processes, since what looks like a process is in fact only an algorithmic description; (*ii*) it is particularly difficult to sample large social networks over long periods of time and consequently there is less data on the issue. Moreover, we are mostly interested in performing simulations on the final network. Nonetheless, it is worth noting how processes which are inspired by actual human behavior [43, 93, 112, 177] are some of the most promising models.

2.6 Conclusions

In this Chapter we described the history and the main subject of social network analysis. Social network analysis is the main tool we use to study networks for various purposes, such as the feasibility of the the approach for creating a distributed social networking system that we we discuss in Chapter 7.

Moreover, we introduced some of the models that are at the basis of the construction of the meta-model for simulating processes over network presented in Chapter 5, and subsequently, we compared the features of the networks they generate with those of real online social networks. Such models offer interesting insights regarding the basic processes that drive the formation and evolution of networks, but fail to describe other laws that are likely to be present. Moving towards agent-based modeling (Chapter 4) could provide models that can more easily cope with the complexity of human interactions and more faithfully describe the processes involved.

Chapter 3

Multi-Agent Systems

3.1 Introduction to Multi-Agent Systems

Software engineering is the discipline that studies in a systematic and rigorous way the problem of designing, developing and, successively, maintaining software. Software (*i*) does not usually exist independently of hardware and (*ii*) is created to solve specific problems. As a consequence, from one side, software evolves to fulfil our necessities and, on the other, to better exploit the hardware infrastructures available. Software engineering develops new methods in order to cope with such evolution.

In the last twenty years, several trends in software can be clearly observed: (*i*) ubiquity, (*ii*) interconnection, (*iii*) intelligence, (*iv*) delegation, and (*v*) human orientation [194].

As for ubiquity, computer-grade computational capabilities appeared in several devices such as mobile phones or music players. Moreover, many electric appliances have capabilities to interact with computers in several ways.

The trend towards interconnection is even more evident. Not many years ago, connection to the internet was an option for few people. On the other hand, nowadays, it is not uncommon to own several devices that are always connected, which synchronize data with the cloud. In fact, many software

systems are created with the assumption of ubiquitous connectivity.

Another important trend is that of creating increasingly "intelligent" and complex systems and to delegate them important tasks. Autonomous machines manage, for example, airplanes, telephone grids, power-plants, medical appliances, weapon systems, and economic markets.

Eventually, as user interfaces moved from the command line oriented designs of the seventies to the graphical user interfaces of the nineties, they are presently evolving towards even more declarative and intuitive paradigms, such as multi-touch or vocal interfaces.

One of the most interesting answers given by software engineering to the necessities we framed are Multi-agent Systems. A Multi-agent System (MAS) is a system composed of multiple intelligent agents interacting within an environment. However, the development of multi-agent systems should not be traced entirely inside the software engineering community, and, in fact, they originated from the confluence of different research areas, such as distributed systems, artificial intelligence, game theory and social sciences [194]. In fact, all those aspects are important to face the challenges that developing modern software poses.

The inherent interconnection among systems requires techniques from the distributed systems area, because multi-agent systems are distributed systems and should be understood and conceived as such. Even if all the agents operate in the same computational environment, concurrent execution and communication can only be solved using techniques from such domain.

However, since MAS may be constituted by non-necessarily cooperating units, also techniques from game theory in general (and mechanism design specifically) are required. Eventually, since the network of interconnected devices essentially creates an artificial society, insight from the social sciences becomes precious.

Moreover, since we want software to autonomously take decisions and solve tasks, techniques from game theory and artificial intelligence must be used. Eventually, in order to create more intuitive and declarative interfaces, nat-

ural language must be understood, perhaps even spoken language, which are problems typically studied in artificial intelligence.

As a consequence of the mixed origins of multi-agent systems, there is little consensus on the main defining features. For example, there is not even a single definition of an agent – see, for example, [79, 162, 195]; however, all definitions agree that an agent is essentially a special software component that: (*i*) has autonomy, in the sense that it operates without the direct intervention of humans or others and has control over its actions and internal state, and (*ii*) provides an interoperable interface to an arbitrary system, i.e., has clearly distinguishable boundaries that define what is part of the agent and what is outside.

Another defining feature of agents is "intelligence", though in this context it has a different meaning of that used in the context of game-theory[1]. In the context of multi-agent systems intelligence is usually defined in terms of autonomous properties; we expect intelligent agents to be: (*i*) *reactive*, because it perceives its environment, and responds in a timely fashion to changes that occur in the environment; (*ii*) *pro-active*, because it does not simply act in response to its environment, and it is able to exhibit goal-directed behavior by taking the initiative; and (*iii*) *social*, because it is able to interact with other agents (and humans) in order to satisfy its design objectives.

Multi-agent systems are generally considered appropriate for modeling complex, distributed systems, even if such a multiplicity naturally introduces the possibility of having different agents with potentially conflicting goals. Agents may decide to cooperate for mutual benefit, or they may compete to serve their own interests. Agents use their social ability to exhibit flexible coordination behaviors that make them able to both cooperate for the achievement of shared goals or to compete on the acquisition of resources and tasks. Agents have the ability of coordinating their behaviors into coherent global actions.

Another important area of multi-agent systems is that of modeling and

[1] An agent is *intelligent* if it knows everything that we know about the *game* and it can make any inferences about the situation that we can make. [135]

simulation: the use of MAS is especially appropriate for the modeling of systems that are characterized by a high degree of localization and distribution and dominated by discrete decisions. In particular, the use of such an approach gave important results in social science because it allows a very useful means for studying social phenomena by providing a natural way of representing and simulating both the behavior of individuals or groups of individuals and their interactions that concur to the global behavior [11, 53, 55]. The discussion of agent-based modeling and simulation is the main topic of Chapter 4.

Over the years, MAS researchers have developed a wide body of models, techniques and methodologies for developing complex distributed systems, have realized several effective software development tools, and have contributed to the realization of several successful applications. However, even if today's software systems are more and more characterized by a distributed and multi-actor nature, that lends itself to be modeled and realized taking advantage of multi-agent techniques and technologies, very few space in software development is given to the use of such techniques and technologies. Researchers have proposed several explanations of the low adoption of multi-agent techniques [30, 46, 125, 126, 190].

A detailed discussion of such reasons is beyond the scope of this dissertation; however, we believe that one of the most important reasons is that software developers usually have limited knowledge about MAS technologies and solutions, and, moreover, the multi-disciplinary approach of MAS is itself a limiting factor in its adoptions, since many MAS techniques require specific training in the field.

Essentially, many multi-agent approaches are too sophisticated and hard to understand and to be used outside the research community. To support this thesis, we notice how Agent-based Modeling (ABM) is a very successful branch of MAS research and in ABM several advanced issues are not present. For example, in ABM agents usually have limited intelligence and autonomy [49] and, moreover, ABM targets essentially researchers, albeit not necessarily coming from the main disciplines originating MAS.

The reaction of the MAS community has been the creation of a wide range of tools which take into consideration the necessities of the industry, essentially moving from systems with heavyweight agents to systems with heterogeneous and hybrid agents [21, 74, 153]. The discussion of such classes of agents is the subject of Section 3.2.

3.2 Evolution of Multi-Agent Systems

The definition of "software agent" covers a wide variety of computer programs, ranging from so-called "heavyweight agents", with higher internal complexity and cognitive capabilities, to so-called "lightweight agents", with simple or no internal state and exposing mainly reactive behavior. Between the two extremes there is a continuous spectrum of different types of software agents, with some features taken by both main types: (*i*) deriving from different agent theories, for the description of the mental states of agents; (*ii*) implementing different agent architectures, which guide the development of real systems from abstract theories, and (*iii*) featuring different programming constructs or specialized languages.

Heavyweights, intentional agents. According to various cognitive theories, software agents can be usefully described in terms of the intentional stance, i.e., on the basis of their mental attitudes, such as knowledge, beliefs, wants etc. The mental attitudes could be about facts in general, but in some cases they could also regard other agent attitudes, thus allowing for higher order reasoning. There is not a single theory for describing the cognitive model of software agents. Traditional AI studies led to various mental models that have been refined and applied to software agents. One of the most widely accepted theories describes agents in terms of their BDI [158]. Mainly during the 1980s, there has been a shift of researchers' attention from monolithic intelligent systems to Distributed Artificial Intelligence. Wooldridge [193], in particular, developed some logic theories for describing Multi-Agent Systems. In those

systems, communication plays a fundamental role. The speech act theory, deriving from previous studies conducted by Austin [9], Searle [169], Cohen and Perrault [40], Cohen and Levesque [39], has often been applied to MAS, because it allows to associate message performatives and message contents with the mental state of agents. According to this theory, agent communications are essentially pragmatic actions, performed with the intention of procuring some change in their world. Among the various types of speech acts, probably the most easily distinguished are directives, e.g., for request messages, and representatives, e.g., for inform messages. Various frameworks for BDI agents have been realized, including Jason [28], JACK [192] and Jadex [155].

Lightweight, swarming agents. Lightweight agents are developed essentially in contrast with traditional AI and cognitive theories in general. The principal question was the applicability of cognitive theories to concrete problems, with highly uncertain and changing perceptions about the world, and multifaceted data, which easily become unmanageable through a cognitive approach, as reasoning typically requires an exponential complexity versus the size of the problem. The theory behind the lightweight approach to developing multi-agent systems derives from various research works and in particular from the famous and provocative "subsumption architecture" proposed by Brooks [32]. In lightweight MAS, in fact, intelligence is not supposed to exist in the symbolic reasoning of agents, but it emerges as the result of the continuous perceptions and reactions of multiple simple agents, possibly coordinating their actions without explicit and direct exchange of knowledge.

Intermediate agents. Between those two contrasting approaches, various practical systems are modeled using software agents that are neither fully intentional entities with cognitive capabilities, neither purely reactive entities, without any internal state. Among these systems, many are designed in terms of a Finite State Machine (FSM). The internal status of agents determines the kind of behaviours it exposes [36]. The single behaviors can range in complexity

from symbolic representation and reasoning systems to much simpler stimulus-response rules. Among these systems, MANA [114] and EINSTein [79], behave on the basis of matrix multiplications, with a vector of weights representing the dynamical "personality", i.e. the effect of environmental stimuli on agents. On the other hand, more complex systems [149, 151] are based on uncertain knowledge. Those propositions are represented as the nodes of a Bayesian network and processing consists in propagating evidence across the network; thus, they combine symbolic and numeric processing. Also in Agentcities, a large international research project advancing agent technologies, many deployed agents exploited the semantics of the Agent Communication Language (ACL) proposed by the FIPA for high-level service composition, yet they based their internal operation on the intrinsic FSM model of JADE behaviours [154]. Being other parts of the project developed according to a BDI model, the overall system was quite variegated, with respect to both agent technologies and models, but with all parts adhering to FIPA specifications for interoperability.

Heterogeneous systems. Various research works highlight the advantages and issues of integrating swarming and intentional agents. Parunak et al. [149] describe various experiences in projects of this kind, with quite different requirements and architectures. In TacAir-Soar and delegate MAS, they propose a delegation from heavyweight agents to lightweight agents, whose result are either taken into account as a whole, or considered individually. In Transitional Agents and Ant CAFÉ, they propose to dedicate heavyweight agents to the interaction with users, while assigning pre-assembled tasks to lightweight agents. In another presented projects, instead, they follow a quite unconventional approach of using heavyweight agents as problem-solvers for producing basic results, and swarming agents for their final integration. The idea of heterogeneous agent systems was already envisioned in early works on software agents. For example, Genesereth proposes the idea of an ACL (KQML, specifically) as an interoperability tool for the integration of heterogeneous systems [79]. More specifically, with heterogeneous agent systems we mean the integration of ar-

bitrarily different agent models, not necessarily sharing a single language or communication medium. Apart from the communication issues among agents expecting intentional messages and agents interacting through a sense-react cycle, Parunak et al. [149] highlight other integration issues. In particular, they cite internal issues, i.e. designing different agents, but are able to deal with their tasks, and system issues, i.e. dedicate the various types of agents to the most appropriate tasks. Nevertheless, the authors highlight the advantages of mixing the two kinds of agents, thus exploiting both the high level intentional behaviours of BDI agents and the scalability of swarming agents to the size of practical problems.

3.3 MAADE

After considering the state of the art in MAS, Poggi and coworkers [73] developed MAADE, a software framework created to simplify the realization of distributed, pervasive and adaptive applications, merging the client-server and the autonomous agent paradigms.

The idea is that developers with a background in MAS can use MAADE to create with little effort systems where lightweight and heavyweight agents coexist. However, MAADE can also be used by developers without specific MAS background to create distributed systems using the agents simply as a concurrency primitive.

MAADE allows the realization of systems based on two types of processes: (*i*) agents, either cognitive or not, and (*ii*) servers. Agents and servers can be distributed on a (heterogeneous) network of computational nodes (from now called runtime nodes), connected using different protocols, and still constitute a single application.

An agent is an active process with reactive and pro-active behavior. However, not necessarily both aspects are used in the same agent. Pro-active behavior means that agents can autonomously start the execution of tasks. Agents run either in their own thread of execution, or rely on a thread-pool. They

perform tasks sending synchronous and asynchronous messages to both servers and other agents.

Servers, on the other hand, have only reactive behavior, similarly to servers in the standard client-server paradigm. However, they can compose services offered by other servers and agents in order to fulfil the requests. Servers are also the more suitable components to provide services to external applications, exposing appropriate public interfaces.

Agent Model

In MAADE, an agent has a unique identifier in the system, a type and a behavior, which determines its actions and reactions. A registry of all the agents in the application exists and can be queried according to type or for a given identifier. Moreover, an agent can advertise a description of its capabilities and the registry can be queried accordingly.

An agent can interact with the other agents through the exchange of messages based on one of the following three types of communication:

- synchronous communication, the agent sends a message to another agent and waits for its answer;
- asynchronous communication, the agent sends a message to another agent, performs some actions and then waits for its answer;
- one-way communication, the agent sends a message to another agent, but it does not wait for an answer.

In fact, agents do not exchange messages unmediated. Each agent is associated with a mailer that: (*i*) receives the incoming messages from the other mailers, (*ii*) buffers the incoming messages until the agent is ready to receive them, and (*iii*) forwards the outgoing messages to the appropriate agents.

Each mailer has two lists of message filters: the elements of the first list, called input message filters, are applied to the input messages and the others, called output message filters, are applied to the output messages. Figure 3.1 shows the flow of the messages from the input message filters to the output

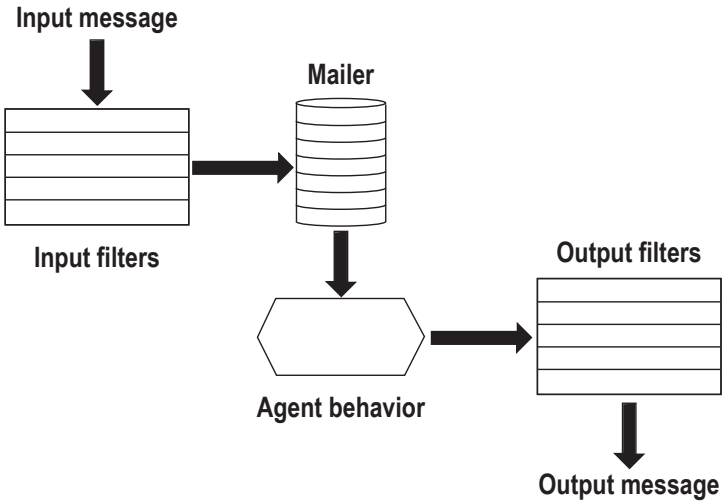

Figure 3.1: Simplified representation of the agent model.

message filters). When a new message arrives or must be sent, the message filters of the appropriate list are applied to it in sequence until a message filter fails; therefore, such a message is stored in the input queue or is sent only if all the message filters have success.

As for the formal definition, a message filter is a composition filter [26] whose primary scope is to define the constraints on the reception/sending of messages; however, it can also be used for manipulating messages (e.g., their encryption and decryption) and for the implementation of replication and logging services.

The same mechanism used by message filters to decide whether they should process any given message is also used by the agent to query the mailer for specific messages that it may need to complete its task.

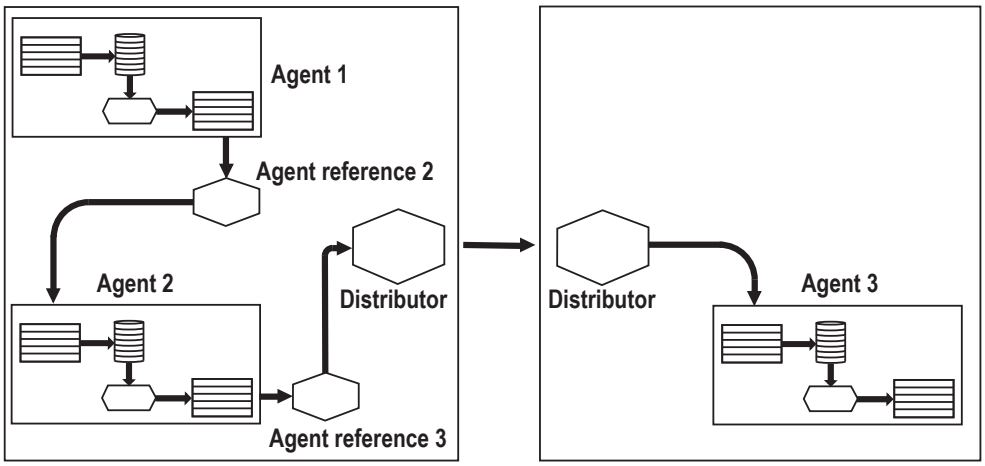

Figure 3.2: Exchange of messages in a distributed system.

Agent Communication

MAADE does not mandate a specific ACL, but allows the use of one of the ACLs provided by the environment. The current release provides: (*i*) an implementation of the FIPA ACL [66], (*ii*) a simple language, called MACL, which supports the typical request-response interaction, (*iii*) the possibility to implement another well-known ACL (e.g., KQML [60]), and (*iv*) the possibility to define new ACLs.

This solution allows the implementation of both "standard" multi-agent systems based, for example, on FIPA ACL, which allows the interoperability with other multi-agent systems, and "specialized" multi-agent systems where a "custom" ACL can, for example, reduce the development cost and/or improve the performances.

Agents can also communicate with agents in different runtime nodes. The communication is mediated by the distributors of the respective nodes. MAADE is agnostic with respect to the communication technology used by distributors;

support is provided out of the box for Java RMI [152], JMS [132] and MINA[2] communication technologies. In Figure 3.2 we summarize the functionality provided by distributors.

3.4 Multi-Agent Systems for Social Networks

In this Section we review the state of the art regarding the creation of MAS to support social networking applications [24, 71] and we draw connections between the literature and our work, described in Chapter 7. The works that deal with ABM of social networks are discussed in Section 4.4. Here we consider two categories of MAS: (*i*) systems that essentially implement some feature typical of OSN and (*ii*) platforms, middle-wares or frameworks that support OSNs.

As for the first category, the most widely studied aspect that is present in modern OSN is that of expert-finding, referral, or matchmaking. Such features are specifically part of work-related OSN, such as LinkedIn, or dating systems, which can be considered the direct ancestors of all modern OSNs [29].

A seminal work in this area is ReferralWeb [101, 102], an interactive MAS that "reconstructs" a social network from data available on the web. In the reconstructed network two people are connected if their names appear in close proximity in a web page or if there is a documented co-authorship relationship.

A ReferralWeb user can (*i*) require the referral chain between himself and a named individual , (*ii*) search an expert in a given topic, providing a maximum distance, or (*iii*) request a list of documents written by people "close" to a specified expert.

A detailed discussion of agent-based algorithms providing answers to similar queries is presented by Yu et al. [198–201], and discussed in an experimental context.

ReferralWeb is truly an innovative system, considering that it somewhat provided some of the features that made systems such as LinkedIn popular. In

[2] `http://mina.apache.org/`

fact, while modern social networking systems require their users to manually insert the information about themselves and OSN providers are investing large amounts of money for tools that somewhat automate the process[3], the MAS provides such features since the beginning.

Another ReferralWeb contemporary system, Yenta [63–65], provides somewhat complimentary features. Yenta uses agents to implement a full-fledged match-making system, i.e., a system to help people with similar interests to get in touch. Yenta did not use web-pages to gather information, preferring to scan mails, documents and usenet posts instead.

Similar features are provided by modern social networking platforms, albeit with a slightly different focus: nowadays people are mostly interested in socializing with "real-life acquaintances", on the other hand, in the nineties, people used internet to interact with others on the basis of common interests.

An important aspect of Yenta was that a multi-agent approach was adopted not only for efficiency reasons, i.e., for distributing the computation among the many nodes involved in the network, but also for privacy concerns. Foner [65] felt that it could be unlikely that users would grant full trust to a third party entity and give it their documents, so that it could analyze them and answer with a list of possible connections. Agents, on the other hand, can run on the local machine, so that the full documents is not disclosed and only the few pieces of information necessary for establishing the connection could be exchanged. We have also studied a similar problem and reached a similar conclusion on the suitability of agents for the task [68].

Although history proved Foner wrong regarding the dispositions of users towards privacy, in that apparently most people do not seem to care about the confidentiality of the data they provide until some incident occurs, he was absolutely right in that users should have been concerned with such issues. For similar reasons, we created our own distributed privacy-aware OSN [72].

In more recent years, other works considered the development of multi-

[3] http://www.eweek.com/c/a/Messaging-and-Collaboration/LinkedIn-Buys-Rapportive-Gmail-Contact-Plugin-208870/

agent recommender systems. For example, Gursel et al. [86] applied the ideas to photo searching and recommendations for the popular social networking system flickr.com. Other works used MAS technologies in social networking contexts. Walter et al. present a model of a recommender system based on social networks, autonomous agents and trust relationships [184]. The aim is to both reach information not available in close nodes and filter information to be processed. The system is analyzed with varying network density, preference heterogeneity and knowledge sparseness.

Although, many MAS focus on specific aspects of OSNs, virtually no major social networking platform is based entirely on multi-agent systems. Even in the contexts where the relatively lower operative costs could ease the adoption of the systems [22], agent-based solutions are not widely adopted.

Among the research works, MAgNet [17] is a multi-agent system built using JADE [20] and FOAF [31]. It is a prototype application providing social oriented services to mobile users, i.e. defining groups of users and arranging group events.

Another distributed social networking system was being built in Japan roughly 10 years earlier. Hattori et al. [91] deal with network communities and not specifically with social networks; however, their definition of network community as a collection of personal units, community agent(s), and the set of relations between them, is in fact also a definition for a social network.

Hattori et al. system is composed of many personal units, each consisting of a user and a personal agent. Each personal agent helps the user by gathering and exchanging information, visualizing contexts, and recommending or assisting the user in making a choice. All the agents in the system cooperate and act as a unit, with the users being the central figures. The community agents have the function of providing shared information, knowledge, or contexts within the community and act as mediators for informal communications between people.

The ideas expressed in the paper have been implemented in a first version of a prototype called “Community Organizer”. The main difference between

the prototype and the system described so far is that the actual system has just one (essentially centralized) community agent, which essentially means the implemented system was a centralized system. On the other hand, Shine (SHared INternet Environment) [197] is a fully Peer-to-Peer (P2P) framework for network community support. The framework also provides design guidelines and enables different applications to share program components and cooperate and features a peer-to-peer architecture through which personal agents can flexibly form communities where users can exchange information with peer agents.

Not dissimilarly from "Community Organizer", Shine provides a personal agent to each user and several core modules compose each agent: Person database, plan execution module and communication module. In addition, one or more applications are installed in each agent. These applications provide their services to the user using functionalities of the core modules via an API.

All considered, we feel there is a huge disproportion in the number of MAS that implement specific aspects of OSN and those that implement the whole system. On the other hand, many non agent-based distributed systems are being created [5, 33, 34, 42, 72, 75, 83]. While some aspects of a full-fledged social networking platform are suitable for agent-based programming, for other aspects the advantages are relatively minor and more traditional techniques are as successful. All considered, probably the most promising route is developing hybrid systems in which agents are used where they can shine and other techniques are used were agency is not necessary. We adopted a similar approach for our own work [72]

Chapter 4

Agent-based Modeling and Simulation

ABM is a very powerful technique that has been applied increasingly often in the last years in a variety of different contexts. Examples of those contexts are (*i*) social sciences [11, 13, 53–55, 108, 123], (*ii*) economy [8, 14, 56], and marketing [146], (*iii*) epidemics and medicine [15, 35, 62], (*iv*) archeology [45, 109], and (*v*) warfare [92, 97, 131].

However, agent-based modeling is more than a simulation technique that is appropriate for inherently complex problems: agent-based modeling is a cognitive approach to science. In Section 4.1, this point of view and the epistemological foundations of agent-based modeling are laid out; agent-based modeling is also compared with traditional equation-based modeling. After discussing in Section 4.2 how agent-based models should be designed, in Section 4.3 agent-based modeling is discussed in the context of social sciences and then, in Section 4.4, some successful agent based models related to social networks are presented. Eventually, in Section 4.5 the more popular tools for agent-based simulation are compared. These tools will be compared with the one developed in this work in Chapter 6.

4.1 Epistemological foundations of Agent-based Modeling

In ABM the subject of the modeling is described in terms of a collection of autonomous decision-making unit, called agents, that are grouped together to form an agent-based model. Each agent individually assesses the situation and makes its own decision. A model consists of a group of agents, their individual behaviors and their mutual relationships.

The ideas behind agent-based modeling are rather simple: the decision process is decentralized and distributed among the simulated unit, which are basically given the same information their real counterparts have. Moreover, the interactions among the agents tend to be simple and somewhat limited. Simpler interactions are usually easier to factor out from the system to model and, moreover, often agents formally act under the hypothesis of bounded rationality [170], i.e., they are limited (*i*) by the information they have, (*ii*) by their cognitive abilities, and (*iii*) by the finite amount of time available to reach a decision.

Nonetheless, self-organization, patterns, structures and behaviors that have not been explicitly programmed arise from the individual interactions. In fact, *emergence* is one of the most peculiar features of ABM. Essentially, the whole point of ABM is studying the emergence of the macro-level features of interest from the explicitly programmed micro-level interactions.

The intuitive notion of emergence, i.e., the manifestation of some property that seems to appear spontaneously in the system, is not hard to grasp and is quite a fascinating idea. Consequently, the notion has been often abused in contexts where it is not appropriate and, most of the times, is not defined at all. Researchers, such as Epstein and Axtell [56], and Axerlrod [10] frame the context of emergence defining an emergent phenomenon as a stable macroscopic pattern arising from the local interactions of the agents.

We care to stress the part where in the definition the cause of the phenomenon is given in the interactions of the agents: for example, let us consider

a simulation of the migrations of birds around the world. A careful modeler would encode the global wind patterns of each region of interest, perhaps in the environment, see Section 4.2. According to Thorup et al. [179] older birds of prey are better at correcting the wind drift than juvenile ones. An agent-based simulation could easily take into account the individual abilities to correct wind drifts. However, if the simulation showed some large scale effect, e.g., that there are different wintering areas for juveniles and elder birds, such phenomenon could not be called "emergent", because it does not arise from the *interactions* of the agents, but only on their "individual features".

On the other hand, in the classic ant colony optimization problem, where a minimum path between the colony and a source of food is found, it is the interaction of the ants by leaving appropriate pheromone that elaborates a short path. That is an *emergent* effect: the minimizing effect is not encoded in the ant "program", it emerges from their interaction.

Another important consideration with emergence, is that perfect knowledge of the micro-level interactions does not allow to predict macroscopic structure, so emergece cannot be usually predicted. However, agent-based models specified in terms of micro-level interactions can be *executed* so that the macro-level structure can be actually observed.

In fact, agent-based modeling is more than just a modeling and simulation technique. Agent-based modeling is an epistemological paradigm shift is the way science is done. Traditional sciences usually follow (*i*) a *deductive* approach, where phenomena explanations are deduced from more general rules, assumed correct, (*ii*) an *inductive* approach, where the explanations are inferred from repeated observations, or, perhaps, (*iii*) a combination of the two. On the other hand, the question of the scientist using ABM is:

> "How could the decentralized local interactions of heterogeneous autonomous agents generate the given regularity?" [53]

Epstein proposes to use the term *generative* to characterize this approach. He also discusses the epistemological soundness of the proposed approach [53]. A full discussion of the subject, even if fascinating, is beyond the scope of this

work. For our purposes, it is sufficient to say that in the generative method, demonstration obtained with ABM is taken as a *necessary condition*. In other words, if the agent-based model did not generate the desired phenomenon, then the modeled interactions do not explain the phenomenon itself. However, if the modeled interactions generate the expected emergent behavior, then they are only a *candidate explanation*. If more than one candidate explanations exist, additional care should be exercised to determine which explanation is the most tenable, typically considering the plausibility of the explanation and resorting to the same ideas used when proposing a hypothesis in inductive science, e.g., Ockham's razor.

Another important consideration regarding the relations between agent-based models and the propositions we can derive from their execution shall be made. Let M be an agent-based model and let v be a tuple of parameters that are suitable for M to be run and let $\mathbb{V}$ be the set of the suitable tuples. Example of such parameters can be the number of units to be simulated, positions of objects in the simulation or a relevant parameter determining the behavior of some agents. Then we indicate with $M(v)$ the result of running model M with starting parameters v. Moreover, let $P(x)$ be a proposition on the result of the simulation, so that $P(M(v))$ is either true or false and let $P(M(\hat{v}))$ be true. The question is which components of $\hat{v}$ can be varied and how much without making P false. This kind of problems arises quite often in different forms: (*i*) does a given phenomenon arise only for populations above/below a given threshold?; (*ii*) in which range of a given parameter α a certain property emerges?; (*iii*) before how much time does it emerge?; (*iv*) after how much time does it disappear, if it does?

Some instances of the problem are essentially non-existent in equation-based modeling. A lot of research has been made on solving differential equations depending on some initial parameters, and, excluding problems inherent with the complexity of the model itself, are well studied problems. Similarly, it may be easier to determine if a given property is present *eventually*, both in the case when the population goes to infinity or when time goes to infin-

ity. However, other problems are harder with equations: e.g., studying finite but non-statical populations or studying transient properties that appear but eventually disappear.

The key difference is that, for an agent-based model, the execution constitutes the "solution", and, as a consequence, the modeler can easily inspect the whole dynamic history of the system. This property of ABM is especially relevant in the context of social sciences, where many interesting processes are known to reach equilibrium only after disproportionately large amounts of time [14].

Although studying the system at the equilibrium remains an interesting problem, real societies are generally in the turbulent step before the equilibrium is reached and, moreover, perturbing factors are likely to emerge after a while, so that analysis cannot consider too large period of times without modeling the insurgence of such factors.

Another problem is that some concepts are very hard to consider in equation based modeling altogether. For example, "rational behavior" or any other kind of decision process that results in non-continuous variations of the agent externally perceivable state are very hard to model. Moreover, the notion of environment and position also increases the number of variables to consider and, potentially, introduces constraints on the problem. Eventually, modeling heterogeneous units with equations is a generally hard problem, because "individuality" is inherently hard to model.

All these factors together define an ample category of problems where ABM is not necessarily superior to equation-based modeling, but certainly is easier to develop.

4.2 Structuring Agent-based Models

In this Section we describe the main features of an agent-based model. A generic agent-based model has three main elements: (*a*) a set of agents, with their individual *attributes* (from now on referred to as *state*) and *behavior*;

(*b*) a set of agent relationships and interactions, that is essentially defined by a *topology*; and (*c*) the agent environment.

As the agents described in Chapter 3, agents in ABM have the capability to act autonomously. However, the notion of autonomy in ABM is significantly different to that considered in Chapter 3. In fact, there is a distinction between being autonomous and simulating an autonomous being; typically simulation have rules concerning time-flow, for example, that impose constraints on the agents that are no longer completely autonomous. Drogul [49] arrives to the conclusion that agent-based modeling is not agent-based programming in the sense discussed in Chapter 3.

Regardless of their autonomy level, agents can modify the environment and communicate with the other agent. An agent is essentially self-contained and has a boundary that clearly indicates what is part of the agent and what is not. Moreover, any change to an agent internal state is mediated by the agent itself.

Features often present in ABM are *adaptivity*, *goal-directedness* and *heterogeneity*. Adaptivity is essentially the capacity of the agent to change its own behavior according to external stimuli. A goal-directed agent may seem similar to an adaptive agent, since (*i*) it evaluates its performance according to given criteria, and (*ii*) it modifies its behavior to meet the goals. However, the difference is that it could only be able to vary some parameters and it does not necessarily have the ability (*i*) to reason about the rules governing its behavior and (*ii*) to possibly change them. On the other hand, being adaptive does not mean having goals to pursue. An agent can also be both adaptive and goal directed. As for heterogeneity, in many simulations, agent have different behaviors, different goals, different parameters and abilities. They are, in fact, different individuals.

An important design principle in ABM is that information should be local to the individual agents. There are no central authorities or repositories of knowledge. Moreover, the agents interact only with their neighbors according to the underlying topology.

Some ABMs imply a grid topology or other regular structures that clearly define the neighborhood. Other simulations use a more general network topology to determine the neighborhood relationships. Eventually, some of the most complex models have a virtual "real" environment determining the topology. The agents are free to roam in the environment and position their relative position determines the notion of neighborhood.

An example of such complex model is the Artificial Anasazi project [45]. The project goal is to reproduce spatial and demographic features of the Anasazi population from about A.D. 800 to 1300. In the model, the agents represented individual households, and the environment was reconstructed from paleoenvironmental records.

The environment is an important element in agent-based simulations, considering that it provides not only the conditions under which an agent exists, but it also provides, as *communication environment*, principles, processes and structures that enable information exchange among agents. When agents interact in a coordinate manner, it becomes a *social environment* [147].

As for the topology, different models have radically different notions of environment. An environment typically bidirectionally interacts with the agents and is essentially stateful. Environments range from simple means of communication, like, for example, that of ant colony simulations, or have a more crucial role, like the already mentioned Anasazi model.

In fact, we are mostly interested in ABM for social networks and, then, a few propositions are in order. The first is that a terminological ambiguity arises between the agent relationships mentioned in the (b) item at the beginning of the Section, that we call *system-level relationships* henceforth, and the actual relationships that are in the social network (*network-level relationships* or *links*). Although agents that are linked in the network do also have a system level relationship (i.e., being linked in the network), the opposite does not hold.

The second issue to discuss is the nature of the environment topology, which is either (i) a fully-connected graph where each agent can communicate

with any other agent, as is the case, for example, of network formation models, or (*ii*) the social network itself, that determines the communication channels among the agents.

As a consequence, even though proper agent-based models should allow only "local" (according to the environment, i.e., neighborhood relation according to the topology), for many interesting simulations it is required to allow direct communication among non-neighboring agents. Simulations involving weak links [84] are a typical example, in that weak link formation is not easily simulated with only local interactions.

Eventually, we think that from the agent-oriented perspective, the true environment for any social network simulation is a representation of the network itself, because agents exist only as nodes situated the network; moreover, for most processes the network is also the vehicle for information spreading and, eventually, the social environment where agents meet.

4.3 Agent-Based Modeling for Social Simulations

Even simple models can exhibit complex behavior patterns and can provide valuable information about the dynamics of the real-world system they simulate. In addition, models are often important for the unanticipated behaviors they allow to emerge.

Agent-Based Social Simulation (ABSS) [120] is an application field that has rapidly grown over the past few years as a specialized instantiation of ABM. With no loss of generality, we can say that ABSS is about using models of societies of agents, transferring such models to running software, observing the behaviors of the realized societies, and translating, if possible, such observations into quantitative data meant to support statistical analysis.

ABSS is well-founded on an epistemological level by the assumption that synthetic data enables the construction of theories that can be used to tackle real problems. ABSS does not simply try to reproduce phenomena that are

met in the real world; it also creates virtual scenarios that can give relevant understanding of related, yet possibly not observable, real world situations.

From a theoretical point of view, ABSS relies on the integration of several well known and appreciated theories, e.g., the theories of dynamic systems, of cellular automata, and obviously the theories of agents and multi-agent systems. ABSS uses such theories to formally characterize how diverse concepts used to describe social phenomena, e.g., individuals, groups and coalitions, interact and it enables the formal study of emergent phenomena.

Unlike rational decision theory, which is still often used to formally characterize social phenomena, ABSS is based on the following assumptions on agents:

- Heterogeneity: they are all different behaviorally;
- Complexity: they all have internal knowledge and regulatory mechanisms with varying degrees of complexity;
- Adaptability: they are capable of learning on the basis of external events;
- Flexibility: they are capable of a diversified response to variations in the external conditions, and they are capable of desisting from useless and/or (self-)destructive behavior;
- Versatility: they possess a strategy and complex rules that enhance the agents' efficiency and multipurpose nature.

From a methodological standpoint, ABSS is an exploratory-experimental technique based on a computational approach that aims at: (*i*) providing a rapid and efficient demonstration of the properties and performances of an artificial system; (*ii*) displaying in silico the effects of an artificial world including dynamic manipulation of the system's characteristics and the recording of the effects of such changes.

4.4 ABMs of Social Networks: state of the art

In this Section we review in a brief and yet formal way some agent-based models relative to social networks; we selected the models because of their

close relation to some of the most important aspects of ABM we discussed in this Chapter and we develop in our approach, presented in Chapter 5

The first model deals with social network formation using a game theoretic approach. The original analysis was performed analytically computing the stable states of the system [99]. The model has been later formulated as an ABM [95] and some additional facts were discovered that completed the study of the model.

The second model considered describes a bi-modal network of developers and open source projects. The most interesting aspect is that the agent-behavior has been specified directly using really observed micro-level interactions. Equation based modeling, for example, would have required additional work in singling out the general laws from the raw data.

Then, we review a seminal work that used social networks to improve the performance of a then popular P2P network. In order to assess the effectiveness of the proposed method, the authors performed extensive simulations. The main link with our own work is the usage of simulations to experimentally evaluate the design of P2P systems without needing a large number of users to adopt the proposed systems (Chapter 7).

Subsequently, we briefly considers the issue of trust management in social networks using agent-based modeling. Finally, we reconnect to the problem of referral systems we already discussed in Section 3.4 in the engineering context of multi-agent systems. In this case, however, agents are not only a software engineering technique, but a tool to assess the effectiveness of algorithms in various contexts.

Utility based model for network formation

In Section 4.1 we discussed at length of the different approach and results that can be expected using either ABM or equation-based modeling. Here we show how when the two approaches are combined, interesting results can be achieved.

A clear example of the stress on dynamic properties of agent-based simulations with respect to traditional mathematical analysis can be found in the contributions (both theoretical and practical) that Hummon [95] made in the field of modeling social network formation using game theoretic techniques.

In Section 2.3 we reviewed some social network formation models that evolved a social network using random processes. An alternative approach is that of specifying a *utility function* that should be maximized during the network formation process. Here the term *utility* is used in the sense of game theory: the process is essentially formulated as a game where the nodes are the players and their moves are (*i*) the creation or (*ii*) the severing of any edge.

More formally, let V be a set of players and let $G = (V, E)$ be a network of players. These players can establish a connection with any other player, and we say they are *immediately connected*. E is the set of such immediate connections. Connections are symmetric. A player cannot form a connection with himself.

If there is a path of length k between i and j, then $\left(A^k\right)_{ij} > 0$; if for any $k > 1$ $\left(A^k\right)_{ij} > 0$, then i and j are *indirectly connected*.

A network $G = (V, E)$ is *efficient* relative to the profile of utility functions $(u_i, \ldots, u_n)$ if:

$$U(G) = \sum_i u_i(G) \geqslant \sum_i u_i(G') = U(G') \tag{4.1}$$

for any $G' = (V, E')$. Efficiency is essentially an optimality property.

Let $b : \{1, \ldots, n-1\} \to \mathbf{R}$ be the net benefit that a player receives from (indirect) connections as a function of distance between the players. Let l_{ij} be the shortest path length between the players i and j. Let k_i be the degree of player i, i.e., the number of his connections, and let c be a constant. The *distance-based utility model* is one in which a player's utility is:

$$u_i = \sum_{i \neq j} b(l_{ij}) - k_i c \qquad \text{if there is a path between } i \text{ and } j \tag{4.2}$$

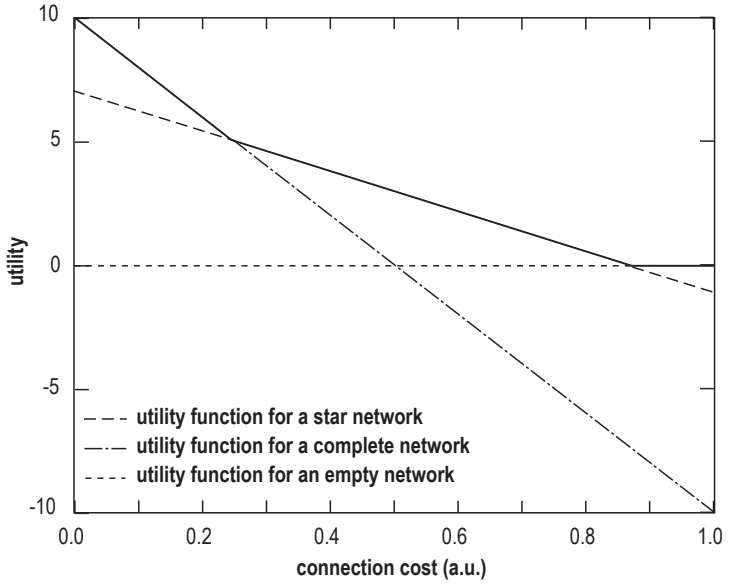

Figure 4.1: Optimality transition of utility functions for three networks shapes (5 nodes) (see Theorem 1). The thick solid line represents the best achievable utility value.

We also require the following conditions to be true:

$$b(k) > b(k+1) > 0 \quad \forall k \in \{1, \dots\} \tag{4.3a}$$

$$c > 0 \tag{4.3b}$$

$$b \in C^2 \tag{4.3c}$$

This model was originally proposed by Jackson and Wolinsky [99] to investigate the formation of social and economic networks from the economical point of view of the "utility" that people receive from being connected with other people. Their main results are summarized in the following theorem and are visually presented in Figure 4.1.

Theorem 1 (Jackson-Wolinsky)**.** *The unique efficient network structure in the distance-based utility model is:*

1. *the complete network if* $b(2) < b(1) - c$
2. *a star encompassing all nodes if* $b(1) - b(2) < c < b(1) + \frac{n-2}{2}b(2)$
3. *the empty network if* $b(1) + \frac{n-2}{2}b(2) < c$.

Theorem 1 determines the optimal network for any value of the cost factor c. However, the question whether a society is actually able to converge in such network structures remain unanswered. On the other hand, Hummon [95] created a multi-agent simulation based on the same model model. The formation of a link requires the consent of both parties involved, but the severance can be done unilaterally. Essentially, this means that any tie must not decrease the utility of any involved actor. Ideally, the network would evolve into an optimal shape according to the specified value of c.

Since Hummon was able to study the system dynamically, he was able to observe that under a wide range of parameter values, group structures first form star-like structures and then they may transform to other more complete structures. This is coherent with observation in human groups.

However, an even more important result was the discovery of some equilibria not present in the theoretical analysis of Jackson and Wolinksy. This discrepancy is studied and explained by Doreian and by Xie [47, 48, 196]. Such equilibria are not Pareto efficient, thus are not considered by Theorem 1, nonetheless the agents have no way to move towards another state. The issue is that agents can only take greedy decisions because of bounded rationality [170]: in order to evolve the system towards a more efficient equilibrium the agents would require global knowledge of the system and the ability to plan the evolution globally.

This result is especially relevant because while Theorem 1 identifies the optimal network configurations, the agent-based simulation tries to achieve an optimal configuration from the bottom up. The new equilibria found by Hummon indicate that without a global plan, as is the case in real societies, the optimal configurations are not always achieved.

A social network of open source developers

Another archetypal example of multi-agent simulation used to understand social networks can be found in the work of Madey et al. [124]. Here the social network taken into account is the collaboration network of open source and free software developers on Sourceforge[1]. The considered social network consists of a collection of global virtual self-organizing teams of software developers. Two developers are linked if they are members of the same project; this means that each project yields a group.

The empirical collected data spans over more than 24 months of snapshot data of developers and project statistics between January 2001 and May 2003. The number of projects considered is over 120.000. Each developer and project has a unique id. Developers and projects also have a degree: for a developer, it is the number of project he contributed to, for projects it is the number of contributors. It appears that both degrees follow power-law distributions. Growth of projects and the number of participating developers has been extracted in order to parameterize simulation with reasonable values.

Each developer has been modeled with an agent. The agents can create, join or abandon a project each day or continue their current collaborations. The probability for each of these actions has been calculated from the original corpus of data. The original simulation has been performed using the Swarm toolkit [129] (a brief review of Swarm and other agent-based simulation systems used in the field of social network systems is given later in this chapter). The simulation has been run with different behavioral patterns of the developers and the model was improved in multiple steps in order to better approximate the real data, until the model was actually able to simulate the real process.

[1] `www.sourceforge.net`

P2P Systems using SN to improve performance

A completely different agent-based simulation using social networks is described by Upadrashta et al. [183]. The authors notice how the then-popular Gnutella peer-to-peer system is inefficient with regards to the number of messages exchanged between nodes to perform queries. Their claim is that the phenomenon is due to the fact that the nodes do not remember who has interest similar to theirs and consequently is more likely to host files of interest.

In order to prove their claim, they constructed an agent-based simulation, where each agent stands for a network node. Each agent constructs a model of his social network based on the answers it received from previous queries. The constructed social network is then employed to preferentially query agents, which are believed to possess the required file with a higher probability. The simulation proved that their claim was correct, since social network directed query have been observed to be much more efficient in terms of exchanged messages.

This example shows how agent-based simulations may be of interest to unveil or to better substantiate laws that rule the behavior of complex social networks.

Trust network simulation

The concepts of reputation and trust are hot topics in multi-agent systems. Dealing with open systems where agents come from heterogeneous sources, it is important to develop models of trust in a similar way to what people actually do in real life. For the general problem of trust and reputation in distributed settings we refer to the work of Huynh, Jennings and Shadbolt [96]. Instead, we focus on the approach described in [163]. The authors combine different dimensions of reputation. In this Section we are mostly concerned with the individual dimension and the social dimension. The individual dimension of reputation is related to the actual interactions between two agents. However, in open system it is very easy for two agents to have never interacted before. In

this case, the only mean to compute the reputation is using each agent social network. Perhaps agent a does not know anything about agent c, but knows agent b who does. This way the social network is used to attribute reputation values to unknown agents. An experimentation tool called Regret based on the aforementioned ideas is considered at the end of the paper.

Referral system simulations

In Section 3.4 we discussed how multi-agent systems have been used to successfully implement referral systems. The distributed nature of MAS is well-suited to search for expertise in open and dynamic contexts. A related question is the choice of algorithms to propagate the query in order to receive an appropriate answer, quickly and without flooding the network.

Agents can also be helpful doubling their role as simulation entities and components of a multi-agent system. In other words, it is possible to create a simulations were agents model the behavior of software agents. Zhang and Ackerman [204] created such model in order to investigate the performance of search algorithms in the referral context.

Among the reviewed algorithms we find the classical Breadth First Search (BFS), as well as the "best-connected" heuristic (BCS) suggested by Adamic and Adar [2], and Yu and Singh algorithm [201] and described in Section 3.4.

The parameters taken into account were the success rate (very high with every algorithm), the number of agents "bothered" with each query, the length of the average query path and the labor distribution (that is to say the number of times each agent has been bothered). The simulation was performed both with and without "weak-ties". The results showed that in a corporate environment (the Enron body of emails was employed for the simulation) the BCS strategy was very successful.

4.5 Tools for Agent-based Modeling

While many agent-based models use ad hoc systems created on top of agent based platforms such as Jade, the cost is often considered prohibitive and dedicated toolkits are used. This Section presents the state of the art of agent-based modeling toolkits. Our own project is presented in Chapter 6.

The first reviewed toolkit is Swarm [129], which is also the oldest system chronologically. In Swarm the model consists of a collection of independent agents interacting via discrete events. A collection of agents plus a schedule constitutes a swarm. A swarm is also an agent and in this case the agent's behavior is determined by the emergent phenomena of the agents inside its warm. There is no limit to the level of nesting. Each swarm agent can also "own" swarms, that is to say that it can make decision based on simulations he performs. In order to set up the simulation, the researcher must write the behavior of the single agents, i.e., write Objective-C classes describing the agent, with methods describing the available actions. The actual agents are the instances of the classes. Moreover, Swarm offers "probe" facilities in order to observe each aspect of the computation. Swarm also has a Java layer allowing the use of Java to write the agents.

Another important simulation toolkit is MASON [121]. Mason is described by the authors as a fast, easily extensible, discrete-event multi-agent simulation toolkit in Java, designed to serve as the basis for a wide range of multi-agent simulation tasks ranging from swarm robotics to machine learning to social complexity environments. A comparison among Mason, Swarm and RePast is also available in their paper [121].It is worth noting that MASON has a specific module to perform social network analysis.

NetLogo [180] is a multi-agent programming language and modeling environment for simulating natural and social phenomena. NetLogo can be described as Logo with agents, which are akin to the famous Logo turtle; the programmer can specify the behavior of the turtles and there are special observer agents, which can both observe emergent behaviors and issue commands

to the turtles. NetLogo comes with a wide selection of pre-built modules.

NetLogo focuses on the 2-dimensional grid topology, which is useful for many simulations, but not particularly relevant in our domain. While it is possible to simulate a different topology, many of the powerful features of the toolkit are created with the 2D grid topology in mind.

Another problem we have found with NetLogo is that the visualization of models tends to leak in the rest of the model. Moreover, the visualization features are closely related to the idea of moving turtles on the grid, which feels awkward in case of social networks. For example in the demonstrative models about social networks provided with NetLogo, node-turtles "walk" on the grid while the simulation progresses just to present a visualization of the social network. On the other hand, if visualization is required, social network analysis packages (e.g., NetworkX [87]) offer easier to use, more efficient and more scalable alternatives.

Eventually, although we agree that agent-based simulations benefit from a custom language, we feel that the NetLogo language lacks: (*i*) many interesting features found in general purpose languages that may be useful to implement algorithms and (*ii*) useful libraries (e.g., proper social network analysis toolkits, statistical toolkits). In fact, using network analytic or statistical techniques is hardly uncommon inside a model, and many of these algorithms may well be as complicated to implement as the simulation itself, especially considering that NetLogo is also aimed at non-programmers. An extension to NetLogo [178] allows to communicate with an R[2] process to perform analysis, but we think that easier solutions can be provided using just one language.

RePast [144] is another agent-based simulator. RePast was initially a library of Java classes interacting and simplifying Swarm, and now has grown to an independent toolkit, with its own class library, visual modeler and visualizers. RePast agents can be written in Java, C# or a Python-like language. RePast is especially strong in its support for networks (and specifically social networks).

[2] `http://www.r-project.org/`

Currently, RePast evolved in RePast Symphony [145]. RePast Symphony models can be written in Java, which remains the main implementation language, using flowcharts or in Groovy. Moreover, there is a NetLogo inspired language called ReLogo, that is essentially a internal domain specific language implemented in Groovy. For a discussion of domain specific languages, we refer to Section 6.2. ReLogo is created with the 2D Grid in mind that was made popular by NetLogo and according to Lytinen [122], it does not interoperate well with the rest of the RePast framework.

To conclude, Mason, RePast, and Swarm are mostly libraries or frameworks for general purpose languages (Java for Mason and RePast and Objective-C or Java for Swarm) and this is a clear advantage when interfacing with external libraries (e.g., the statistics or network analytic packages we already mentioned), to the point that RePast, for example, already provides integration with Jung, a popular Java library for network and graph analysis. However, the integration layer is, in our opinion, very basic, although user friendly.

All considered, those systems are relatively difficult to learn because they have rich and elaborate API [157]. Simulations are software projects typically split in more files and require quite a lot of boilerplate code, since they are written in Java (or Objective-C). We feel that although such complexity is needed to cope with the general case of agent-based simulations, restricting the domain greatly helps in keeping things relatively simple and allows for simpler tools. Moreover, the integration with different libraries, when not already provided, may be not as easy for people without a strong programming background, especially because correctly hooking into a complex framework is not particularly easy task.

This considerations suggested us the development of a new tool for ABM that we named PyNetSYM and that is described in Chapter 6.

Chapter 5

Social Network Simulations as Multi-Agent Systems

In the last sixty years of research, several models have been proposed to explain (*i*) the formation and (*ii*) the evolution of networks, or, more in general, (*iii*) various kinds of processes over networks.

Such models have been developed and studied by researchers coming from many different areas, such as computer science, economics, natural sciences, meteorology, medicine, pure or applied mathematics, sociology and statistics. As a consequence, a lot of material exists on the subject and it is beyond the scope of this work to review and analyze it thoroughly; we refer to Jackson for the economic and game-theoretic point of view [98], and to Snijders for a complete review of the state of art of statistical models [173]; in Section 2.3, we focus on the methods and models originated in computer, natural and physical sciences.

Several agent-based models have also been successfully developed in order to study specific problems that, because of either (*i*) the complexity of the agent to agent interactions, or (*ii*) the richness of the underlying environment, were relatively impervious to analysis using traditional techniques [8, 14, 15, 35, 45, 56, 62, 92, 97, 109, 131, 146].

However, because of the specialization required for the problems, most of the agent-bases models are not general. On the other hand, many of the traditional models focus on elementary interactions that are often part of several different processes, and, consequently, the effects of these elementary interactions have been thoroughly studied.

We believe that such interactions should be introduced as basic building blocks even for agent-based models, enriched with more "agent-ness" when it is the case. However, several assumptions that are perfectly legitimate in a stochastic process are not well rendered in an agent-based model. Moreover, many considerations that we derive from the conversion from traditional stochastic to agent-based models are similar to the ones that should be made by implementing the stochastic model in a generic non agent-based concurrent environment.

In Sections 5.1 and 5.2 we focus on the description of the issues we found in interpreting the traditional models as agent-based ones; we also present a novel meta-model that we singled out from the study of several stochastic models; our meta-model can be used to (*i*) easily convert stochastic models to agent-based, and (*ii*) easily devise new fully agent-based models with minimal effort. In Section 5.3 we show some interesting experimental results on our approach to the transition. Eventually, in Section 5.4 we fit some popular models meta-model as examples.

5.1 Towards Agent-based Models

The models presented in Section 2.3 and many other important network models, such as SIR/SI epidemic models [150], are successful in modeling the desired macro-level phenomenon using only micro-level interactions. From the epistemological point of view, they are kin to agent-based models; however, they are formulated as stochastic Markov processes, and make very strong assumptions, such as (*i*) uniformity of node behavior, (*ii*) lack of memory, and, consequently, (*iii*) also uniformity of behavior in time.

As for uniformity, these models assume that all the nodes behave according to the same probability distribution (or to a small set of probability distributions, that entirely encompass the node behavior). The problem of lack of memory is that what occurs at each step s_i of the simulation essentially depends only on the state of the network at step s_{i-1}, disregarding how the network arrived in that state. Moreover, most of the studied properties are derived mathematically at the thermodynamic limit, even if the real populations are typically too small to be discussed by using thermodynamic arguments.

Part of the issue is also that the processes that drive real social networks are likely to be far more complex than those employed in the models, and we do not really want to assume that the premises we adopted for the sake of mathematical tractability are actually true in real (and finite-sized) populations. However, some of the individual micro-level interactions have been observed in human societies (preferential attachment, transitive linking) and are a realistic element of human interactions.

On the other hand, agent-based models can express the variety of human behavior and heterogeneity more easily. Moreover, agent-based models naturally deal with finite-sized population, and, in fact, can *only* deal with finite-sized populations, so that properties at thermodynamic limits may only be approximated. Eventually, as we discussed in Section 4.1, agent-based models can be studied during the whole system evolution, and not only at the equilibrium.

Unfortunately, each agent-based model tends to be an ad-hoc construction specifically tailored for a research problem, which yields advantages and disadvantages. As a consequence, no agent-based model has been extensively studied as the traditional models and, because of its nature, is probably impervious to general studies. Moreover, a thorough study of an agent-based model is not even meaningful outside the specific phenomenon it was tailor-made to model. In this sense, many of the traditional models are, in essence, far more general and could be used in many different fields.

All considered, we think it is interesting to devise procedures to fit the well-

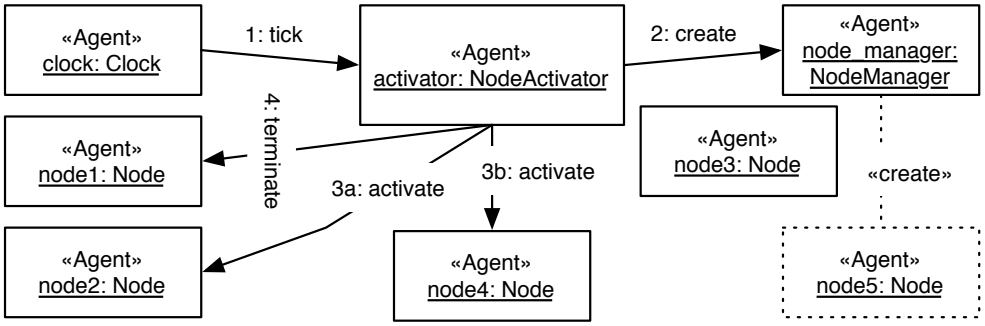

Figure 5.1: Interaction diagram of the main agents in the meta-model.

studied stochastic models in an agent-based scenario. Consequently it becomes possible to start from well studied building blocks and gradually remove some or all the assumptions that were originally made, and, eventually, adding novel features or combining more basic interactions to form richer processes.

A meta-model for agent-based models of social networks

In order to simplify (*i*) the creation of new agent-based models and (*ii*) the adaptation of traditional models into agent-based ones, we designed a conceptual framework that separates the various concerns and allows to write simulations with few lines of code [25, 70].

Analyzing the traditional models described in Section 2.3 as well as some other non-generative network processes, such as the infection diffusion models described by Pastor-Satorras and Vespignani [150], we singled out a meta-model that is suitable to implement said models as agent-based models. The meta-model also allows for features typical of agent-based models to be introduced gradually and to added to the agent-based variants of the traditional models.

Since in many simulations the nodes do not have a pro-active, goal-directed behavior, but, instead, perform actions when required by the model, we pro-

vide the meta-model with a Clock and a special agent called Activator. The Clock beats the time, which is discrete. At each step, the Activator selects (*i*) which nodes to activate, and (*ii*) decides whether nodes shall be destroyed, or (*iii*) created, negotiating the eventuality with the NodeManager. The nodes execute actions after receiving the activation message. However, if the model is fully agent-oriented, they can also perform actions autonomously, without waiting for activation. The nodes can also leave the simulation, require the creation of other agents, and send messages to the other nodes. The general structure of the execution phase is presented in Figure 5.1.

In essence, a simulation is fully described providing the specifications of: (*i*) the selection process of the groups of nodes to create, activate or destroy, which is performed by the Activator agent; (*ii*) the behavior of the nodes themselves Notice that the general structure does not change significantly even when introducing some agent-ness in the simulations, e.g., introducing goal-directed behavior.

The problem of time in distributed systems

In order to implement the stochastic models as agent-based, care must be taken, as it occurs in any generically concurrent implementation. In fact, in order to properly study inherently large scale social phenomena such as social networking sites or, more generally, social media, it is extremely desirable to distribute the computations, and to move towards massively concurrent implementations. As a consequence, the considerations we draw in this Chapter apply to both agent-based computing and, more generally, to high-performance distributed computing.

In fact, the notion and representation of time is a very crucial difference between stochastic models and agent-based models. Time in stochastic models is represented as the flow of discrete, strictly sequential and distinct steps, and during each step all the events are instantaneous and similarly the consequences of their actions are immediate. With "event" we indicate anything that is relevant for the model, such as taking a decision in a given world state,

or establishing a new link. If the model has its notion of time, model-time advancement is an "event" itself; in fact, in the case of consequences that occur at a later time, are events both (*i*) the actual occurrence and (*ii*) the fact that in a later time the consequence is bound to occur.

On the other hand, time is a very delicate matter in agent-based systems and in distributed systems in general. In fact, in a distributed system it is completely impossible to define a *unique (linear) global time* [3]. Each agent in the system has only a local time and can order the events which occur locally or, in other words, its own state changes. Such changes can be endogenously generated (e.g., because it is advancing in the execution of its program) or exogenous. In the latter case, the change can be only triggered by the reception of a message. However, the local orderings are not completely unrelated. In fact, the causal relationships between events occurring at different agents produce a partial order of events, the *activation order*.

It is worth noting that the lack of a natural total ordering among the events does not imply that a system cannot enforce a specific total ordering. Consider for example Cook's hardware modification machine [52] or, more simply, a system where a "global synchronizer" controls each and every step of the other agents that consequently have to wait for its permission to perform any action. It is not hard to see that such a system relates every two events with a causal relation, so the activation order is total. However, the price is that such an entirely synchronous system does not present the benefits of a distributed systems.

5.2 Concurrency Issues in ABM

In Section 5.1 we discussed the problem of time in distributed systems. In this section we assume a more practical point of view on the matter and show to what extent the theoretical problem concretely affects a simulation. Eventually, we discuss the BA model in a concurrent context and present some results from an experimental evaluation.

The models described in Section 2.3 were intended for computer simulation from the start, albeit they were not agent-based. In fact, some of them have been only studied by means of simulation (BPA [112] and TL [43]), while others were subject to analytic study only at a second time (WS [187]). Eventually, some properties were never formally proved, such as the expected clustering coefficient of networks generated with the BA model.

The models were implemented using some imperative or, in some instances, object oriented language and did not assume to be run in heavily concurrent scenarios. In such languages, there is a causal relationship among all the executed program lines and consequently the "activation order" is total.

On the other hand, agent-based languages or frameworks allow for both asynchronous and synchronous communication. If synchronous communication is not provided out of the box, it can be easily simulated using asynchronous communication. With synchronous communication, both the sender and the receiver must be ready to communicate before a message can be sent, or, in practice, the sender waits that the receiver actually receives the communication. In other words, the "send" operation is blocking. With asynchronous communication a message is simply sent to the receiver without further attention from the sender, that proceeds with its activities.

First, we have to consider the concurrency environment where the simulation is run. The simplest case is when a single executor is available (e.g., single-core single computer) and when concurrency is achieved using either preemptive or cooperative-multitasking. From our point of view, it matters little whether the concurrent execution primitive is an operating system process, an operating system thread or a lightweight in-process thread.

Another basic issue is whether a message handler is interruptible or not. The distinction is especially relevant if only one executor is available: in this case, uninterruptible handlers guarantee that the state of the world does not change during the execution of the handler because nothing external to the handler can occur before the handler terminated. The *Actor Model*, for example, assumes that the handlers are not interruptible [3]. Uninterruptible

handlers allow for easier reasoning on the formal properties of the system and are not considered a serious parallelism-limiting issue, because, in general, the handlers can be kept brief. In a cooperative lightweight threading scenario it is very natural to implement uninterruptible handlers: the agent simply relinquishes control when it finishes to process a message it has previously received. However, care should be taken in the case of preemptive multitasking.

A more general case is when multiple executors are available: as a consequence, a higher amount of concurrency is available as well. Moreover, the multiple executors could be distributed on different machines, potentially in different physical locations. This may be the case for very large scale simulations. With multiple executors, the properties deriving from cooperative multitasking are somewhat weaker, because in the case of multiple machines more than one handler is executed at a time.

In this situation, every time an asynchronous message is sent, the successive action performed by the agent may be executed in a world state where such message has been: (*i*) received and processed (either entirely or partially), (*ii*) received but not processed or, potentially, (*iii*) not even received. There are two main implications:

- the message sender cannot predict the state of the world at the moment in which the recipient is going to process the message;
- the message sender cannot assume that the receiver actually received the message it just sent, nor has guarantees when it will occur.

From our point of view, it means that when we translate a sequential model into an agent-based one, any "decision" that follows an asynchronous send operation can find the world in a different state from that of the corresponding sequential model.

Barabàsi-Albert model: a case study

In order to clarify this issue, we make some considerations regarding the different notion of time implicit in the Barabàsi-Albert Model (BA Model) [16], and the one in an agent-based variant of the same model. We opted for the

Barabàsi-Albert (BA) model for its simplicity and because it yields networks with power-law degree distribution with exponent $\gamma = 3$, which is both a very distinctive feature and a feature that is easy to measure. Since the only primitive interaction operating in a BA model is that of preferential attachment and, since that process yields networks with said degree distributions, we can measure whether concurrency is interfering with the preferential attachment, once the model is executed in a concurrent environment.

Although the model is really easy to understand, we point out some details that are not important in the "traditional" scenario, but that become extremely relevant in the concurrent setting. In the BA model, at each step:

- all the targets are chosen together, or at least, before that the network is modified (the two strategies are indistinguishable in the mathematical model), and

- when node t_{i+1} (i.e., the node created at time t_{i+1}) is created, node t_i has already been connected with all its target agents. The choices at step t_{i+1} according to the degree distribution including the new links created at step t_i.

In other words, let C_j^i be the time when node t_i chooses its j-th target and let M_j^i be the time when the system is actually updated so that i and its j-th target $i(j)$ are linked. In fact, there are also the times (a) S_j^i, when the message is actually sent, and (b) R_j^i when node $i(j)$ receives the message. The multiple causal relation $C_j^i < S_j^i < R_j^i < M_j^i$ holds. However, the Ss and the Rs times are not necessary for our analysis. Cs are read operations, Ms are write operations. In the original BA Model (n steps, m edges per node, henceforth indicated with BA(n, m)) the following relations are true:

```
1:  class Node(Agent):
2:      handler act():
3:          ts ← {}
4:          while |ts| < m :
5:              t ← pa(graph)
6:              if t ∉ ts :
7:                  ts ← ts ∪ {t}
8:          for t in ts :
9:              send(t, "link-to")
10:
11: class Activator(Agent):
12:     handler run():
13:         for s in 1 ... (n − m_0) :
14:             node ← create-node()
15:             send(node, "act")
```

Figure 5.2: Agent-Based Barabàsi-Albert Model.

$$C_j^i < C_k^i \quad \forall i \in \{1,\dots,n\},\ \text{if } j < k \tag{5.1}$$

$$M_j^i < M_k^i \quad \forall i \in \{1,\dots,n\},\ \text{if } j < k \tag{5.2}$$

$$C_j^i < M_j^i \quad \forall i \in \{1,\dots,n\},\ \forall j \in \{1,\dots,m\} \tag{5.3}$$

$$C_{j_1}^i < M_{j_2}^i \quad \forall i \in \{1,\dots,n\},\ \forall (j_1,j_2) \in \{1,\dots,m\}^2,\ j_1 \neq j_2 \tag{5.4}$$

$$M_{j_1}^i < C_{j_2}^{i+1} \quad \forall i \in \{1,\dots,n\},\ \forall (j_1,j_2) \in \{1,\dots,m\}^2,\ j_1 \neq j_2 \tag{5.5}$$

Equation 5.1 is trivially true. Considering that the model is meant sequentially executed in an imperative language, also Equation 5.2 is trivially true. Equation 5.3 expresses the causal relation between the choice of a node as the recipient of a link and the creation of that link. Equation 5.4 indicates that in a single step all the targets are chosen before any link is added to the network. Equation 5.5 states that before processing step $i + 1$, all the links decided in step i must have been created.

In Figure 5.2 we present how a program implementing the model could look like in an agent-based pseudo-language. The "pa" procedure on line 5 returns a node according to preferential attachment. In fact, depending on the semantics of the operations and from the guarantees made by the system,

some of the equation may not be true anymore.

For the sake of simplicity, suppose we are in the one-executor, uninterruptible handlers scenario. In this situation, once a handler is executed, no operations outside the handler occur. Nonetheless, even in this simple case, some of the equation can be violated. When a node processes the `"act"` message, it choses its targets according to preferential attachment and then it sends them the message. Since in this case the handlers are uninterruptible, we know that before any of these messages is processed, all of them have been sent. However, there is no reason why the `"link-to"` messages should be processed exactly in the same order in which the targets were chosen. This technically violates Equation 5.2; nonetheless, it has little impact on the generated network.

The problem is that, in general Equation 5.5 can be also be violated, because the Activator sends all the `"act"` messages together and when a node t processes its `"act"` message, there are no guarantees that the `"link-to"` messages sent by the previously created nodes have already been processed.

The consequences could be far more severe: in the worse case, we can assume that all the `"act"` messages are processed before any `"link-to"` message is. In other words, it could be that $C_*^k < M_*^j$ for all k and j. If every selection is made before the nodes get the links, each node selects its targets among the nodes created before it that still have the same degree and consequently the preferential attachment selection degenerates in a random selection. Barabasi et al. proved that the latter creates a completely different degree distribution altogether and, effectively, it would not be a BA process anymore [16].

With the implementation provided in Figure 5.2, Equation 5.3 and Equation 5.4 are always true, even with interruptible handlers. However, in some circumstances a different implementation could be more desirable. In Figure 5.3 we show such an alternative implementation. Suppose for example that we have multiple executors: a simulation system could use one thread to deliver the messages to the recipients (perhaps residing on a different machine) and another to actually run the agent. The preferential attachment selection

```
1: class Node(Agent):
2:     handler act():
3:         targets ← {}
4:         while |targets| < m :
5:             target ← pa(graph)
6:             if target ∉ targets :
7:                 targets ← targets ∪ {target}
8:                 send(target, "link-to")
```

Figure 5.3: Alternative implementation of BA nodes.

is pretty expensive with large networks and if all the choices are made before any message is delivered, CPU time is used less efficiently than immediately delivering each message, which is taken care of by another thread on another CPU/core, and proceed with the following choice.

However, in this scenario Equation 5.4 is clearly violated. On the other hand, the improved efficiency could outweigh the correctness violation. Intuitively, as long as not "too many links" are created before all the choices are made, the resulting network still has the desired properties. While it is not hard to prove that, with a reasonably fair scheduler, the extreme situations described in the case of the violation of Equation 5.5 are almost impossible, it is much harder:

- to prove that concurrency does not *slightly* modify the model behavior as in the case just described,
- if it does so, to estimate the introduced bias, and
- to give explicit conditions on the scheduling algorithm such that networks generated with the agent-based concurrent model are indistinguishable from networks created with the sequential generator.

However, such eventualities can have ill effects in the long terms on the quality of the results. As a consequence, we feel that there is a very strong need to investigate the effects of concurrency in traditional processes. Considering the difficulty of doing so analytically, we decided to uses experimental evaluations in controlled settings.

5.3 Experimental evaluation of Concurrency Issues in Agent-based Simulations

In the previous Section, we showed how concurrent agent-based code can violate some assumption implicitly made in a traditional model. Although it is possible to limit concurrency in a way that such violations do not occur, it would mean to renounce to many of the advantages offered by agent-based systems. Moreover, considering that we want to create fully agent-based models that only occasionally use some ideas from the traditional models, such limitations would be unacceptable. Eventually, enforcing some guarantees is simply impossible in a truly distributed "grid" or "cloud" environment.

In this Section we fully embrace the generative point of view of agent-based modeling and we experimentally evaluate the impact of the different concurrency environments using an agent-based model similar to the one presented in Figure 5.3, based on the meta-model described in Section 5.1. In environments with limited concurrency, the meta-model is completely equivalent to the one of Figure 5.2, however, increasing concurrency could introduce more visible biases. We repeated the experiments with several software frameworks that we built around the meta-model. The data presented here, comes from the simulation using PyNetSYM, whose design and implementation are the subjects of Chapter 6. When the tools are set to operate with the same concurrency semantics, no discernible differences can be found in the result.

Our first experiment is conducted in the first scenario described in Section 5.2, that is to say single-executor, uninterruptible handlers. In this scenario, Equation 5.5 is not true and a particularly biased scheduler could make the preferential attachment process degenerate into an uniform selection. In order to see whether the default schedulers are fair enough to avoid such eventualities, we experimentally compare the networks created using agent-based variant of the BA model with those created using a sequential implementation. Specifically, as the sequential implementation, we used a library function contained in the network analysis package NetworkX.

Table 5.1: Comparison between networks created according to agent-based and traditional Barabàsi-Albert models (denoted by the label "Sequential"). Several networks have been generated with different sizes (n) and number of starting edges per node (m). We compare the clustering coefficient C, the characteristic path length CPL and the γ exponent of a power-law fitting the degree distribution of networks generated with an agent-based simulator and using a sequential implementation of the BA model. We also report the values for the clustering coefficient and the CPL computed using the theoretical formulas derived at the thermodynamic limit (with the label "Analytical"). Unsurprisingly, the latter are not very accurate for the small networks considered here.

Conditions		Agent-Based			Sequential			Analytical	
n	m	C	CPL	γ	C	CPL	γ	C	CPL
1000	5	3.25e-02	3	2.7	4.21e-02	3	2.8	5.62e-03	3.6
1000	8	4.93e-02	2.7	2.8	5.48e-02	2.7	2.8	5.62e-03	3.6
1000	11	5.89e-02	2.6	2.9	6.16e-02	2.6	2.8	5.62e-03	3.6
1000	20	9.00e-02	2.2	2.9	9.79e-02	2.2	2.9	5.62e-03	3.6
5000	5	8.37e-03	3.6	2.9	1.06e-02	3.5	2.9	1.68e-03	4
5000	8	1.36e-02	3.1	2.8	1.65e-02	3	2.9	1.68e-03	4
5000	11	1.82e-02	2.9	2.9	2.02e-02	2.9	2.9	1.68e-03	4
5000	20	2.86e-02	2.7	2.9	3.00e-02	2.7	2.9	1.68e-03	4
10000	5	5.66e-03	3.7	2.9	6.46e-03	3.7	2.9	1.00e-03	4.1
10000	8	8.17e-03	3.3	3	9.71e-03	3.3	2.8	1.00e-03	4.1
10000	11	1.08e-02	3	2.9	1.17e-02	3	2.9	1.00e-03	4.1
10000	20	1.69e-02	2.8	2.9	1.76e-02	2.8	2.9	1.00e-03	4.1
50000	5	1.30e-03	4.3	3	1.92e-03	4.1	2.9	2.99e-04	4.5
50000	8	2.16e-03	3.7	3	2.77e-03	3.7	2.9	2.99e-04	4.5
50000	11	2.77e-03	3.5	3	3.33e-03	3.4	2.9	2.99e-04	4.5
50000	20	4.78e-03	3	3	5.02e-03	3	2.9	2.99e-04	4.5

In Table 5.1 we report some results: we generated networks of different sizes (n) and with different starting number of edges per node (m) with both tools and found out that the clustering coefficient C and the characteristic path length CPL are significantly close one to each other (typically they differ by less than 10%). We also compared the values with those predicted by the purely analytic methods described in Section 2.3: such methods assume a network of infinite size and are very rough estimators of what occurs in large but not huge networks, and, in fact, underestimate the clustering coefficient of an order of magnitude and also overestimate the average path length.

The most defining feature of the Barabàsi-Albert model is that it yields networks whose degree distribution is a power-law with exponent $\gamma = 3$, even for such small networks. Consequently, we decided to test the degree distributions generated with our tool and with NetworkX using the techniques discussed by Clauset [37]; their γ coefficients are, as predicted by analytical means, close to 3. In Figure 5.4 we report the degree distributions $f(k)$ (upper panel) and the complementary cumulative distribution functions $S(k) = 1 - \sum f(k)$ (lower panel) of the two networks. The agent-based variant do not seem to introduce relevant biases in the degree-distribution of the generated networks.

Another interesting question is whether the possibility to interrupt the message handlers changes the model behavior. The possibility to interrupt handlers means that the actions of multiple nodes may be mingled and generally could make more extreme the effects of concurrency. In fact, we studied the results of simulations where the agents had and had not interruptible handlers respectively, and we found out that, the generated networks are mostly indistinguishable. In Figure 5.5 we plotted the degree distributions and the CCDFs. The two degree distributions are extremely similar and both have the theoretically predicted exponent, evaluated with the method described in [37]. Apparently, interrupting the handlers and consequently allowing the "choices" in a potentially different network state does not seem to affect the outcomes: the scheduler is appropriately designed.

5.4 Agent-based adaptation of traditional models

In this Section, we show how the traditional models can be adapted to agent-based models using the meta-model we just introduced. It is worth noting that all the generation models have explicitly or implicitly a notion of "time", in the sense that in the BA, Transitive Linking (TL) and Biased Preferential Attachment (BPA) models the generation algorithm is already presented in a step-wise way. However, also for WS it is easy to interpret the rewiring procedure that occurs on each node as a single step of the global rewiring process; the main difference is that in BA, TL and BPA the number of steps is independent of the size of the model (although not all combinations of network size and number of steps produce meaningful networks), while in Watts-Strogatz (WS) the number of steps would be a function of the network size.

As a consequence, all these models can be easily fitted in the meta-model we described. For example, let us consider the BA and the WS models. In the BA model, there are a fixed number of steps and at each step a new node-agent is created and subsequently activated. When activated it chooses m targets according to preferential attachment and sends them a *link-to* message. The semantics of the model is that such messages are always accepted and this causes a new link to be created.

In the original formulation of the WS model, for every node, we rewire each link with probability p. In the agent-based formulation:

- the activator activates each node in the network in sequence and
- each agent, when activated, rewires each of its connections with probability p.

One of the advantages of out approach is that, starting from this basic structure, variations of the WS model could be developed where the rewiring process is not governed by probability, but by other criteria, such as similarity between agents.

A second advantage is that, as the traditional models become more com-

plicated, the corresponding agent-based model usually does not increase in complexity. For example, when some of the more advanced generation models, like the BPA model, want to account for different, although coarse grained, node behavior, they introduce different sets of nodes and use set membership to distinguish among the nodes. Such models usually become too complex to be analytically analyzed. Moreover, this approach quickly becomes infeasible when the number of the different behaviors increases.

On the other hand, the same idea can be easily followed in agent-based model, due to the agent-based nature itself. To further explain the same example, in the agent-based variant of the BPA model we simply create three different behaviors to react to activation:

- a passive agent (which represent a node in P) simply refuses to act;
- an inviter agent (a node in I) "invites a new agent", that is to say, it asks the simulation engine to create a new agent;
- a linker tries to link to an existing node in the network with preference to "popular" agents.

There are some immediate advantages: (*i*) implementing new behaviors is relatively trivial and simply executing the simulation their impact is assessed; and (*ii*) in the agent-based version each agent could change its behavior during the simulation when appropriate conditions occur. The latter point is extremely important: for example, in the real world "inviters" do not have an unlimited number of friends to invite and at a given point in time inviters should become either linkers or passive nodes (or perhaps acquire a whole new behavior).

As the last example, we propose the TL model. The idea here is that at each step an agent is activated and, depending on the number of connections it has, it either introduces:

- two friends one to the other; or
- to another random node.

At each step a node is chosen and with probability p it is removed and replaced with a new agent with one random link. Probably the more revealing

symptom of the non agent-based origin of the TL model is the choice to model only the simultaneous circumstance that an agent leaves the network and another enters with a single event occurring with probability p. This assumption is quite unrealistic when modeling social networks, because people leaving or joining the network are mostly independent events. However it makes analytic study much easier because the number of nodes in the network is constant.

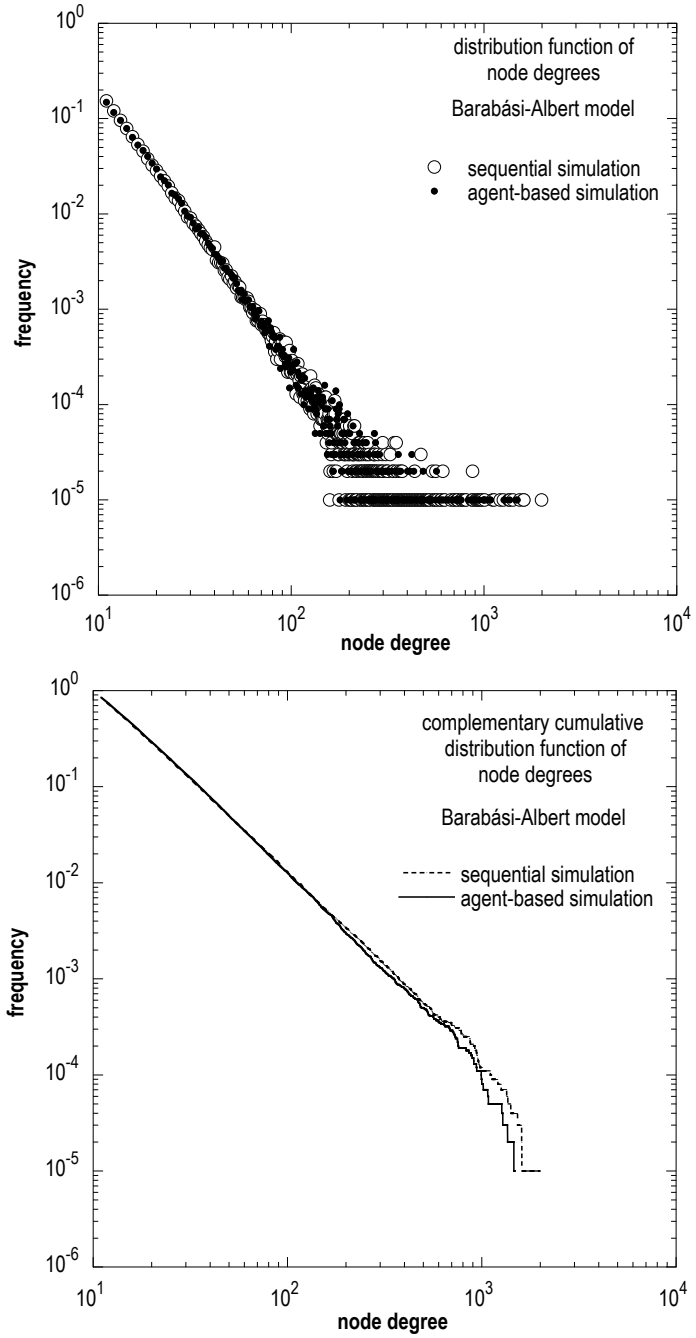

Figure 5.4: Comparison between the sequential and the agent-based implementation of the BA model with 100000 nodes and 11 starting edges per node. The upper and the lower panels represent, respectively, the distribution function and the complementary cumulative distribution function of the nodes in the network.

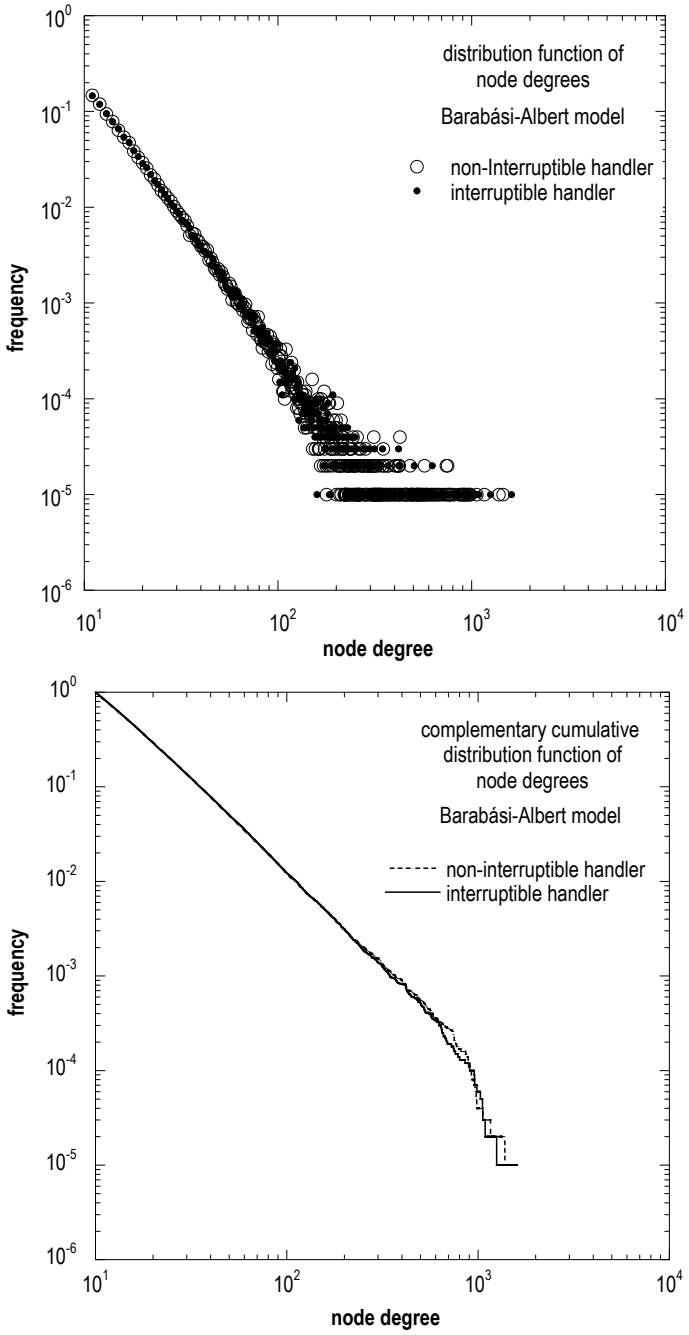

Figure 5.5: Comparison between agent-based implementations of BA model with 100000 nodes and 11 starting edges per node with interruptible and non interruptible handlers. The upper and the lower panels represent, respectively, the distribution function and the complementary cumulative distribution function of the nodes in the network.

Chapter 6

Design and Implementation of an Agent-Based Simulation Framework for Online Social Networks

In the last two decades, ABM was widely adopted as a research tool in the fields of social and political sciences, as we discussed in Chapter 4, since multi-agent systems are especially appropriate to model systems (*i*) where complex features arise from repeated relatively simple interactions, and (*ii*) dominated by discrete decisions. In particular, ABM gave important results in social science because it represents and simulates not only the behavior of individuals or groups of individuals but also their interactions that concur to the emergent behavior [12, 53].

In parallel with the theoretical development of ABM a number of software platforms were developed to ease the task of running the simulations; among these the most popular are Swarm [129], Mason [121], RePast [145] and NetLogo [180], which, however are not specifically tailored for social net-

work simulations.

In this Chapter, instead, we introduce a different kind of ABM tool that we specifically created for network generation and general processes over networks [69]. The tool we propose does not aim at being a general purpose agent-based modeling tool, thus remaining a relatively simple software system, whereas it is extensible where it really matters (e.g., supporting different representations for networks, from simple memory-based ones to pure disk-based storage for huge simulations). Its theoretical foundations lie deep in the generative approach to science that we discussed in Section 4.1 and in the meta-model that we developed in Section 5.1.

From a more practical point of view, the social network simulation system we propose has the following defining goals: (*i*) it must support both small and large networks; (*ii*) simulations shall be effortlessly run on remote machines; (*iii*) it must be easy to use, even for people without a strong programming background; (*iv*) deploying a large simulation should not be significantly harder than deploying a small one.

In our approach, the simulation system has four main components that can be modified independently: (*i*) the user interface, (*ii*) the simulation engine, (*iii*) the simulation model and (*iv*) the network database. The simulation model needs to be specified for each simulation and is the only part that has to be completely written by the user. Its specification is partly declarative and partly object-oriented. The user interface is responsible for taking input from the user, e.g., simulation parameters or information on the analysis to perform, and is specified declaratively. The simulation engine defines the concurrency model of the simulation, the scheduling strategies of the agents and the communication model among the agents. The network database actually holds a representation of the social network; it may be in-memory or on persistent storage, depending on the size of the simulation. Multiple network database implementations are provided and the framework can be extended with additional ones.

Large scale simulations typically require more resources than those avail-

able on a desktop-class machine and, consequently, need to be deployed on external more powerful machines or clusters. In order to simplify the operation, we designed our system so that a simulation can be entirely specified in a single file that can be easily copied or even sent by email.

Considering that the simulations are frequently run on remote machines, we opted for a command line interface, because a traditional GUI becomes more complex in this scenario. An added benefit is that batch executions are also greatly simplified. We also support read-eval-print-loop (REPL) interaction to dynamically interact with the simulation.

In order to allow people without a strong programming background to easily write simulations, we decided to create a *Domain-Specific Language* (DSL). A Domain-Specific Language (DSL) is a language providing syntactic constructs over a semantic model tailored towards a specific domain (e.g., building software). The idea is that DSLs offer significant gains in productivity, because they allow the developers to write code that looks more natural with respect to the problem at hand than the code written in a general-purpose language with a suitable library.

DSLs are usually categorized in *internal* and *external*: an external DSL is a new language that is compiled or interpreted (e.g., makefile),while an internal DSL is represented within the syntax of a general-purpose language, essentially using the compiler/interpreter and runtime of the host language. Mernik et al. discuss the issue thoroughly [127].

Traditionally, DSLs were sparingly used to solve engineering problems. Nowadays, with the widespread diffusion of very expressive languages such as Python, Ruby, Haskell or Scala the general attitude towards DSLs has changed ([67, 80]), especially because writing internal DSLs in such languages significantly lowers the developing costs to a point that is comparable to an API based solution [111].

Consequently, we decided to develop an internal DSL over a very high level programming language, in order to provide an environment that is friendly enough for scientists without strong programming background, without being

limiting for the others. Moreover, the internal DSL approach greatly simplifies external library usage, since all the libraries available for the host language are available for the DSL as well, and we think that accessing high quality scientific libraries and using an expressive language are both features of paramount importance.

In Section 6.1 we describe the runtime system of the simulation framework from the point of view of agent-based and distributed systems, in Section 6.2 and Section 6.3 we present the DSL design and the concurrency model, respectively. We also show how some well-known models are implemented in our toolkit at the end of Section 6.2. Eventually, in Section 6.4 we draw some conclusions.

6.1 PyNetSYM runtime system

The social network simulation system we propose, PyNetSYM (PYthon NETwork Simulation-analYsis-Method), has an elaborate runtime system that supports the execution of simulations providing only brief specifications. The nature of these specifications is described in Section 6.2. In this Section we describe (*i*) the various components that support the simulation, and (*ii*) the general structure of the simulation execution. We also discuss general semantic properties of the simulation engine as a concurrent system.

Simulation Engine

The central element of the runtime system is the agent, since the elements under simulations and several infrastructure components of the runtime system are implemented as agents. In the following we describe the design characteristics of PyNetSYM agents.

For our purposes an agent is a bounded unit with its own thread of execution. By bounded we mean that there is a clear separation between what is *inside* the agent and what is *outside* the agent. Agents have their own state, and access to that state is mediated by the agent itself. All the communication

among the agents occurs through message passing; each agent has a mailbox where the incoming messages are stored, and a unique identifier that is used to address the messages.

Agents also perceive and modify the environment. Our agents are not necessarily autonomic or goal-directed. Since they are used as a computational primitive, we need a lower-level specification that can be enriched to provide "real" agents but which does not place unnecessary overhead on the system.

The communication primitive is the **send** command. When an agent y executes the command **send**(x, `"m"`), (*i*) a message $m\{\}$ is created, (*ii*) it is delivered in the mailbox of x, and (*iii*) an empty placeholder is immediately returned to y as the return value of the **send** call, so that $r =$ **send**(x, `"m"`) is a valid command. When x processes $m\{\}$, its method `m` is invoked and the return value is placed in r.

Send provides both the semantics of synchronous and asynchronous messaging. In the following cases, the semantics is that of an asynchronous message: (*i*) **send** was invoked just as **send**(x, `"m"`), so that the return value is simply ignored, or (*ii*) **send** was called as $r =$ **send**(x, `"m"`), but the caller ignores the return value r.

On the other hand, the caller y can force the retrieval of the value of r and wait until $m\{\}$ is processed. In this case, y is blocked and control is given to another agent; y will be activated again after the value of r has been supplied by x. This is essentially the semantics of a synchronous message passing. Agent y can also check without blocking whether the value of r is ready; another possibility is trying to get the value blocking for a limted amount of time.

Messages with parameters can be sent with either:

$$\textbf{send}(x, \texttt{"m"}, p_1 = v_1, \ldots, p_n = v_n) \tag{6.1}$$

$$r = \textbf{send}(x, \texttt{"m"}, p_1 = v_1, \ldots, p_n = v_n) \tag{6.2}$$

In these cases the message $m\{p_1 = v_1, \ldots, p_n = v_n\}$ is delivered to x and the m method of x is invoked with actual parameters $v_1, \ldots, v_n$ passed to the formal arguments $p_1, \ldots, p_n$ of the method.

The default behavior of the agents is waiting for the arrival of the messages and then processing the incoming messages with the appropriate handlers. However, full agent behavior (autonomous, goal-oriented, and pro-active) can be implemented either supplying a different behavior or augmenting the default with pro-activeness.

Another important design decision regarding the system semantics is whether to implement cooperative or preemptive multi-tasking. Several popular languages and platform chose preemptive multi-tasking because in general purpose systems the probability and the risks of a misbehaving application consuming all the CPU time is too high. However, for a simulation oriented platform, such risks are minimal and we opted for cooperative multi-tasking because it allows a more deterministic control of complex time sequences.

As a consequence, in PyNetSYM a method handling a message can only voluntarily "give up" the execution for a while, either explicitly going to sleep or by requesting a blocking operation. In all other situations, when an agent starts processing a message, it continues until termination. This property is analogue to the semantics of the Actor Model [3] and simplifies formal reasoning on the system. Moreover, from the point of view of the emergent properties of the simulation it has little impact [25].

When an agent has an empty mailbox, it can choose to be removed from main memory and have its state saved on secondary storage. If the stored agent is subsequently sent a message, it is restored in main memory from the saved state. This behavior is extremely convenient considering that for most social network topologies, a small fraction of agents is responsible for the vast majority of the links. Since in most processes over networks the agents with few links are seldom activated, we can save memory keeping them in secondary storage and do not lose much CPU time.

Another important memory-related issue is the storage of the network itself. A possible solution is completely distributing the knowledge of the network among the agents, so that each agent only knows its neighbors and the network must be reconstructed from their interactions. However, several net-

work processes could not be implemented efficiently in this scenario, because several elementary operations would involve too much communication among the agents. Consequently, we prefer to maintain a global view of the network. From the point of view of ABM, the decision is consistent with the interpretation of the network as the environment, as: (*i*) agents can interact with it by creating or destroying links, and (*ii*) the agents behavior is influenced by the network in several process dependent ways.

This view is presented as a software component that we call network database and that provides a unified interface that agents can use to modify and query the network state. We provide multiple implementations in order to be able to balance the various trade-offs. Some implementations are RAM based, and their main advantage is to provide more efficient accesses when the network is not excessively large; others are backed with various secondary-storage based solutions, which results in slower operations, but allows for simulations on larger networks.

As a general comment, it should be said that, although PyNetSYM provides an interface to the actual representation, memory and CPU issues remain. Let us consider two different network representations that are reasonable extremes regarding RAM usage. While the NetworkX library provides many useful algorithms and has a thorough support for edge and node attributes, a network of order $n = 10^6$ and size $m \sim 10 \cdot n$ occupies 4-5 GB of RAM when represented as a NetworkX graph. On the other hand, the same network, represented as a sparse matrix occupies less than 2 MB; however, with the latter approach the attributes are much harder to manage. Different situations have different trade-offs and consequently we are working to make the choice of underlying representation as transparent as possible.

Moreover, depending on the actual network process, some implementations are more appropriate than others, because they are optimized for certain operations that occur frequently in the process or choice.

Simulation structure

The actual simulation is divided in two distinct phases (*i*) setup and (*ii*) execution. During the first phase (setup), the system is initialized so that it reaches the initial configuration specified by the simulation. First, various infrastructural agents (e.g., Activator, NodeManager) are created and started, so that they are ready to receive messages, and the Clock (if present) is also created, but not started.

Later during this phase, the Configurator agent instructs the NodeManager to (*i*) create the initial nodes in the network, (*ii*) to give them instructions to establish the connections they are meant to have at t_0, and (*iii*) to provide them with any other initial information that the simulation may require .

The NodeManager is generally responsible for (*i*) creating the new agent-nodes, passing them the appropriate initialization parameters and (*ii*) monitoring them, so that their termination (exceptional or not) is managed.

We created different Configurator agents for the most frequent needs, that are (*i*) reading an initial network specification from disk and setting the system up accordingly, or (*ii*) creating n node-agents of a given kind. When reading network specifications from file, we support (*i*) popular file formats for exchanging networks, such as GraphML or Pajek, (*ii*) networks saved as sparse matrices in HDF5 files, and (*iii*) networks stored in various DBMS, such as MongoDB.

After configuring the simulation, (*i*) the Configurator agent terminates, and (*ii*) if present, the Clock agent starts, marking the transition to the execution phase, or (*iii*) the node-agents are otherwise notified that the simulation begins.

During the execution phase the node-agents perform the simulation according to their behavior. Although from a theoretical point of view such behavior can be completely autonomous and do not rely on an external time schedule, in practice most network generation models and generic processes over networks can be described in terms of a relatively simple meta-model as the one

we described in Section 5.1.

6.2 Defining a Domain-Specific Language for network simulations

At the beginning of this Chapter, we mentioned the advantages of a DSL in terms of ease of development for both programmers and non-programmers because of the increased language expressivity. In this Section we describe the general structure of our DSL, which is an internal DSL hosted by Python. We also present some examples.

Our choice of language was motivated by the following features: (*i*) focus in readability; (*ii*) ease of use; (*iii*) availability of useful libraries for scientific computations and consequently (*iv*) widespread adoption in many scientific areas; (*v*) solid choice of concurrency frameworks and frameworks for distributed or GPU based computing; (*vi*) powerful REPL implementations (*vii*) advanced metaprogramming capabilities, which we extensively use to improve the expressiveness of the DSL. The host language is almost an implementation detail, since other object oriented high level languages such as Ruby or Scala could have provided the same features.

Here we review some aspects of the Python language that are significantly different from static languages such as Java or C++:

(a) **class** A(B) means that class A is a subclass of class B.

(b) Methods are defined with the **def** keyword. For clarity, we differentiate among methods and message handlers; the latter are introduced with the **handler** keyword, even though this keyword is not part of Python's syntax.

(c) The explicitly specified formal parameter "self" is the object the method is invoked on.

(d) Methods can be invoked with named parameters.

(e) Set ($\{a, b\}$) and hash-map ($\{\mathrm{k} : \texttt{"v"}, \ldots\}$) literals are available.

(f) Everything defined directly in the class body becomes a class attribute. A callable defined in the class body is a method.

(g) Classes are first class objects: when called, are factories.

(h) Classes can be defined inside other classes; in this case, they are "attributes" of the enclosing class. As a consequence, an inner class can be invoked as a method of the enclosing class and returns the appropriate instance object.

(i) The **send**(x, `"m"`) command discussed in Section 6.1 becomes a method defined in the Agent class and, as such, is invoked with self.send(x, `"m"`).

As described in Section 6.1, our simulations have two distinct logical elements:

1. the essentially imperative/object oriented description of the agents behavior (e.g., the nodes and the Activator agent)
2. the mostly declarative description of the simulation itself and of the simulation options specification.

Regarding the imperative/object oriented part, it suffices to say that the nodes and the Activator agent are implemented as subclasses of Agent, which means they are essentially Python objects with agent semantics provided by PyNetSYM and discussed in Section 6.1. Their behavior is thus specified using the host language (Python) for maximum expressivity.

The other logical element defining a model is the Simulation class. A Simulation is not an agent and is essentially the executable specification of the simulation that collates together all the other elements. A Simulation object has a `run` method that is called to execute the simulation. When `run` is called, both (*i*) command line arguments and (*ii*) actual parameters directly passed into `run` are taken into account, i.e., it processes the "simulation options" field and creates a command line parser that can parse command line options according to the specification

The specification is a list of allowed options, each with its name, default value and the function to convert the string value as specified in the command line to the proper value. For example, the following specification would let

the simulation accept two options, *option1* and *option2*; the first option has default value 1.0 and is of type float, the second has 1 as the default value and is an integer:

```
command_line_options = (
    ('--option1', {'default': 0.0, 'type': float}),
    ('--option2', {'default': 1, 'type': int}))
```

Moreover, a help message is automatically constructed from the options, and is displayed if the simulation is run with the `"--help"` option. Such message lists the various options and, if a documentation string was provided in the specification, also their meaning.

When a simulation is executed, it always instantiates some objects and agents that fill pre-determined roles, such as the network database, the Configurator agent, the Activator agent, or the Clock agent.

Additionally, the behavior of simulations can be declaratively customized. If the body of the Simulation subclass has an attribute in the form `role_type`, it is used as the factory for the object in question in place of the default implementation. For example, a different Activator can be requested with the `activator_type` option.

A subset of the simulation options is passed to the constructors of objects and agents instantiated by Simulation. The subset is determined in the following order: (*i*) inspecting the "options" attribute in the object class definition, (*ii*) using an attribute named role_options defined in the Simulation class, and (*iii*) performing introspection on the names of the parameters of the object/agent constructor. Additionally, the Simulation class has an `additional_agents` attribute, where additional "roles" can be defined.

If the configuration is wrong or incomplete (e.g., because of a missing or wrongly typed parameter), the simulation fails as early as possible, ideally at "compile time" (i.e., when classes are evaluated).

```
1: class SIR_Activator(pynetsym.Activator):
2:     handler infected(self, node):
3:         self.infected_nodes = self.infected_nodes.add(node)
4:     handler not_infected(self, node):
5:         self.infected_nodes = self.infected_nodes.remove(node)
6:     def nodes_to_activate(self):
7:         return self.infected_nodes
```

Figure 6.1: SIR model Activator implementation.

```
8: class SIR_Node(pynetsym.Node):
9:     handler initialize(self, state):
10:        self.state = state
11:        if state == "I" :
12:            self.send(Activator.name, "infected", node=self.id)
13:    handler infect(self, state):
14:        if self.state == "S" :
15:            self.state = "I"
16:            self.send(Activator.name, "infected", node=self.id)
17:    handler activate(self):
18:        if state == "I" :
19:            for each node in self.neighbors()
20:                if random.random() < self.infection_rate :
21:                    self.send(node, "infect")
22:            if random.random() < self.recovery_rate :
23:                self.state = "R"
24:                self.send(Activator.name, "not_infected", node=self.id)
```

Figure 6.2: SIR model node implementation.

SIR model

In this Subsection we present the implementation of the Susceptible-Infected-Recovered (SIR) model in our DSL. The SIR model was originally proposed to study the outbreak of contagious illnesses in a closed population over time [103] and was subsequently adapted as a social network process [150]. For our purposes, the SIR model is the network adapted variant. In this form, each node has three possible states: susceptible (S), infected (I) and recovered (R), hence

```
25: class SIR_Simulation(pynetsym.Simulation):
26:     simulation_options= {
27:         ("--infection-rate", {"default" : 1., "type" : float}),
28:         ("--recovery-rate", {"default" : .4, "type" : float}),
29:          ("--initial-infected-fraction",
30:             {"default" : .01, "type" : float})}
31:     activator_type = SIR_Activator
32:     activator_options = {}
33:     class configurator_type(pynetsym.NXGraphConfigurator):
34:         node_type = SIR_Node
35:          node_options= {
36:             "infection_rate",
37:             "recovery_rate",
38:             "initial_infected_fraction" }
39:         def initialize_nodes(self):
40:              infected_nodes = random.sample(
41:                 len(self.node_identifiers) * self.initial_infected_rate)
42:             for each node in self.node_identifiers
43:                 if node in infected_nodes :
44:                     self.send(node, "initialize", state="I")
45:                 else:
46:                     self.send(node, "initialize", state="S")
```

Figure 6.3: SIR model Simulation specification.

the SIR name. The system starts with a given ratio r of infected patients, and each infected patient can recover with probability γ. Moreover, each infected patient infects each of its neighbors with probability β.

We fit the SIR model to our meta-model so that, at each step, the Activator agent only activates infected nodes (Figure 6.1, lines 6–7). Susceptible and recovered nodes do not need to take action. Consequently, when a node is infected, it sends a message to the activator to inform it that it is infected and, similarly, when it recovers, it sends a message indicating its recovery. When an infected node is activated, it tries to spread the disease among its neighbors sending them messages. Afterwards, it may recover with a given probability.

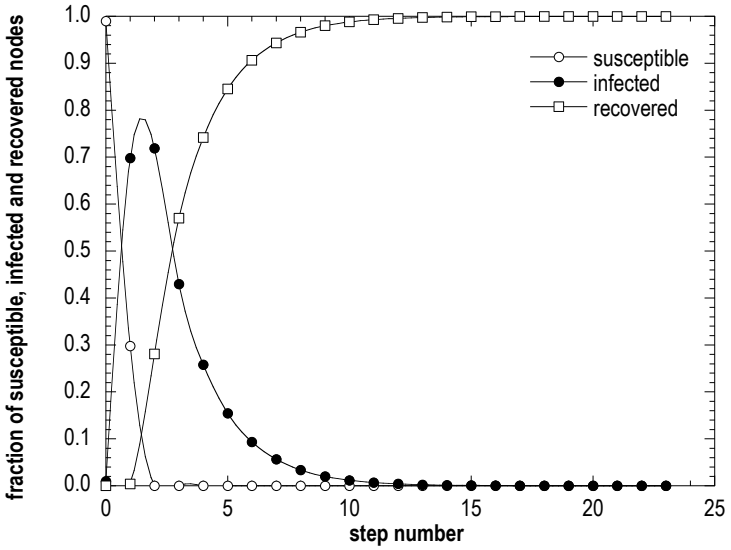

Figure 6.4: Time evolution of the fraction of susceptible, infected and recovered nodes as functions of the number of steps according to the PyNetSYM agent-based simulation of the SIR model. The starting network is a $G(10^5, 120)$ network, the starting parameters are $\beta = 1.0$, $\gamma = 0.4$, $r = 0.01$. The solid lines are guides for the eye.

In the implementation with PyNetSYM the simulation has three initial parameters: the infection rate β, the recovery rate γ and the initial fraction of infected nodes r. The declaration of the simulation options is given in lines 26–30 of Figure 6.3.

An additional starting parameter is the shape of the network. The initial configuration of the network is read from a file, hence we use NXGraphConfigurator (line 33 of Figure 6.3), that (*i*) reads files in any format supported by NetworkX, and (*ii*) it creates the appropriate nodes along with their connections. The type of the nodes and the simulation options they require are specified with the node_type and node_attribute of the Configurator respectively. Eventually, initialize_nodes is a special method of the Configurator agent which is called after the nodes have been created to provide them with

additional setup. In this case, it randomly selects which nodes start infected.

In Figs. 6.1 and 6.2 the implementation of the SIR nodes and of the SIR Activator are shown. Essentially, their behavior is that described when discussing how to fit the SIR model to the meta-model.

In Figure 6.4 we plot the evolution of the fraction of susceptible, infected and recovered nodes as functions of the elapsed number of steps as given by the PyNetSYM simulations of the SIR model. The starting network is an Erdős-Rényi graph $G(10^5, 120)$ [57], i.e., a random graph where each of the 10^5 nodes has 120 random edges. The 1% of the nodes starts as infected ($r = 0.01$) and the parameters here are infection rate $\beta = 1.0$ and recovery rate $\gamma = 0.4$.

Barabàsi-Albert model

In Figure 6.5 we show the code for a simulation implementing the BA model [16]. Line numbers reported in this Subsection refer to Figure 6.5. The BA model starts with n_0 nodes and no edges. At each step a new node with m random links is added. The m links are directed towards nodes with a probability proportional to their degree, this strategy is called *preferential attachment*.

In Lines 16–23 the simulation configuration is specified. Line 17 specifies that the BA_Activator (defined in lines 9–15) is the activator to be used. Classes can be stored in variables like any other object.

Lines 18–20 specify the command line options, their default values and the type of the parameters. When values for some options are not specified in the command line, the default value is used.

The BasicConfigurator (Lines 21–23) reads the value n_0 of the simulation option `"--network-size"` and requests to the NodeManager the creation of n_0 nodes of type specified by the `node_type` attribute

In Lines 1–8 and 9–15 there are the specifications for the nodes and the activator respectively. The nodes are specified providing a handler for the *activate*{} message. Moreover, in the Node class we defined handlers for the *link_to*{source} message, so that the graph is updated with the information.

```
1: class BA_Node(pynetsym.Node):
2:     handler activate(self):
3:         targets = set()
4:         while len(targets) < self.starting_edges :
5:             target = self.graph.preferential_attachment()
6:             if target not in targets :
7:                 targets.add(target)
8:                 self.send(target, "link_to", source=self.id)
9: class BA_Activator(pynetsym.Activator):
10:    options = {"starting_edges"}
11:    handler tick(self):
12:        answ = self.send(NodeManager.name
13:            "create_node", cls=BA_Node,
14:            parameters= [self.starting_edges])
15:        self.send(answ.get(), "activate")
16: class BA_Simulation(pynetsym.Simulation):
17:    activator_type = BA_Activator
18:    simulation_options= {
19:        (network_size, {"default": 100, "type": int}),
20:        (starting_edges, {"default": 5, "type": int})}
21:    class configurator_type(BasicConfigurator):
22:        node_type = BA_Node
23:        node_options = {"starting_edges"}
```

Figure 6.5: BA model simulation implementation.

The Activator accepts a *tick*{} message from the clock. In Line 11 the Activator sends the Node-Manager: (*i*) the class of the agent to create in the "cls" parameter and (*ii*) the arguments to pass to their constructor with the "parameters" parameter. This is a synchronous call, because the Activator blocks waiting for the identifier of the newly created agent. Then, it immediately sends an *activate*{} message to the new agent (line 15).

6.3 PyNetSYM Concurrency Approach

In Section 6.1 we defined the runtime model of our simulation framework. Essentially, we assumed that agents do have their own thread of control, but

Table 6.1: Comparison between Greenlet and thread efficiency: the main agent spawns a new agent and sends it a message with the numeric value of the number of agents to spawn, then it waits for a message on its own queue. When the other agent receives a message with value n, each secondary agent: (*i*) spawns a new agent and sends it a message with the value $n - 1$ if $n \neq 0$; (*ii*) on the other hand, if $n = 0$, it means that all the agents have been created and sends a message to the main thread. Notice that the agents do not exist all at the same time: once they have received and sent a message, they terminate.

# items	Thread execution (s)	Greenlet execution (s)
$1 \cdot 10^2$	0.022	0.002
$5 \cdot 10^2$	0.110	0.009
$1 \cdot 10^3$	0.224	0.018
$1 \cdot 10^4$	2.168	0.180
$1 \cdot 10^5$	21.85	1.796
$1 \cdot 10^6$	223.3	18.24

did not specify how concurrency was implemented. The traditional approach in agent-based frameworks is to implement concurrency mainly with threads and to provide cooperative multitasking as an option [19].

Instead, we decided to use gevent[1], a networking and concurrency framework that uses coroutines to implement cooperative tasks, called "greenlets" in the gevent lingo. Coroutines are a generalization of subroutines allowing multiple entry points for suspending and resuming execution at certain locations [44]. Gevent also provides the greenlets scheduler. Frameworks such as gevent are popular for writing programs with very high concurrency, since greenlets: (*i*) are less expensive to create and to destroy; and (*ii*) do not use memory structures in kernel space. In fact, greenlets live entirely in the user-space, thus context switches between different greenlets are inexpensive as well.

In Table 6.1 we report execution times of a simple benchmark performed with a different number of threads/greenlets. It is easy to see how the greenlet based solutions are roughly ten times faster than the corresponding thread-

[1] `http://www.gevent.org`

based ones. Further comparisons between greenlet and thread performances can be found in the literature [78].

The better performance of greenlets is particularly relevant in our case, because in ABM it is frequent to support millions of concurrent agents, and, since greenlets use no kernel resources, their number is limited only by the amount of available RAM. Moreover, experience suggests that a modern operating system supports few thousand threads. Although thread pools offer a viable, albeit more complex solution, cooperative multitasking, as provided by greenlets, gives a finer grained control over scheduling and concurrency in general.

6.4 Conclusion

In this Chapter, we have presented PyNetSYM, a novel language and runtime for network specific simulations. PyNetSYM is built for the generative approach [53] to science typical of Agent-based Modeling (ABM). We believe there is a strong need for tools that are both: (*i*) easy to use (especially for people with little programming background but with a significant expertise in other disciplines, such as social sciences) and (*ii*) able to tackle large scale simulations, using remote high-performance machines and potentially distributing the computation on multiple servers. Therefore, while our primary goal is maintaining our toolkit simple and easy to use, efficiency is our second priority, since nowadays there is a wide interest on networks of huge size.

Specifically, we created PyNetSYM: (*i*) to easily support both small and huge networks, using either in-memory and on-disk network representations, and (*ii*) to be as easy to be used both on personal machines or on remote servers.

We designed PyNetSYM so that all the entities, both the infrastructural ones and those under simulation, are agents: defining the network simulation is essentially a matter of specifying (*i*) the behavior of the nodes and (*ii*) a few additional simulation parameters (e.g., storage strategy and user-customizable

options).

Given the merits of Domain-Specific Languages (DSLs) in general, and specifically the ones concerning how to make development easier both for programmers and non programmers alike, we decided to create a DSL to drive PyNetSYM simulations, so that it is possible to write programs that are machine-executable, high-level formulations of the problem to solve. Specifically, our DSL is an internal DSL over Python.

As PyNetSYM provides the simulation engine, the simulation can be written in a simple file using our DSL. Thus, PyNetSYM models are very easy to deploy (copying around a single file is sufficient) and can also be effortlessly shared between researchers. Moreover, PyNetSYM models can also be written and executed interactively from a REPL, a feature that is extremely useful when exploring the results of a simulation, because it makes possible to interactively and dynamically perform analysis and visualize data.

We also implemented some classic processes over networks (generation models, infection/information diffusion processes) with PyNetSYM and found that the resulting programs are readable and fluent; in fact, they are very similar to a pseudo-code formulation of the models, even though they are efficiently executable.

We also used PyNetSYM to simulate the behavior of users in a novel fully distributed social networking platform, in order to understand the condition under which the information propagates to the intended recipients. The results are presented in Chapter 7.

Our results show that our approach is successful in providing a friendly and easy to use environment to perform agent-based simulations over social networks. Agent-based simulation is a powerful conceptual modeling tool for social network simulations and with the present work we contribute a natural and expressive way to specify social network simulations using a DSL.

Chapter 7

Agent-based Models of Distributed Social Networking Systems

A rather recent research trend is that of distributed social networking systems [5, 33, 34, 42, 72, 75, 76, 83]. The proponents of such systems put forward several reasons for their creation, among which the most important are:

- the demanding terms of service, essentially asking their users a non-exclusive, transferable, sub-licensable, royalty-free, worldwide license to use content that they submit [59, 182]
- the fact that service providers can provide a-priori or a-posteriori censorship of the data submitted and may be forced for legal reasons: (*i*) to perform such actions and (*ii*) to disclose all the information they have, no matter how private
- general privacy concerns related to the use that service providers can make with the data, especially considering the terms of service, and the fact that the data can be used for commercial reasons.

There are two main categories of decentralized and distributed social networks: federated and P2P. In a federated system multiple entities cooperate

to provide the service and each of them gives access to the whole system to a subset of the total users. On the other hand, in a P2P system, every participant is both a user and a system provider at the same time. However, purely P2P system can have issues with data availability, i.e., post from very badly connected users can be difficult to obtain. Consequently P2P system may introduce "super-nodes" with a role akin to that of a federated provider.

We developed a novel P2P micro-blogging platform that we called Blogracy [72]. We built Blogracy upon the Bittorrent file-sharing protocol [38], and rely on the built-in Bittorrent Distributed Hash Table (DHT) to locate information. As such, Blogracy is entirely distributed and does not require any centralized infrastructure.

However, it is possible that it suffers of the same data availability issues that some P2P systems experience. In order to understand whether Blogracy can provide a satisfactory user experience, without considering the unlikely eventuality of a large scale deployment, we resort to modeling and simulation by taking advantage of the modeling framework that we created as part of our research and that we described in Chapter 6. In Section 7.1 we describe the models and in Section 7.2 we discuss some metrics we propose to assess whether the system could be successful. In Section 7.3 the results of the simulation are presented and some conclusions are drawn.

7.1 Agent-based model of a P2P OSN

In a centralized social networking system, when a user creates a new resource, e.g., a post, it sends the resource to the remote system, where (*i*) it is stored, and (*ii*) it can be retrieved by other users at a later time. In constrast, in a P2P OSN there is no centralized storage. The resources are shared among the system users, and when someone requires a resource, the peers that already own it should provide it.

Similarly to what occurs in regular P2P system, a serious issue with OSN is the availability of the resources, which are available only if some of the

peers that previously downloaded them are active. The availability of popular resources is usually good, because if several users own the resource, some of them is likely to be sharing it. On the other hand, rare resources may be difficult to obtain.

In order to improve availability, several mechanisms are created to encourage users to keep sharing the resources they obtained [203]. The most popular P2P systems, Bittorrent and eMule, have reputation-based strategies that reward users for letting their peers download resources that they have previously downloaded. As a consequence, users tend to cooperate and re-share [7].

If a P2P OSN is based on existing P2P systems, as is the case of Blogracy [72], such mechanisms are already in place. It is also reasonable to assume that users reshare recently generated resources obtained from their friends, especially considering that such behavior is necessary also to speed-up their own networking experience.

Another strategy that may improve the availability of the resources in P2P OSNs is the introduction of "supernodes" that are always active, so that these nodes can act as buffers for otherwise not very popular resources, improving the overall system performance. Considering that most resources in an OSN are text or relatively small images, the effort required for providing such supernodes is relatively mild. Moreover, it is possible to reward users engaging in such behavior, without externally providing the service.

Nonetheless, it is an interesting question whether P2P OSN could be successful in practice, i.e., whether the various resources that OSN users generate are actually obtainable by their friends. Therefore, we propose several models to asses the feasibility of our approach in building a P2P OSN.

In order to better understand the system behavior, we started creating a very simple model. The users of the OSN are modeled as nodes in a graph where there is a link from node x to node y if x is interested in the messages created by y. This is essentially a "follows" relation. The network is not assumed to vary, because we simulate only relatively brief timespan, so that the effects of the network variations are negligible.

At each step t_i of the simulation:

1. Each node writes a new message with probability p_w; if a node writes, it is added to the set of nodes that are active at step t_i ($\mathrm{Act}(t_i)$).
2. Each node reads posts from its neighbors with probability p_r. The reading nodes are also added to the list of nodes active at step t_i.
3. Each active node:
   a) requires the list of unread messages from each of its active neighbors,
   b) marks which neighbors it has seen during step t_i; this fact is used to retrieve only messages that have not been read yet.

Moreover, we account for a fraction r of "supernodes" that are always active. The supernodes do not necessarily create posts at each step: the writing behavior is governed by the same probability p_w of every other node. We consider two strategies to attribute the supernode status: (*i*) according to a uniform distribution, (*ii*) proportionally to the node degree, that hereafter will be termed as "ran" and "pa", respectively.

In this simple model, the only nodes that are able to provide a resource (a message) to those requesting it are the message creators. We call this model "read-and-forget" (RAF), because we are essentially assuming that the followers immediately delete the content they retrieved after reading it.

On the other hand, in a P2P system, once a peer downloads a resource, it reshares it. In our case, this would potentially make the resource available even in steps when the resource creator is not active.

In order to model this second scenario, which we call "read-and-share" (RAS), we add to the model a fourth step:

4. Each active node
   a) compiles a list of neighbors that are not active at step t_i;
   b) it queries its active neighbors for messages from the non active neighbors.

In this model, each node saves all the messages it received. This assumption is not unrealistic considering that we model only relatively brief periods

of time, during which a real system would cache the retrieved messages to improve repeated readings in any case.

7.2 Quality metrics

In this Section, we discuss some metrics that can be used to assess the system performance. Be M a message, then $T_c(M)$ is the step of the simulation when the message was created and $C(M)$ is its creator. With $N(x)$ we denote the set of the neighbors of node x and with k_x its degree. Consequently, $N(C(M))$ is the set of the recipients of message M, and $k_{C(M)} = |N(C(M))|$.

We indicate with $R(M, y)$ the step when the node y receives the message M, with $y \in N(C(M))$. If y never receives M, $R(M, y) = \infty$.

Let $L(M, y)$ be the reception lag of message M by node y, let $L_\mu(M)$ be the mean reception lag for message M and $L_{\max}(M)$ the maximum reception lag:

$$L(M, y) = R(M, y) - T_c(M) \tag{7.1}$$

$$L_\mu(M) = \frac{1}{k_{C(M)}} \sum_{y \in N(C(M))} R(M, y) - T_c(M) \tag{7.2}$$

$$L_{\max}(M) = \max_{y \in N(C(M))} R(M, y) - T_c(M) \tag{7.3}$$

The distributions of $L_\mu(\cdot)$ and $L_{\max}(\cdot)$ provide us with important information regarding the system performance. If on average the messages have low reception lags, it means that the users interested in a given post are likely to receive it timely and the system performance is good. Moreover, if also the maximum lags experienced are not too high, it means that also the worse case is acceptable.

However, if a node y receives a post M after a large number of steps, there are two factors that contribute to the large value of $L(M, y)$:

– it was difficult to find another node that could share the message, which is an issue of low availability and ultimately a problem with the system; and

– the receiving node did not connect for a long time, so that, in fact, the message would have been available for download earlier, it was just the downloader that was not online.

The latter factor is not an issue with the system, and users are not going to complain, because, from their point of view, the only perceivable lags are those originated by the first factor. In order to evaluate the respective contributions of the two factors, another metric is more suitable.

Let $\mathrm{OR}(M, y)$ be the optimum reception step for message M by node y, i.e., the first step after the creation of M when y became available. In formulas, $\tau(y)$ be the set of steps when y was available, then:

$$\mathrm{OR}(M, y) = \min\{t \in \tau(y) : t > T_c(M)\} \tag{7.4}$$

In other words, $\mathrm{OR}(M, y)$ is the step in which y would have received the message had it been accessible from a centralized and always connected repository. "Optimum" here means that y could not have received the message earlier, since y itself was not connected.

Using $\mathrm{OR}(\cdot)$ instead of $T_c(\cdot)$, it becomes possible to define a different set of metrics, which are: (*i*) the optimum reception lag $\mathrm{OL}(M, y)$ of message M by node y, (*ii*) the mean optimum reception lag $\mathrm{OL}_\mu(M)$ for message M, and (*iii*) the maximum optimum reception lag $\mathrm{OL}_{\max}(M)$. The meaning of "optimum" is the same explained in the previous paragraph. In formulas:

$$\mathrm{OL}(M, y) = R(M, y) - \mathrm{OR}(M, y) \tag{7.5}$$

$$\mathrm{OL}_\mu(M) = \frac{1}{k_{C(M)}} \sum_{y \in N(C(M))} (R(M, y) - \mathrm{OR}(M, y)) \tag{7.6}$$

$$\mathrm{OL}_{\max}(M) = \max_{y \in N(C(M))} (R(M, y) - \mathrm{OR}(M, y)) \tag{7.7}$$

7.3 Simulation Results and Conclusions

Analysis of the Simulation Parameters

As discussed in the previous Section, the starting parameters for a model are:

- the network that defines the follows relations;
- the probability p_w to write a post at each step;
- the probability p_r to read messages;
- the fraction of supernodes r, which ranges from 0.0 (no supernodes) to 1.0 (all supernodes);
- the supernode status assignment strategy, whether the supernode status is assigned uniformly or with a bias towards highly connected nodes ("ran" or "pa").

We discussed two models: in the first one, the nodes "forget" a resource as soon as they receive it (RAF), in the second, they keep sharing every resource that they receive (RAS). We already argued that the latter behavior is not unrealistic provided that we simulate the system for relatively brief timespans.

A few considerations regarding the various simulation parameters are in order. Our main starting network is a random network generated with the Holme and Kim [93] algorithm, briefly discussed in Chapter 2. Essentially, the Holme and Kim algorithm is a preferential attachment process with an additional triadic closure step to increase the clustering coefficient. Networks generated with this algorithm have a power-law degree distribution and high clustering coefficient. Therefore, they are a suitable starting point for our models. We generated networks with 1000 nodes, 50 starting edges for new node, plus 20% probability to add a triadic closure. Consequently the generated networks have clustering coefficient $C = 0.2$ which is a realistic value according to the analysis made in Chapter 2.

We also run the simulation with larger networks, but the results do not change significantly. This can be explained considering that in our model all the interactions involve only neighbors and do not propagate through the network. Consequently, the non-local properties or the scale of the networks have little impact on the outcome.

In order to attribute realistic values to p_w and p_r, we considered Twitter usage patterns. In Twitter the average number of post per user per day is almost 1. We assume that each step of our simulation engine corresponds to

15 minutes of real time. Considering the model and the 1 step to 15 minutes correspondence, the number of post each node generates in a day is a random variate with binomial distribution $B(n, p_w)$, where $n = 24 \cdot 4$ is the number of steps in a day. Since the expected value of $B(n, p_w)$ is $n \cdot p_w$ and since we want it to be equal to 1, we set $p_w = \frac{1}{96}$. Our simulations typically last 100 steps (roughly 1 day) and data is collected in sequences of 10 simulations, so that we can take average values.

More accurate models can take into account that there is a rather strong correlation between node degree and the number of generated posts [1], i.e., very active users tend to have more followers. However, such condition would simply make the models predict a better performance with respect to our model, because if more messages come from very active users, such messages are also easier to be downloaded. Consequently, if our simplified model show that the system could be successful, then the system would only be more successful in practice.

Attributing a realistic value to p_r is harder, since there are no precise statistics on how often users read the posts. It is however known that users visit social networking sites several times a day [172]. Therefore, we tried different values and the simulation results do not change considerably and then we somewhat arbitrarily set $p_r = \frac{6}{96}$, i.e., we assume that users visit the site 6 times per day on average.

The ratio of supernodes in the system is r. There are multiple reasons why a user has supernode behavior. In our case, since Blogracy runs over Bittorrent applications that can also be used for regular filesharing, it is reasonable to assume that some users would not only use Blogracy, but also the underlying P2P platform, so they would want to be always connected anyway. Another kind of supernodes could be a few purposely created nodes that act in a way similar to the federated servers in a federated social networking system.

It is worth noting that $r = 1$ would be the ideal condition when everyone is online at the same time, so no reception lag would occur at all. We performed

[1] http://www.sysomos.com/insidetwitter/

r	$\langle L_\mu \rangle$	$\langle \mathrm{OL}_\mu \rangle$	$\langle L_{\max} \rangle$	$\langle \mathrm{OL}_{\max} \rangle$
0	155.6	142.8	638.65	572.16
0.1	129.7	117.9	583.57	517.63
0.25	94.29	84.43	483.17	419.13
0.5	43.42	36.85	325.21	267.29

Table 7.1: Mean values $\langle L_\mu \rangle$, $\langle \mathrm{OL}_\mu \rangle$, $\langle L_{\max} \rangle$ and $\langle \mathrm{OL}_{\max} \rangle$ of L_μ, OL_μ, $L_{\max}$ and $\mathrm{OL}_{\max}$, respectively, calculated by using the RAF model and the "ran" status assignment to supernodes.

simulations with values of r equal to 0, 0.1, 0.25 and 0.5. We do not expect 0.5 to be a realistic value, but it was interesting to see the impact of such a high ratio.

Finally, we should discuss the reasons of the supernode status assignment strategy. The simplest strategy ("ran") we account for is choosing the supernodes according to a random distribution. Such strategy models the fact that supernode status depends on some variable that is unrelated to any node feature. The other strategy ("pa"), instead, assumes that highly connected nodes are more likely to be supernodes. The last strategy is based on the fact that supernodes introduced by system operators would probably have very high degrees, because several users would like to link them to potentially improve the diffusion of their posts. Similarly, in real micro-blogging platforms we observed that well connected accounts are also more active.

RAF model

Here we comment some of the simulations based on the RAF model and the "ran" strategy. Let us consider the distribution of mean optimum reception lags OL_μ (Figure 7.1) for different values of r.

We can see that with $r = 0$ the distribution is essentially a normal distribution with mean $\mu \sim 140$ and $\sigma \sim 40$. On the other hand, with $r > 0$ some nodes act as supernodes and consequently the messages they create are always available for download. The distribution essentially becomes bimodal, with a

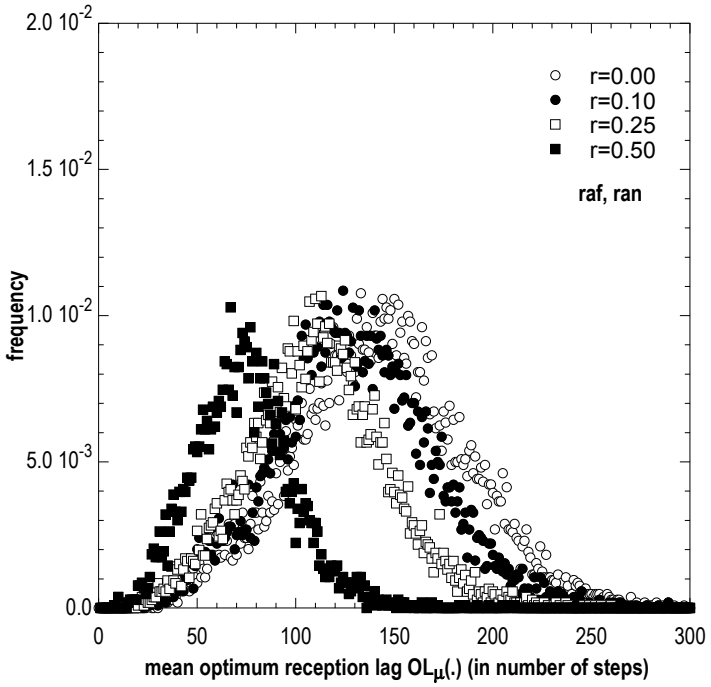

Figure 7.1: Distribution of the mean optimum reception lag OL_μ for different values of the supernode fraction r, calculated by using the "RAF" model and the "ran" status assignment to supernodes. See text.

fraction $r_0 \sim r$ of messages that are received immediately after being created and the other $1 - r_0$ arrival times that are still roughly gaussian.

This fact can be explained considering that if r is the fraction of supernodes and the message creation rate is not correlated with supernode status, the expected ratio of messages generated by supernodes is also r, and those messages have no lag.

As it can be seen in Figure 7.2, the mean value of OL_μ decreases with increasing r. Part of this is because of the $r_0 \sim r$ fraction of nodes with $\mathrm{OL}_\mu(M) = 0$, part is because of the mean of the rest of the Gaussian-like

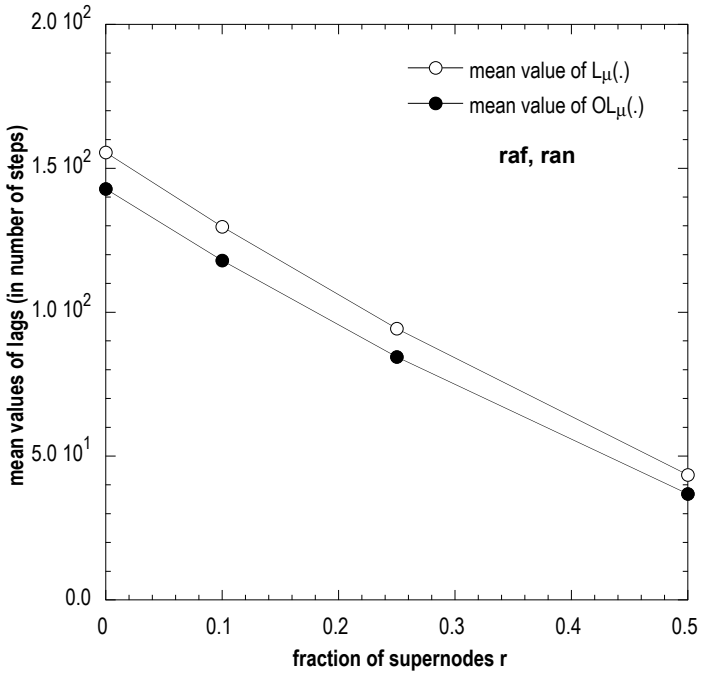

Figure 7.2: Mean values $\langle L_\mu \rangle$ and $\langle \mathrm{OL}_\mu \rangle$ of L_μ and OL_μ as functions of the supernode fraction r, calculated by using the RAF model and the "ran" status assignment to supernodes.

component in Figure 7.1 shifts leftwards for progressively high values of r.

As we can see, a system where only the original post author shares a resource, like in the RAF model, would not provide a satisfactory experience. The mean values for L_μ, $L_{\max}$, OL_μ and $\mathrm{OL}_{\max}$ are reported in Table 7.1. Even with extreme and unrealistic values of supernodes, under the 1 step = 15 minutes conversion, a user would on average see the messages of his friends after a whole day, and, for some messages, he could have to wait as long as almost 3 days.

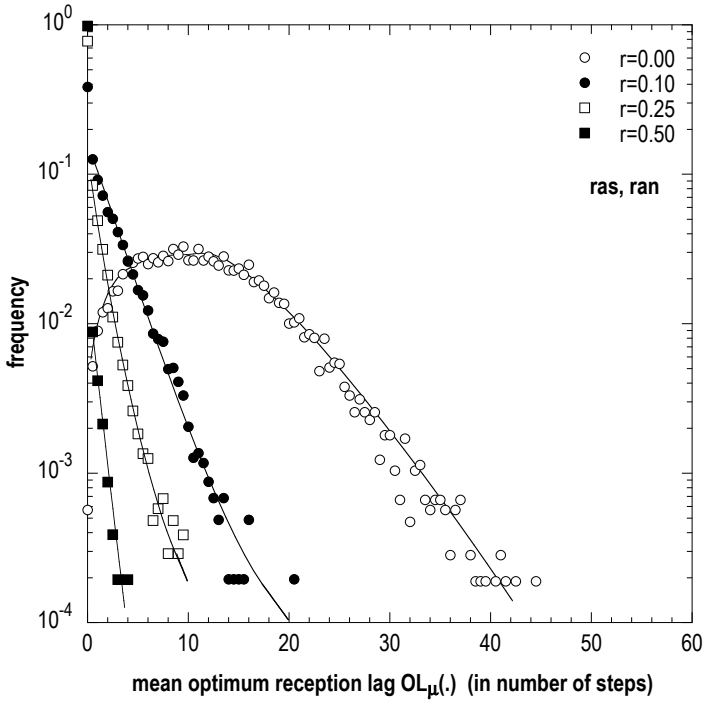

Figure 7.3: Distribution of the mean optimum reception lag OL_μ for different values of the supernode fraction r, calculated by using the RAS model and the "ran" status assignment to supernodes. See text. The solid lines are guides for the eye.

RAS model

In Figure 7.3 we report the degree distribution of the mean optimum lag values for simulations of the RAS model and random supernode status assignment ("ran").

In the RAS model, the impact of the peers resharing the resources they acquired is enormous: even with $r = 0$, the mean of the mean of $\langle \mathrm{OL}_\mu \rangle$ drops from the ~ 142 steps of the RAF model to only 12, i.e., 4 hours considering 1 step = 15 minutes.

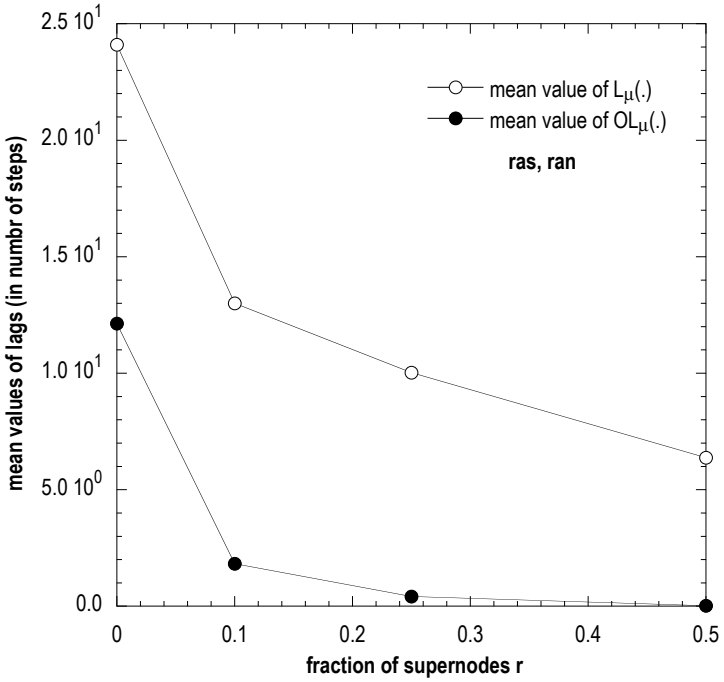

Figure 7.4: Mean values $\langle L_\mu \rangle$ and $\langle \mathrm{OL}_\mu \rangle$ of L_μ and OL_μ as functions of the supernode fraction r, calculated by using the RAS model and the "ran" status assignment to supernodes.

When we consider positive values of r, even a moderate ratio of supernodes ($r = 0.1$) makes $\langle \mathrm{OL}_\mu \rangle$ drop by an order of magnitude. In Figure 7.4 the effect of r on the system is represented concisely studying $\langle \mathrm{OL}_\mu \rangle$: each time we increase r by a factor of 2, $\langle \mathrm{OL}_\mu \rangle$ decreases of an order of magnitude.

Moreover, already at $r = 0.25$ the system performance is near optimal, as users receive 90% of the messages in one step or less. We consider this kind of performance highly satisfactory.

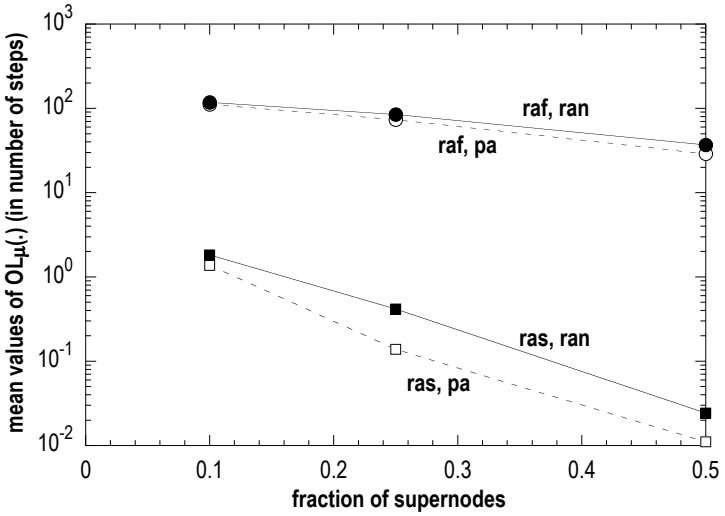

Figure 7.5: Mean values $\langle \mathrm{OL}_\mu \rangle$ of OL_μ as functions of the supernode fraction r, calculated by using the RAS and RAF models and the "ran" and "pa" status assignment to supernodes. The tags "ras" and "raf" refer to the RAS and RAF models, respectively.

The effects of preferential assignment of supernode status

In Section 7.1 we mentioned that we could allocate supernodes with two different strategies:

- uniformly choosing the nodes (random assignment, "ran");
- choosing the nodes with a probability proportional to their degree (preferential assignment, "pa").

The simulation results presented up to this point used models that chose uniformly the supernodes ("ran"). In this Subsection, we evaluate the impact of the preferential assignment strategy ("pa").

In Figure 7.5 and Table 7.2 we gathered the results of simulations of models that use both the random assignment (ran) and the preferential assignment (pa) approaches. We do not report values for $r = 0$ as in this case there would

	RAF $\langle \mathrm{OL}_\mu \rangle$			RAS $\langle \mathrm{OL}_\mu \rangle$		
r	ran	pa	pa/ran	ran	pa	pa/ran
0.1	117.9	111.1	0.94	1.82	1.37	0.75
0.25	84.43	73.54	0.87	0.42	0.14	0.33
0.5	36.85	28.86	0.78	0.02	0.01	0.45

Table 7.2: Mean values $\langle \mathrm{OL}_\mu \rangle$ of OL_μ as functions of the supernode fraction r, calculated by using the RAS and RAF models and the "ran" and "pa" status assignment to supernodes. "pa/ran" are the ratios between $\langle \mathrm{OL}_\mu \rangle$ calculated using the "pa" and "ran" statuses, respectively, for both the RAF and RAS models.

be no supernodes. In the case of the RAF model, the effects of "pa" are almost negligible for most values of r considered, and they start being noticeable only at relatively high values of r.

On the other hand, in the RAS model, the improvements are already relevant for $r = 0.1$ and become massive for $r = 0.25$. These results are not unexpected, as supernodes in RAS models are really the pillars of information sharing, and allocating the status to nodes with high degree greatly improves the number of messages they can share.

If in the real system there is correlation between the degree of the nodes and the supernode status, as we deem plausible and likely, our model predicts that with 10% of nodes acting as supernodes, almost 70% of the messages are received in 1 step or less and more than 80% in 2 steps or less. With 25% of supernodes, 90% of the messages are received istantaneously (0 steps) and 97% in 1 step or less. Such results are extremely satisfactory.

The effects of different network topologies

All the simulations performed so far used graphs generated by the Holme and Kim algorithm [93], with the following parameters: 1000 nodes, 50 starting edges per node, 0.2 probability to add a triadic closure step. These networks have a high clustering coefficient and a fat tailed degree distribution, i.e., they have a relatively high chance of having several nodes with an extremely high

Algorithm	$\langle k \rangle$	C
HK(1000, 50, 0.2)	93.5	0.20
BA(1000, 50)	95.0	0.17
WS(1000, 95, 0.41)	94.0	0.20
ER(1000, 0.095)	94.7	0.095

Table 7.3: Expected values of average degree $\langle k \rangle$ and clustering coefficient C for networks generated according to the HK, BA, WS and ER algorithms. The starting parameters are choosen so that the networks generated have the same number of nodes and the same expected number of edges (47000). Averages over 100 runs.

degree.

In order to understand how the topological features of the starting network affect the simulation, we performed simulations with starting networks generated according to different models. To assess the individual importance of fat tailed degree distributions, one set of simulations starts with networks generated according to the BA model with 1000 nodes and 50 starting edges per nodes. The Holme and Kim (HK) model was derived by the BA model by only adding the triadic closure step.

The BA model is known to yield networks with a relatively low clustering coefficient, as discussed in Chapter 2. In particular, with the theoretical analysis of the BA model at the thermodynamic limit, the expected clustering coefficient is only 0.005. In fact, experience shows that for relatively small networks such as the ones we are discussing, it is much better to use simulations to find out the expected clustering coefficient, that we measured as 0.17 (mean of 100 simulations). This value, although smaller than that of networks generated by the HK model, is nonetheless rather high.

In order to assess the importance of the fat tail that is typical of both the BA and the HK models, we also performed simulations starting with networks generated according to the WS model. Such networks have a high clustering coefficient, but have a significantly lower ratio of very high degree nodes compared to the networks generated with either the HK or the BA models. We

model	strategy	r	HK	BA	WS	ER
RAF	pa	0	143.0	142.5	143.0	143.4
		0.1	111.0	112.3	115.4	115.9
		0.25	73.54	73.38	82.47	80.89
		0.5	28.86	29.41	35.98	35.40
	ran	0	142.8	144.0	144.0	142.5
		0.1	117.9	112.9	122.7	117.5
		0.25	84.43	79.4	76.6	84.12
		0.5	36.85	32.4	30.7	37.10
RAS	pa	0	12.3	20.2	16.8	26.2
		0.1	1.37	4.83	5.29	6.85
		0.25	0.13	1.01	1.60	1.31
		0.5	0.01	0.21	0.20	0.09
	ran	0	12.1	19.7	16.9	25.9
		0.1	1.82	6.94	5.83	7.84
		0.25	0.41	2.09	1.54	1.68
		0.5	0.02	0.28	0.26	0.10

Table 7.4: Mean values $\langle \mathrm{OL}_\mu \rangle$ of optimum reception lags OL_μ resulting from simulations carried out using the HK, BA, WS and ER algorithms, with the RAF and RAS models, with the "pa" and "ran" strategies and with different values of the supernode fraction r.

chose the initial parameters for the WS model in order yield networks that have the same number of nodes and roughly the same number of edges of the networks generated with the other algorithms.

Eventually, as a control test, we performed simulations starting with networks generated according to the Erdős-Rényi (ER) random graph model $GNP(1000, 0.095)$ [57]. As can be seen from Table 7.3, such networks have the same expected average degree $\langle k \rangle$ as that of the networks generated with the WS model. Moreover, random graphs do not have a fat tail, but, as opposed to the high clustering coefficient typical of the WS networks, they have a very low clustering coefficient.

In Table 7.4 and Figure 7.6 we gathered all the results for our simulations carried out using the HK, BA, WS and ER algorithms, with the RAF and RAS models, with the "pa" and "ran" strategies and with different values of the supernode fraction r. Results concerning the RAF model, both using the

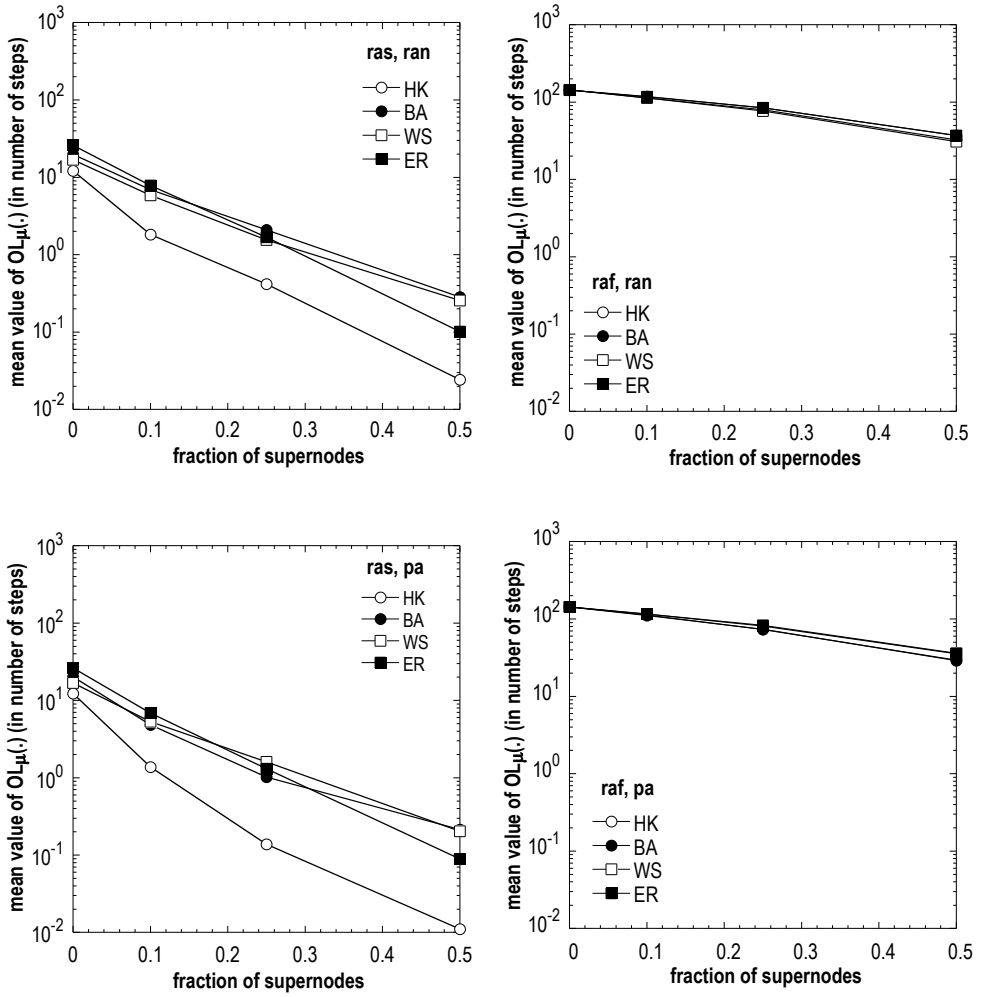

Figure 7.6: Mean values $\langle \mathrm{OL}_\mu \rangle$ of optimum reception lags OL_μ resulting from simulations carried out using the HK, BA, WS and ER algorithms, with the RAF and RAF models, with the "pa" and "ran" strategies as functions of the supernode fraction r.

"ran" or the "pa" strategy, are significantly similar regardless of the network.

On the other hand, with the RAS model several differences show up. First, we notice how simulations starting with HK networks give more favorable results than those starting with BA networks. Moreover, increasing r has a more positive effect on simulations starting with HK networks than on simulations starting with BA networks. This fact suggests that an even marginally higher clustering coefficient is an important factor.

However, clustering coefficient alone is not the only topological feature that has a positive effect on the simulation results. Given our parameter choices, the networks generated with the WS algorithm have an expected clustering coefficient similar to those generated with the HK algorithm. Nonetheless, simulations of the RAS model starting with WS networks give less favorable results than those starting with HK networks; for example, with $r = 0.1$ and "pa", the average optimum lag takes almost 4 times longer in the WS case.

Eventually, we compared the results of simulations starting with WS networks to those of simulations starting with a random graph with similar average node degree, and we found out that systems starting with random graphs perform slightly worse than those starting with BA and WS networks for small positive values of r.

Our opinion is that although a higher clustering coefficient greatly enhances performance when the starting network has a fat tailed degree distribution, i.e., when there is a relatively high chance to have nodes with extremely high degree, the presence of such nodes is just as important. In fact, the two features, when taken separately, do not yield huge benefits over completely unstructured networks such as random graphs. On the other hand, their combination, which occurs in HK networks guarantees a much better overall performance.

Chapter 8

Conclusions

With the pervasive diffusion of social networking systems in everybody's daily lives, social scientists have the unprecedented possibility to study quantitatively large social systems. Moreover, since the social networking revolution crossed age, gender and nationality boundaries, insight on some real life behaviors can be obtained from the analysis of the online structures.

However, researchers still lack effective ways to perform proper experiments, because the virtual nature of online social networks does not remove ethical issues in interfering with the systems. As a consequence, modeling and simulation in general, and, specifically, agent-based modeling and simulation are going to have an increasing relevance.

Considering (*i*) the lack of specific support for social networks in agent-based simulation toolkits, and (*ii*) the respective advantages of the highly specialized agent-based models and the more general traditional ones, we developed a unified conceptual framework for agent-based modeling and simulation of social networks.

First, we singled out a meta-model from the critical analysis of the several agent-based and traditional models developed in the literature. Such meta-model can be used:

– to convert traditional models into agent-based, concurrent models,

- to merge in the same model some aspects of traditional stochastic models and more pro-active elements, typical of agent-based modeling, and
- to express full-fledged agent-based models.

As a second step, we developed a domain-specific language that allows to express models conforming to our meta-model in an executable way. Our language is implemented as an internal domain-specific language embedded in a high-level general-purpose language. This approach allows us to provide a very expressive language that can be used productively by researchers with a strong background in computing, but that does not forgo simplicity, in order to be profitably used by researchers with different backgrounds.

Moreover, with our approach, it is extremely simple to call external libraries. Several high quality scientific and statistical libraries are available, which, in turn, are built over highly optimized C and Fortran code.

In the third step, for executing the models written in our domain-specific language, we developed a software framework that provides runtime support for the simulations. The runtime system is an effective agent-based platform that is used both (*i*) to run the simulation units which are part of the model, and (*ii*) to implement the various software components that support the simulation execution.

The social network simulation framework we propose has the following defining features: (*i*) it supports both small and large networks; (*ii*) simulations can be effortlessly run on remote machines; (*iii*) it is easy to use, even for people without a strong programming background; (*iv*) deploying a large simulation is not significantly harder than deploying a small one.

The system is highly configurable, so that it offers the possibility to choose the most appropriate components for each model, considering the specific algorithmic requirements. For example, the underlying network representation can be fine-tuned both (*i*) in terms of data structures (e.g., sparse adjacency matrices or adjacency lists), and (*ii*) from the point of view of data storage (e.g., RAM-based, file-based, or interfacing one of the several DBMS that we support).

Moreover, several strategies are used to minimize the memory footprint of the simulations. In general, our design leans the memory vs. CPU trade-off towards memory, because our experience showed that memory is a much more critical resource when performing large scale simulations.

Then, we validated our approach adapting several traditional models to our meta-model and implementing them in the domain-specific language. We carefully compared the results of the simulations with the theoretical features studied in the literature. Our results show that our approach is successful in providing a friendly and easy environment to perform agent-based simulations over social networks; such simulations are of interest both to develop models and to study the results of the models themselves. We remind here that agent-based simulations are powerful conceptual modeling tools for social network simulations, which have relevant advantages over the traditional ones.

Eventually, considering the satisfactory results we obtained, we applied our framework to the still open problem of creating an entirely distributed social networking system, which, as compared to the centralized ones, yields relevant advantages as far as privacy and resilience are concerned. We developed several models to help us in the understanding of the many issues that a P2P social networking system would have when deployed, and specifically the well-known issue of the availability of rare resources. Through simulation, we found some criteria for the design of distributed social networks and some operation conditions which may result in a satisfactory user experience in terms of reduced delays in the propagation of information. Consequently, we are developing a distributed social networking system optimized by means of the results of our simulations.

Appendix A

Acronyms

ABM Agent-based Modeling
ASPL Average Shortest Path Length
ACL Agent Communication Language
BA Barabàsi-Albert
BFS Breadth First Search
BPA Biased Preferential Attachment
CNT Complex Network Theory
CPL Characteristic Path Length
DHT Distributed Hash Table
DSL Domain-Specific Language
ER Erdős-Rényi
HK Holme and Kim
MAS Multi-agent System
OSN Online Social Network
P2P Peer-to-Peer
SIR Susceptible-Infected-Recovered
SNA Social Network Analysis
TL Transitive Linking
WS Watts-Strogatz

Bibliography

[1] L. Adamic. Friends and neighbors on the web. *Social Networks*, 25(3):211–230, 2003.

[2] L. Adamic and E. Adar. How to search a social network. *Social Networks*, 27(3):187–203, 2005.

[3] G. Agha. *Actors: a model of concurrent computation in distributed systems*. MIT Press, Cambridge, MA, USA, 1986.

[4] Y. Y. Ahn, S. Han, H. Kwak, S. Moon, and H. Jeong. Analysis of topological characteristics of huge online social networking services. In *Proceedings of the 16th International Conference on World Wide Web*, pages 835–844, 2007.

[5] L. M. Aiello and G. Ruffo. Lotusnet: Tunable privacy for distributed online social network services. *Computer Communications*, 35(1):75–88, 2012.

[6] R. Albert and A. L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47–74, 2002.

[7] K. Anagnostakis, F. Harmantzis, S. Ioannidis, and M. Zghaibeh. On the impact of practical p2p incentive mechanisms on user behavior. Technical Report 06-14, NET Institute, 2006.

[8] W. B. Arthur, B. LeBaron, R. Palmer, and P. Tayler. Asset pricing under endogenous expectations in an artificial stock market. In W. B.

Arthur, S. Durlauf, and D. Lane, editors, *The Economy as a Complex Evolving System II*. Addison-Wesley, Menlo Park, 1997.

[9] J. L. Austin, J. O. Urmson, and M. Sbisà. *How to Do Things with Words: The William James Lectures Delivered at Harvard University in 1955*. Harvard University. Clarendon Press, 1975.

[10] R. Axelrod. Advancing the art of simulation in the social sciences. *Complexity*, 3(2):16–22, 1997.

[11] R. Axelrod. Advancing the art of simulation in the social sciences. *Japanese Journal for Management Information System*, 12(3):1–19, 2003.

[12] R. Axelrod. Simulation in social sciences. In J. Rennard, editor, *Handbook of Research on Nature-Inspired Computing for Economics and Management*, pages 90 – 100. IGI Global, Hershey, PA, USA, 2007.

[13] R. Axelrod and L. Tesfatsion. A guide for newcomers to agent-based modeling in the social sciences. In L. Tesfatsion and K. L. Judd, editors, *Handbook of Computational Economics*, volume 2, pages 1647–1659. Elsevier, 2006.

[14] R. Axtel, J. M. Epstein, and H. P. Young. Social dynamics. In *Social Dynamics*, Economic Learning and Social Evolution Series. Mit Press, 2004.

[15] R. Bagni, R. Berchi, and P. Cariello. A comparison of simulation models applied to epidemics. *Journal of Artificial Societies and Social Simulation*, 5(3), 2002. `http://jasss.soc.surrey.ac.uk/5/3/5.html`.

[16] A. L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[17] M. Basuga, R. Belavic, A. Slipcevic, V. Podobnik, A. Petric, and I. Lovrek. The magnet: Agent-based middleware enabling social networking for mobile users. In *10th International Conference on Telecommunications*, pages 89–96, 2009.

[18] J. Bearden, W. Atwood, P. Freitag, C. Hendricks, B. Mintz, and M. Schwartz. The nature and extent of bank centrality in corporate networks. In J. Scott, editor, *Social Networks*, volume 3. Sage, London, UK, 1975.

[19] F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa. Jade: A software framework for developing multi-agent applications. lessons learned. *Information and Software Technology*, 50(1-2):10–21, 2008.

[20] F. Bellifemine, A. Poggi, and G. Rimassa. JADE: a FIPA2000 compliant agent development environment. In *Proceedings of the 5th International Conference on Autonomous agents*, pages 216–217. ACM, 2001.

[21] F. Bergenti, E. Franchi, and A. Poggi. Using HDS for realizing multiagent applications. In *Proceedings of the Third International Workshop on LAnguages, Methodologies and Development Tools for Multi-Agent Systems (LADS 2010)*, pages 62–68, Lyon, France, 2010.

[22] F. Bergenti, E. Franchi, and A. Poggi. Agent-based social networks for enterprise collaboration. In *20th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 25–28. IEEE, 2011.

[23] F. Bergenti, E. Franchi, and A. Poggi. Selected models for agent-based simulation of social networks. In D. Kazakov and G. Tsoulas, editors, *Proceedings of the 3rd Symposium on Social Networks and Multiagent Systems (SNAMAS '11)*, pages 27–32, York, 2011. Society for the Study of Artificial Intelligence and the Simulation of Behaviour.

[24] F. Bergenti, E. Franchi, and A. Poggi. Enhancing social networks with agent and semantic web technologies. In S. Brüggemann and C. D'Amato, editors, *Collaboration and the Semantic Web: Social Networks, Knowledge Networks, and Knowledge Resources*, pages 83–100. IGI Global, 2012.

[25] F. Bergenti, E. Franchi, and A. Poggi. Agent-based interpretation of classic network models. *Computational and Mathematical Organization*

Theory, 2013. In press and avaiable online DOI: 10.1007/s10588-012-9150-x.

[26] L. Bergmans and M. Aksit. Composing crosscutting concerns using composition filters. *Communications of the ACM*, 44(10):51–57, 2001.

[27] B. Bollobás and F. R. K. Chung. The diameter of a cycle plus a random matching. *SIAM Journal on discrete mathematics*, 1(3):328–333, 1988.

[28] R. H. Bordini, J. F. Hübner, and M. Wooldridge. *Programming multi-agent systems in AgentSpeak using Jason*, volume 8. Wiley-Interscience, 2007.

[29] D. M. Boyd and N. B. Ellison. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1):210–230, 2008.

[30] L. Braubach, A. Pokahr, and W. Lamersdorf. A universal criteria catalog for evaluation of heterogeneous agent development artifacts. In *Proc. of the Sixth Int. Workshop "From Agent Theory to Agent Implementation" (AT2AI-6)*, pages 19–28, Estoril, Portugal, 2008.

[31] D. Brickley and L. Miller. FOAF vocabulary specification. `http://xmlns.com/foaf/0.1/`, 2005.

[32] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986.

[33] S. Buchegger and A. Datta. A case for p2p infrastructure for social networks - opportunities and challenges. In *Proceedings of Sixth International Conference on Wireless On-Demand Network Systems and Services (WONS 2009)*, pages 161 –168, Snowbird, UT, USA, 2009.

[34] S. Buchegger, D. Schiöberg, L. H. Vu, and A. Datta. Peerson: P2p social networking: early experiences and insights. In *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, pages 46–52, Nuremberg, Germany, 2009. ACM.

[35] K. M. Carley, D. B. Fridsma, E. Casman, A. Yahja, N. Altman, L.-C. C., B. Kaminsky, and D. Nave. Biowar: scalable agent-based model of bioattacks. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 36(2):252 – 265, 2006.

[36] A. Ceranowicz, P. E. Nielsen, and F. Koss. Behavioral representation in JSAF. In *Proceedings of Ninth Annual Computer Generated Forces and Behavior Representation Conference*, 2000.

[37] A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Review*, 51(4):661–703, 2009.

[38] B. Cohen. The bittorrent protocol specification. `http://www.bittorrent.org/beps/bep_0003.html`, 2008.

[39] P. R. Cohen and H. J. Levesque. Rational interaction as the basis for communication. In P. R. Cohen, J. L. Morgan, and M. E. Pollack, editors, *Intentions in Communication*, pages 221–256. MIT Press, 1990.

[40] P. R. Cohen and C. R. Perrault. Elements of a plan-based theory of speech acts. *Cognitive Science*, 3(3):177–212, 1979.

[41] R. Cohen and S. Havlin. Scale-free networks are ultrasmall. *Phys. Rev. Lett.*, 90(5):058701, 2003.

[42] L. A. Cutillo, R. Molva, and T. Strufe. Safebook: A privacy-preserving online social network leveraging on real-life trust. *Communications Magazine*, 47(12):94–101, 2009.

[43] J. Davidsen, H. Ebel, and S. Bornholdt. Emergence of a small world from local interactions: Modeling acquaintance networks. *Physical Review Letters*, 88(12):1–4, 2002.

[44] A. L. De Moura and R. Ierusalimschy. Revisiting coroutines. *ACM Transactions on Programming Languages and Systems*, 31(2):6:1–6:31, 2009.

[45] J. S. Dean, G. J. Gumerman, J. M. Epstein, R. L. Axtel, A. C. Swedlund, M. T. Parker, and S. McCarrol. Understanding Anasazi culture change

through agent-based modeling. In *Dynamics in Human and Primate Societies: Agent-Based Modeling of Social and Spatial Processes* [108].

[46] S. A. DeLoach. Moving multi-agent systems from research to practice. *International Journal of Agent-Oriented Software Engineering*, 3(4):378–382, 2009.

[47] P. Doreian. Actor network utilities and network evolution. *Social Networks*, 28(2):137–164, 2006.

[48] P. Doreian. A note on actor network utilities and network evolution. *Social Networks*, 30(1):104 – 106, 2008.

[49] A. Drogoul, D. Vanbergue, and T. Meurisse. Multi-agent based simulation: Where are the agents? In J. Sichman, F. Bousquet, and P. Davidsson, editors, *Multi-Agent-Based Simulation II*, pages 43 – 49. Springer Berlin / Heidelberg, 2003.

[50] P. Duchon, N. Hanusse, E. Lebhar, and N. Schabanel. Could any graph be turned into a small-world? *Theoretical Computer Science*, 355(1):96–103, 2006.

[51] R. I. M. Dunbar. Neocortex size as a constraint on group size in primates. *Journal of Human Evolution*, 22(6):469–493, 1992.

[52] P. W. Dymond and S. A. Cook. Hardware complexity and parallel computation. In *IEEE Annual Symposium on Foundations of Computer Science*, pages 360–372. IEEE Computer Society, 1980.

[53] J. M. Epstein. Agent-based computational models and generative social science. *Complexity*, 4(5):41–60, 1999.

[54] J. M. Epstein. Modeling civil violence: An agent-based computational approach. *Proceedings of the National Academy of Sciences of the United States of America*, 99(Suppl 3):7243–7250, 2002.

[55] J. M. Epstein. *Generative social science: studies in agent-based computational modeling*. Princeton studies in complexity. Princeton University Press, 2006.

[56] J. M. Epstein and R. Axtell. *Growing Artificial Societies: Social Science from the Bottom Up.* Complex Adaptive Systems Series. Mit Press, 1996.

[57] P. Erdős and A. Rényi. On random graphs. *Publicationes Mathematicae*, 6(26):290–297, 1959.

[58] L. Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, 8:128–140, 1741.

[59] Facebook. Facebook statement of rights and responsibilities. `http://www.facebook.com/terms.php`, 2011.

[60] T. Finin, R. Fritzson, D. McKay, and R. McEntire. Kqml – a language and protocol for knowledge and information exchange. In *Proceedings of the 13th Intl. Distributed Artificial Intelligence Workshop*, pages 127–136, 1994.

[61] C. S. Fisher. *Networks and places: social relations in the urban setting.* Free Press, New York, 1977.

[62] V. Folcik, G. An, and C. Orosz. The basic immune simulator: An agent-based model to study the interactions between innate and adaptive immunity. *Theoretical Biology and Medical Modelling*, 4(1):39, 2007.

[63] L. Foner. A multi-agent referral system for matchmaking. In *Proceedings the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM '96)*, pages 22–24, 1996.

[64] L. Foner. A security architecture for multi-agent matchmaking. In *Proceeding of Second International Conference on Multi-Agent System (ICMAS'96)*, pages 80–86. AAAI Press, 1996.

[65] L. N. Foner. Yenta: a multi-agent, referral-based matchmaking system. In *Proceedings of the first international conference on Autonomous agents*, AGENTS '97, pages 301–307, Marina del Rey, CA, 1997. ACM.

[66] Foundation for Intelligent Physical Agents. Fipa specifications. Available from: `http://www.fipa.org`, 2002.

[67] M. Fowler. *Domain-Specific Languages*. Addison-Wesley Signature Series (Fowler). Addison-Wesley Professional, 2010.

[68] E. Franchi. A multi-agent implementation of social networks. In *Proceedings of Undicesimo Workshop Nazionale "Dagli Oggetti agli Agenti" (WOA 2010)*, Rimini, 2010.

[69] E. Franchi. A domain specific language approach for agent-based social network modeling. In *Proceedings of the International Conference on Advances in Social Network Analysis and Mining (ASONAM 2012)*, Istanbul, Turkey, 2012. IEEE Computer Society.

[70] E. Franchi. Towards agent-based models for synthetic social network generation. In G. D. Putnik and M. M. Cruz-Cunha, editors, *Virtual and Networked Organizations, Emergent Technologies and Tools*, volume 248 of *Communications in Computer and Information Science*, pages 18–27. Springer Berlin Heidelberg, 2012.

[71] E. Franchi and A. Poggi. Multi-agent systems and social networks. In M. Cruz-Cunha, G. D. Putnik, N. Lopes, P. Gonçalves, and E. Miranda, editors, *Handbook of Research on Business Social Networking: Organizational, Managerial, and Technological Dimensions*, pages 84–98. IGI Global, Hershey, PA, 2011.

[72] E. Franchi, A. Poggi, and M. Tomaiuolo. Open social networking for online collaboration. *International Journal of e-Collaboration*, -. In press.

[73] E. Franchi, A. Poggi, and M. Tomaiuolo. Developing pervasive and adaptive applications with MAADE. *Journal of Computer Sciences and Applications*. In press.

[74] E. Franchi, A. Poggi, and M. Tomaiuolo. Developing applications with HDS. In *Proceedings of Dodicesimo Workshop Nazionale "Dagli Oggetti agli Agenti" (WOA 2011)*, 2011.

[75] E. Franchi and M. Tomaiuolo. Distributed social platforms for confidentiality and resilience. In L. Caviglione, M. Coccoli, and A. Merlo,

editors, *Social Network Engineering for Secure Web Data and Services*. IGI Global, Hershey, PA, -. In press.

[76] E. Franchi and M. Tomaiuolo. Software agents for distributed social networking. In *Tredicesimo Workshop Nazionale "Dagli Oggetti agli Agenti" (WOA 2012)*, Milan, Italy, 2012.

[77] L. C. Freeman. *The Development Of Social Network Analysis: A Study In The Sociology Of Science*, volume 1. Empirical Press, Vancouver, CA, 2004.

[78] R. M. Friborg, J. M. Bjørndalen, and B. Vinter. Three unique implementations of processes for PyCSP. In P. H. Welch, H. Roebbers, J. F. Broenink, F. R. M. Barnes, C. G. Ritson, A. T. Sampson, G. S. Stiles, and B. Vinter, editors, *Communicating Process Architectures 2009 - WoTUG-32*, volume 67 of *Concurrent Systems Engineering*, pages 277–293. IOS Press, 2009.

[79] M. R. Genesereth and S. P. Ketchpel. Software agents. *Communications of the ACM*, 37(7):48–53, 1994.

[80] D. Ghosh. *DSLs in Action*. Manning Publications, 2010.

[81] E. N. Gilbert. Random graphs. *Annals of Mathematical Statistics*, 30:1141–1144, 1959.

[82] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou. Walking in facebook: A case study of unbiased sampling of osns. In *Proceedings of IEEE INFOCOM '10*, San Diego, CA, 2010.

[83] K. Graffi, C. Gross, P. Mukherjee, A. Kovacevic, and R. Steinmetz. LifeSocial.KOM: A P2P-based platform for secure online social networks. In *Tenth International Conference on Peer-to-Peer Computing (P2P)*, pages 1–2, 2010.

[84] M. S. Granovetter. The strength of weak ties. *American Journal of Sociology*, 78(6):1360–1380, 1973.

[85] M. S. Granovetter. The strength of weak ties: A network theory revisited. *Sociological Theory*, 1:201, 1983.

[86] A. Gursel and S. Sen. Improving search in social networks by agent based mining. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 2034–2039, 2009.

[87] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using NetworkX. In G. Varoquaux, T. Vaught, and J. Millman, editors, *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Pasadena, CA, 2008.

[88] F. Haray. *Graph Theory*. Addison-Wesley, Reading, MA, 1969.

[89] F. Haray and R. Z. Norman. *Graoh Theory as a Mathematical Model in Social Science*. Institute for Social Research, Ann Arbor, MI, 1953.

[90] F. Haray, R. Z. Norman, and D. Cartwright. *Structural Models: An introduction to the theory of directed graphs*. John Wiley, New York, 1965.

[91] F. Hattori, T. Ohguro, M. Yokoo, and S. Matsubara. Socialware: Multiagent systems for supporting network communities. *Communications of the ACM*, 42(3):55–61, 1999.

[92] R. Hill, R. Carl, and L. Champagne. Using agent simulation models to examine and investigate search theory against a historical case study. *Journal of Simulation*, 1(1):29–38, 2006.

[93] P. Holme and B. Kim. Growing scale-free networks with tunable clustering. *Physical Review E*, 65(2):2–5, 2002.

[94] A. Howard. *Data for the Public Good*. O'Reilly Media, 2012.

[95] N. Hummon. Utility and dynamic social networks. *Social Networks*, 22(3):221–249, 2000.

[96] T. Huynh, N. Jennings, and N. Shadbolt. An integrated trust and reputation model for open multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 13(2):119–154, 2006.

[97] A. Ilachinski. Irreducible semi-autonomous adaptive combat (ISAAC): An artificial-life approach to land combat. *Military Operations Research*, 5(3):29–46, 2000.

[98] M. O. Jackson. *Social and Economic Networks*. Princeton University Press. Princeton University Press, 2010.

[99] M. O. Jackson and A. Wolinsky. A strategic model of social and economic networks. *Journal of economic theory*, 71:44–74, 1996.

[100] A. Java, X. Song, and T. Finin. Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pages 1–10, San Jose, California , USA, 2007.

[101] H. Kautz and B. Selman. Referral web: combining social networks and collaborative filtering. *Communications of the ACM*, 40(3):63–65, 1997.

[102] H. Kautz, B. Selman, and M. Shah. The hidden web. *AI magazine*, 18(2):27–36, 1997.

[103] W. O. Kermack and A. G. McKendrick. A contribution to the mathematical theory of epidemics. *Proc. Roy. Soc. Lond. A*, 115:700–721, 1927.

[104] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[105] J. Kleinberg. Navigation in a small world. *Nature*, 406(6798):845–845, 2000.

[106] J. Kleinberg. The small-world phenomenon: an algorithm perspective. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*, pages 163–170, 2000.

[107] J. Kleinberg. Complex networks and decentralized search algorithms. In *Proceedings of the International Congress of Mathematicians (ICM)*, volume 3, pages 1–26, Madrid, Spain, 2006.

[108] T. A. Kohler and G. J. Gumerman. *Dynamics in Human and Primate Societies: Agent-Based Modeling of Social and Spatial Processes.* Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press, 2000.

[109] T. A. Kohler, G. J. Gumerman, and R. J. Reynolds. Simulating ancient societies. *Scientific American*, 293:76 – 84, 2005.

[110] D. König. *Theorie der endlichen und unendlichen Graphen.* Akademische Verlagsgesellschaft, Leipzig, DE, 1936.

[111] T. Kosar, P. E. Martínez López, P. A. Barrientos, and M. Mernik. A preliminary study on various implementation approaches of domain-specific language. *Information and Software Technology*, 50(5):390–405, 2008.

[112] R. Kumar, J. Novak, and A. Tomkins. Structure and evolution of online social networks. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '06*, Philadelphia, Pennsylvania, USA, 2006.

[113] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, pages 591–600, Raleigh, NC, USA, 2010. ACM.

[114] M. K. Lauren and R. T. Stephen. Map-aware non-uniform automata - a new zealand approach to scenario modelling. *Journal of Battlefield Technology*, 5(1):27ff., 2002.

[115] J. Leskovec, L. Backstrom, R. Kumar, and A. Tomkins. Microscopic evolution of social networks. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 462–470, Las Vegas, Nevada, USA, 2008.

[116] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceeding of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 177–187, Chicago, USA, 2005.

[117] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Statistical properties of community structure in large social and information networks. In *Proceeding of the 17th International Conference on World Wide Web*, pages 695–704, Beijing, China, 2008.

[118] K. Lewin. *Principles of Topological Psychology*. McGraw Hill, New York, 1936.

[119] K. Lewin. *Field Theory in the Social Sciences*. Harper, New York, 1951.

[120] X. Li, W. Mao, D. Zeng, and F. Wang. Agent-based social simulation and modeling in social computing. In C. Yang, H. Chen, M. Chau, K. Chang, S. Lang, P. Chen, R. Hsieh, D. Zeng, F. Wang, K. Carley, W. Mao, and J. Zhan, editors, *Intelligence and Security Informatics*, volume 5075 of *Lecture Notes in Computer Science*, pages 401 – 412. Springer Berlin / Heidelberg, 2008.

[121] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. Balan. Mason: A multiagent simulation environment. *Simulation*, 81(7):517–527, 2005.

[122] S. L. Lytinen and S. F. Railsback. The Evolution of Agent-based Simulation Platforms: A Review of NetLogo 5.0 and ReLogo. In *Proceedings of the Fourth International Symposium on Agent-Based Modeling and Simulation*, Vienna, Austria, 2012.

[123] M. W. Macy and R. Willer. Computational sociology and agent-based modeling. *Annual Review of Sociology*, 28:143–166, 2002.

[124] G. Madey, V. Freeh, and R. Tynan. Understanding oss as a self-organizing process. In *2nd Workshop on Open Source Software Engineering at the 24th International Conference on Software Engineering (ICSE2002)*, pages 130–131, Orlando, FL, 2002.

[125] V. Mařík and L. J. . Industrial applications of agent technologies. *Control Engineering Practice*, 15(11):1364–1380, 2007.

[126] J. McKean, H. Shorter, M. Luck, P. McBurney, and S. Willmott. Technology diffusion: analysing the diffusion of agent technologies. *Journal of Autonomous Agents and Multi-Agent Systems*, 17(3):372–396, 2008.

[127] M. Mernik, J. Heering, and A. M. Sloane. When and how to develop domain-specific languages. *ACM computing surveys (CSUR)*, 37(4):316–344, 2005.

[128] R. K. Merton. The matthew effect in science. *Science*, 169(3810):56–63, 1968.

[129] N. Minar, R. Burkhart, C. Langton, and M. Askenazi. The swarm simulation system: a toolkit for building multi-agent simulations. Technical report, Santa Fe Institute, Santa Fe, 1996.

[130] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, pages 29–42, San Diego, California, USA, 2007. ACM.

[131] J. Moffat, J. Smith, and S. Witty. Emergent behaviour: Theory and experimentation using the mana model. *Journal of Applied Mathematics and Decision Sciences*, 2006:13, 2006.

[132] R. Monson-Haefel. *J2Ee Web Services*. Building Distributed Applications. Addison-Wesley, 2004.

[133] J. L. Moreno. *Who shall survive?* Beacon Press, New York, 1934.

[134] M. R. Morris, J. Teevan, and K. Panovich. What do people ask their social networks, and why?: a survey study of status message and behavior. In *Proceedings of the 28th international conference on Human factors in computing systems*, pages 1739–1748, Atlanta, GA, USA, 2010. ACM.

[135] R. B. Myerson. *Game Theory: Analysis of Conflict*. Harvard University Press, 1997.

[136] A. Nazir, S. Raza, and C. N. Chuah. Unveiling facebook: a measurement study of social network based applications. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pages 43–56. ACM, 2008.

[137] M. E. J. Newman. Models of the small world. *Journal of Statistical Physics*, 101(3):819–841, 2000.

[138] M. E. J. Newman. Properties of highly clustered networks. *Phys. Rev. E*, 68:026121, 2003.

[139] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.

[140] M. E. J. Newman. *Networks: An Introduction*. Oxford University Press, USA, 2010.

[141] M. E. J. Newman and J. Park. Why social networks are different from other types of networks. *Physical Review E*, 68:1–8, 2003.

[142] M. E. J. Newman and D. J. Watts. Renormalization group analysis of the small-world network model. *Physics Letters A*, 263:341–346, 1999.

[143] V. Nguyen and C. Martel. Analyzing and characterizing small-world graphs. In *Proceedings of the 16th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 311–320. Society for Industrial and Applied Mathematics, 2005.

[144] M. North, N. Collier, and J. Vos. Experiences creating three implementations of the repast agent modeling toolkit. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 16(1):25, 2006.

[145] M. J. North, T. R. Howe, N. T. Collier, and J. R. Vos. A declarative model assembly infrastructure for verification and validation. In *Advancing Social Simulation: The First World Congress*, pages 129–140. Springer Japan, 2007.

[146] M. J. North, C. M. Macal, J. S. Aubin, P. Thimmapuram, M. Bragen, J. Hahn, J. Karr, N. Brigham, M. E. Lacy, and D. Hampton. Multiscale agent-based consumer market modeling. *Complexity*, 15(5):37–47, 2010.

[147] J. Odell, H. Van Dyke Parunak, M. Fleischer, and S. Brueckner. Modeling agents and their environment. In F. Giunchiglia, J. Odell, and

G. Weiss, editors, *Agent-Oriented Software Engineering III*, pages 16–31. Springer Berlin / Heidelberg, 2003.

[148] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, 1999.

[149] H. V. D. Parunak, P. Nielsen, S. Brueckner, and R. Alonso. Hybrid multi-agent systems: Integrating swarming and bdi agents. In *Proceedings of the 2006 Workshop on Engineering Self-Organizing Agents*, Hakodate, Japan., 2007.

[150] R. Pastor-Satorras and A. Vespignani. Epidemic spreading in scale-free networks. *Phys. Rev. Lett.*, 86:3200–3203, 2001.

[151] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausble Inference*. Morgan Kaufmann Publ., 1988.

[152] E. Pitt and K. McNiff. *Java.rmi: The Remote Method Invocation Guide*. Addison-Wesley, 2001.

[153] A. Poggi. HDS: a software framework for the realization of pervasive applications. *WSEAS Trans. on Computers*, 9(10):1149–1159, 2010.

[154] A. Poggi, P. Turci, and M. Tomaiuolo. Service composition in open agent societies. In *Proceedings of Quarto Workshop Nazionale "Dagli Oggetti agli Agenti" (WOA 2003)*, pages 92–99, 2003.

[155] A. Pokahr, L. Braubach, and W. Lamersdorf. Jadex: A BDI reasoning engine. In R. H. Bordini, M. Dastani, J. Dix, and A. Fallah Seghrouchni, editors, *Multi-Agent Programming*, volume 15 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, pages 149–174. Springer US, 2005.

[156] D. d. S. Price. A general theory of bibliometric and other cumulative advantage processes. *Journal of the American Society for Information Science*, 27(5):292–306, 1976.

[157] S. F. Railsback, S. L. Lytinen, and S. K. Jackson. Agent-based simulation platforms: Review and development recommendations. *Simulation*, 82(9):609–623, 2006.

[158] A. S. Rao and M. P. Georgeff. Modeling rational agents within a bdi-architecture. In M. N. Huhns and M. P. Singh, editors, *Readings in Agents*, pages 317–328. Morgan Kaufmann, 1998.

[159] K. Roebuck. *Social Network Analysis: High-Impact Strategies - What You Need to Know: Definitions, Adoptions, Impact, Benefits, Maturity, Vendors*. Emereo Pty Limited, 2011.

[160] M. Russell. *21 Recipes for Mining Twitter*. O'Reilly Media, 2011.

[161] M. Russell. *Mining the Social Web: Analyzing Data from Facebook, Twitter, LinkedIn, and Other Social Media Sites*. Head First Series. O'Reilly Media, Incorporated, 2011.

[162] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach (3rd Edition)*. Prentice Hall, 2009.

[163] J. Sabater and C. Sierra. Reputation and social network analysis in multi-agent systems. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems (AAMAS '02)*, Bologna, Italy, 2002. ACM Press.

[164] S. Scellato, C. Mascolo, M. Musolesi, and V. Latora. Distance matters: Geo-social metrics for online social networks. In *Proceedings of the 3rd conference on Online social networks*, pages 1–8. USENIX Association, 2010.

[165] J. Scott. *Corporations, classes, and capitalism*. Hutchinson, London, UK, 1st edition, 1979.

[166] J. Scott. Networks of corporate power: A comparative assessment. *Annual Review of Sociology*, 17:181–203, 1991.

[167] J. Scott. *Social Network Analisys: a handbook*. Sage, London, UK, 2nd edition, 2000.

[168] J. Scott and C. Griff. *Directors of Industry: The British Corporate Network, 1904-1976*. Polity Press, 1984.

[169] J. R. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, 1969.

[170] H. A. Simon. A behavioral model of rational choice. *The Quarterly Journal of Economics*, 69(1), 1955.

[171] H. A. Simon. On a class of skew distribution functions. *Biometrika*, 42(3-4):425–440, 1955.

[172] B. Skyrms and R. Pemantle. A dynamic model of social network formation. In T. Gross and H. Sayama, editors, *Adaptive Networks*, volume 51 of *Understanding Complex Systems*, pages 231–251. Springer Berlin / Heidelberg, 2009.

[173] T. A. B. Snijders. Statistical models for social networks. *Annual Review of Sociology*, 37(1):131–153, 2011.

[174] F. N. Stokman, R. Ziegler, and J. Scott, editors. *Networks of Corporate Power. A Comparative Analysis of Ten Countries*. Polity Press, Oxford, UK, 1985.

[175] D. Stroud. Social networking: An age-neutral commodity — Social networking becomes a mature web application. *Journal of Direct, Data and Digital Marketing Practice*, 9(3):278–292, 2008.

[176] J. J. Sylvester. Chemistry and algebra. *Nature*, 17:284, 1878.

[177] G. Szabó, M. Alava, and J. Kertész. Structural transitions in scale-free networks. *Physical Review E*, 67(5):1–5, 2003.

[178] J. C. Thiele, W. Kurth, and V. Grimm. Agent-based modelling: Tools for linking NetLogo and R. *Journal of Artificial Societies and Social Simulation*, 15(3):8, 2012.

[179] K. Thorup, T. Alerstam, M. Hake, and N. Kjellen. Bird orientation: compensation for wind drift in migrating raptors is age dependent. *Proc. R. Soc. Lond*, 270(Suppl 1):S8–S11, 2003.

[180] S. Tisue and U. Wilensky. NetLogo: A simple environment for modeling complexity. In *International Conference on Complex Systems*, pages 16–21, Boston, MA, USA, 2004.

[181] M. Tsvetovat and A. Kouznetsov. *Social Network Analysis for Startups: Finding connections on the social web.* O'Reilly Media, 2011.

[182] Twitter. Twitter terms of service. `http://twitter.com/tos`, 2011.

[183] Y. Upadrashta, J. Vassileva, and W. Grassmann. Social networks in peer-to-peer systems. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05)*, pages 1–9, Hawaii, 2005.

[184] F. Walter, S. Battiston, and F. Schweitzer. A model of a trust-based recommendation system on a social network. *Journal of Autonomous Agents and Multi-Agent Systems*, 16(1):57–74, 2008.

[185] S. Wasserman and K. Faust. *Social network analysis: methods and applications.* Cambridge University Press, Cambridge, UK, 1994.

[186] D. J. Watts. Networks, dynamics, and the small-world phenomenon. *American Journal of Sociology*, 105(2):493–527, 1999.

[187] D. J. Watts and S. Strogatz. Collective dynamics of small-world networks. *Nature*, 393(6684):440–442, 1998.

[188] B. Wellman. The community question: the intimate networks of east yorkers. *American Journal of Sociology*, 84:1201–1231, 1979.

[189] B. Wellman and S. D. Berkowitz. *Social Structures: A Network Approach.* Structural Analysis in the Social Sciences. Cambridge University Press, 1988.

[190] D. Weyns, A. Helleboogh, and T. Holvoet. How to get multi-agent systems accepted in industry? *International Journal of Agent-Oriented Software Engineering*, 3(4):383–390, 2009.

[191] R. J. Wilson. *Four Colors Suffice: How The Map Problem Was Solved.* Princeton Paperbacks. Princeton University Press, 2002.

[192] M. Winikoff. Jack™ intelligent agents: An industrial strength platform. In R. H. Bordini, M. Dastani, J. Dix, and A. Fallah Seghrouchni, editors, *Multiagent Systems, Artificial Societies, and Simulated Organizations*, volume 15, pages 175–193. Springer US, 2005.

[193] M. Wooldridge. *The Logical Modelling of Computational Multi-Agent Systems*. PhD thesis, Department of Computation, UMIST, Manchester, UK, 1992.

[194] M. Wooldridge. *An Introduction to MultiAgent Systems*. Wiley, 2009.

[195] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *The knowledge engineering review*, 10(02):115–152, 1995.

[196] F. Xie and W. Cui. Cost range and the stable network structures. *Social Networks*, 30(1):100 – 101, 2008.

[197] S. Yoshida, K. Kamei, T. Ohguro, and K. Kuwabara. Shine: a peer-to-peer based framework of network community support systems. *Computer Communications*, 26(11):1199–1209, 2003.

[198] B. Yu and M. Singh. A social mechanism of reputation management in electronic communities. In M. Klusch and L. Kerschberg, editors, *Cooperative Information Agents IV - The Future of Information Agents in Cyberspace*, volume 1860 of *Lecture Notes in Computer Science*, pages 355–393. Springer Berlin / Heidelberg, 2000.

[199] B. Yu and M. P. Singh. A multiagent referral system for expertise location. In *Working Notes of the AAAI Workshop on Intelligent Information Systems*, pages 66–69, 1999.

[200] B. Yu and M. P. Singh. An evidential model of distributed reputation management. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 294–301, Bologna, 2002. ACM.

[201] B. Yu, M. Venkatraman, and M. P. Singh. An adaptive social network for information access: Theoretical and experimental results. *Applied Artificial Intelligence*, 17(1):21 – 38, 2003.

[202] G. U. Yule. A mathematical theory of evolution, based on the conclusions of dr. j. c. willis, f. r. s. *Philosophical Transactions of the Royal Society of London. Series B, Containing Papers of a Biological Character*, 213(402-410):21–87, 1925.

[203] M. Zghaibeh and F. C. Harmantzis. Revisiting free riding and the Tit-for-Tat in BitTorrent: A measurement study. *Peer-to-Peer Networking and Applications*, 1(2):162–173, 2008.

[204] J. Zhang and M. Ackerman. Searching for expertise in social networks: a simulation of potential strategies. In *Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work*, pages 71–80, Sanibel Island, Florda, USA, 2005. ACM.

Acknowledgements

Stimulating discussions with Prof. A. Poggi are gratefully acknowledged.