Comput Math Organ Theory (2013) 19:105–127
DOI 10.1007/s10588-012-9150-x

SI: SNAMAS

Agent-based interpretations of classic network models

Federico Bergenti · Enrico Franchi · Agostino Poggi

Published online: 8 January 2013

Abstract The paper is aimed at developing agent-based variants of traditional network models that make full use of concurrency. First, we review some classic models of the static structure of complex networks with the objective of developing agent-based models suited for simulating a large-scale, technology-enabled social network. Secondly, we outline the basic properties that characterize such networks. Then, we briefly discuss some classic network models and the properties of the networks they generate. Finally, we discuss how such models can be converted into agent-based models (i) to be executed more easily in heavily concurrent environments and (ii) to serve as basic blocks for more complex agent-based models. We evidence that many implicit assumptions made by traditional models regarding their execution environment are too expensive or outright impossible to maintain in concurrent environments. Consequently, we present the concurrency issues resulting from the violation of such assumptions. Then, we experimentally show that, under reasonable hypothesis, the agent-based variants maintain the main features of the classic models, notwithstanding the change of environment. Eventually, we present a meta-model that we singled out from the individual classic models and that we used to simplify the agent-oriented conversion of the traditional models. Finally, we discuss the software tools that we built to run the agent-based simulations.

Keywords Agent-based modeling · Social networks · Synthetic social network generation

F. Bergenti
Dipartimento di Matematica, Università degli Studi di Parma, Parma, Italy
e-mail: federico.bergenti@unipr.it

E. Franchi (✉) · A. Poggi
Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Parma, Parma, Italy
e-mail: efranchi@ce.unipr.it

A. Poggi
e-mail: agostino.poggi@unipr.it

1 Introduction

The notable interest of today's research on social networks is largely justified by their adoption as a unifying metaphor in very relevant Web-centric services like Facebook and many others. The inherently large scale of such services calls for automated techniques capable of promoting their potentials to unforeseen levels in terms of offered functionality and performance. Such automated techniques are still far from real-world practice because the impact of a novel algorithm (e.g., a friendship-discovery algorithm) cannot be easily assessed. This is the reason why we need effective tools to study, experiment and validate innovative techniques capable of providing concrete evaluation on the net results of the introduction of a novel proposal into a social networking system. *Agent-based simulation* is very helpful to this respect because it provides solid approaches for testing new ideas *in silico* before trying to put them into practice.

In the last sixty years of research, several models have been proposed to explain (i) the formation or (ii) the evolution of networks. Such models have been developed and studied by researchers coming from many different areas, such as computer science, economics, natural sciences, 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 (2010) for the economic and game-theoretic point of view and to Snijders (2011) for a complete review of the state of art in the statistical models; instead, we focus on the methods and models originated in computer, natural and physical sciences. Although we believe that the techniques and ideas exposed henceforth may be profitably applied in other situations as well, we do not discuss the matter here.

One of the main features of the early network formation models is their simplicity. Such models are typically focused on reproducing only few macro-level features in a way that they essentially trade the capacity to faithfully reproduce all the features of given networks to better investigate the process that led to the appearance of the chosen features. In fact, these methods are deeply rooted in the modus operandi of physical sciences and extensively use ideas from *mean-field analysis* in order to derive general features from the simple interaction rules that constitute the models.

However, these models also present a striking affinity with agent-based models, in that, although the techniques used differ, the general idea of studying complex features emerging from individual interactions is in common. The difference was grasped by Epstein (1999) in the context of social sciences and is, in essence, the methodological difference between the *inductive* reasoning of the physical sciences and the *generative* point of view of agent-based models.

From a more practical point of view, early models made some very strong assumptions in order to be tractable with mathematical tools, e.g., (i) uniformity of behavior of the studied entities, which is an unsettling premise for studying populations of human beings; (ii) lack of memory and, (iii) consistency of the real systems with the models studied at the *thermodynamic limit*, which, in turn presumes that said systems have the required weak-convergence properties.

On the other hand, agent-based models make no such assumptions and are especially effective when it is necessary to implement diverse behaviors for the different

components in the system. This concept, applied to a synthetic social network generation process, means that it would be possible to have each node in the network act autonomously, creating or severing links according to its local state and not to fixed rules built in the system. When applied to generic processes on networks, it means that it is possible that each agent varies its behavior according to individuality or history; however, studying the system does not become more complex.

The idea is that mathematical models operate at system level, thus they cannot easily take into account the huge amount of data that online social networks make available, because such data only encompasses the individual interactions among the actors in the network. On the other hand, agent-based modeling can use such data with little pre-processing, essentially building the models bottom-up, from the micro-level data to the macro-level. Another possibility with agent-based modeling is taking advantage, without too many technicalities, of the expertise of social scientists to describe human actions and compare the global trends obtained from running the multi-agent system with those inferred from the real data. The resulting models do not need to be tractable from a mathematical point of view, their only requirement is to be computable. For these reasons, agent-based simulation is now a consolidated field of research, and likewise it is the application of its results to social networks (Franchi and Poggi 2011; Franchi 2012b; Bergenti et al. 2011, 2012).

Another strong point of agent-based models is their inherently distributed nature, that naturally leads to grid-based or cloud-based large scale simulations. However, the amount of concurrency implicit in agent-based models can introduce biases and artifacts in the results of the simulations. Limiting concurrency is possible only in determinate situation, because, for example, synchronizing processes on a cloud simulation has overwhelming costs. Moreover, the semantics of truly agent-based models simply assumes such concurrency and, even when agent-based modeling is simply used as a slightly "richer" discrete-event simulation, limiting concurrency can negatively affect performance.

Consequently, it becomes important to study the effects of concurrency and one of the best ways is converting well-known, widely studied traditional models to agent-based concurrent ones, with the goal to study the effects of concurrency in a familiar scenario and subsequently predict the effects in novel cases.

This paper is organized as follows: in next section we introduce some metrics that are generally applicable to all networks and that we will use to characterize social networks. Such metrics are very common in network theory and we review them just to provide a common notation and a precise understanding of concepts. In Sect. 3, we survey some of the most classic network models and we briefly present some results on the metrics that we introduced in the previous section. In Sect. 4, we present agent-based modeling with more accuracy and then, in Sects. 5 and 6, we analyze the concurrency issues that arise when performing simulations in distributed scenarios. In Sect. 7 we experimentally investigate the issues presented in the latter two sections. In Sect. 8, we describe the meta-model we devised to more easily translate traditional models into agent-based ones and, eventually, in Sect. 9 we present two different software tools built around the ideas presented in Sect. 8. Finally, in Sect. 10 we draw some conclusions and sketch a possible line of work.

2 Networks metrics and measures

In this section we briefly review some classic metrics which give insight on the network structure. The metrics we have chosen are some of the most widely studied ones in network analysis and together they give a rough idea of the structure of a network; we also outline some correlation in the chosen metrics. In this paper we do not deal with advanced analysis techniques, such as community or cluster detection, since most of the papers we discuss in the following sections do not deal with them as well and consequently the results could not be compared.

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 network, k_u is the total number of edges of u.

With $\langle\cdot\rangle$ we refer to the expected value/mean 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.

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 logarithmically from the number of nodes in the network. 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 $G' = (V, E \setminus \{e\})$; 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 (Watts and Strogatz 1998). A different and non equivalent definition is given by Newman (2003a), 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. We say that a network has a power-law degree distribution with exponent γ if $p_k = \zeta(k)^{-1}k^{-\gamma}$. γ is also called the *scaling exponent*. Since networks are finite, most power-law degree distributions are actually power-laws with cutoff: $p_k \propto k^{-\gamma} \cdot e^{-k\gamma}$.

3 Models for simulated networks

In this section we consider some models to generate random graphs which have been proposed to simulate social networks.

The first and still most studied model of random graphs is the *Erdős-Rényi model* (ER) $G(n, p)$ (Erdős and Rényi 1959; Newman 2003b). $G(n, p)$ 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, (i) the expected value of the number of edges is $\langle m \rangle = \binom{n}{2} p$; (ii) the expected mean degree is $\langle k \rangle = (n-1)p$; (iii) the expected diameter is $\log n$; (iv) the degree distribution tends to a Poisson distribution for large n; (v) the clustering coefficient is given by $C = \langle k \rangle / (n-1)$.

Another very important model is the *Watts-Strogatz model* (WS) introduced by Watts and Strogatz (1998). The model starts with a closed linear structure where each node is connected with κ neighbors 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 by Bollobás and Chung (1988), Newman and Watts (1999), and Newman (2000). 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 of long-range links that each node gains is $p\kappa$ and the degree distribution is a Poisson distribution with mean $p\kappa$. For large networks, the average shortest path length is logarithmic with the size of the network and the clustering coefficient $C \to 3/4$.

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 nodes and links are added or removed according to some rules that vary from model to model. A famous model of this family is the *Barabási-Albert model* (BA) described by Barabási and Albert (1999). The BA model starts with m_0 nodes and no edges. At each step a new node with $m \leq m_0$ random links is added. The m links are directed towards nodes with probability proportional to their degree. The process stops after $n - m_0$ steps and yield a network with n nodes.

The BA model generates 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. Cohen and Havlin (2003) 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.

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 is known (Albert and Barabási 2002).

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 is added in the PA step, then it is added also a link between u and a random neighbor w of v. This model yields networks with high clustering coefficient and has been extensively studied by Holme and Kim (2002) and Szabó et al. (2003).

Another model in the family of PA models is the *biased preferential attachment model* described by Kumar et al. (2010). 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:

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

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. Kumar et al. (2010) provided no analytical results about the network metrics. However, they claim to be able to reproduce parameters they measured in two real social networks (Yahoo 360 and Flickr). Consequently we expect that, at least for some choice of parameters, the methods yields a network with high clustering coefficient, short diameter and power-law degree distribution.

The last model we review is called *transitive linking* (Davidsen et al. 2002). 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 $p \ll 1$ (i) the clustering coefficient is rather large, (ii) the TL dominates the process and (iii) the degree distribution is essentially a power-law with a cutoff, as nodes have finite lifetime. For larger values of p the two different process concur to form an exponential degree distribution, while for $p \approx 1$ the degree distribution is essentially Poisson distribution.

4 Agent-based modeling for social simulations

Agent-Based Modeling (ABM) (Bonabeau 2002) is a very powerful simulation technique that has been adopted in a number of application scenarios in the last few years

and that is now gaining increasing interest. ABM is a mindset more than a technology and the ABM mindset consists of describing the subject of a simulation from the perspective of its independent constituent units.

In ABM, the subject of simulation is described in terms of a collection of autonomous decision-making entities, called agents, that are grouped together to form an agent-based model. Each agent individually assesses its situation and makes its own decisions, and a model roughly consists of a possibly large group of agents, together with their relationships.

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) (Li et al. 2008) 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 constructing 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 may be all different one from each other;
- 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.

5 Towards agent-based models

The models presented in Sect. 3 are successful in producing networks with the desired properties with high probability. Moreover, they use only repetitive micro-level interactions that can be seen as sufficient conditions for the desired macro-level features. However, they still 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, the kind of mathematical models studied so far essentially assumes 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 properties that are mathematically inferred are present at the thermodynamic limit, but the real populations are typically much smaller.

Part of the issue is 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 manage to 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.

Unfortunately, each agent-based model tends to be an ad-hoc construction specifically tailored for a research problem, which is both a blessing and a curse. As a consequence, no agent-based models 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 describe, in essence, far more general concepts that arise in many different fields. Preferential attachment, for example, was first studied in the contexts (i) of genus of flowering plants (Yule 1925), (ii) of sizes of cities (Simon 1955), (iii) of citations, scientific productivity and journal use (Merton 1968; Price 1976), (iv) and, eventually, of web links (Barabási and Albert 1999).

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

However, there is a very crucial difference between models either created with mathematical analysis in mind or based on agents, and it is the notion and representation of time. In fact, some mathematical models have a notion of time internal to the model, such as the steps in the BA model or the evolution of a disease outbreak over a network. However, they assume that all the events are instantaneous and similarly

the consequences of their actions are immediate, unless the model specifies that some in-model-time must pass. With "event" we indicate anything that is relevant for the model, such as performing a choice 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, both (i) the actual occurrence and (ii) the fact that in a later time the consequence is bound to occur are events. Essentially, if we imagine the model implemented in a generic programming language, any step with side-effects in the structural operational semantics of the program (Plotkin 2004) is an event.

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* (Agha 1986). 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 (Dymond and Cook 1980) 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 loses the benefits of a distributed systems.

6 Concurrency issues in agent-based simulations

In Sect. 5 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 Sect. 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 and TL), while others were subject to analytic study only at a second time (WS). 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 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 (Agha 1986). 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.

In order to clarify this issue, we choose the Barabasi-Albert Model (BA Model) for its simplicity. The BA model 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 process at 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^i_j be the time when node t_i chooses its j-th target and let M^i_j 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^i_j, when the message is actually sent, and (b) R^i_j when node $i(j)$ receives the message. The multiple causal relation $C^i_j < S^i_j < R^i_j < M^i_j$ 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:

$$C^i_j < C^i_k \quad \forall i \in \{1, \ldots, n\}, \text{ if } j < k \tag{2}$$

$$M^i_j < M^i_k \quad \forall i \in \{1, \ldots, n\}, \text{ if } j < k \tag{3}$$

$$C^i_j < M^i_j \quad \forall i \in \{1, \ldots, n\}, \ \forall j \in \{1, \ldots, m\} \tag{4}$$

$$C^i_{j_1} < M^i_{j_2} \quad \forall i \in \{1, \ldots, n\}, \ \forall (j_1, j_2) \in \{1, \ldots, m\}^2, \ j_1 \neq j_2 \tag{5}$$

$$M^i_{j_1} < C^{i+1}_{j_2} \quad \forall i \in \{1, \ldots, n\}, \ \forall (j_1, j_2) \in \{1, \ldots, m\}^2, \ j_1 \neq j_2 \tag{6}$$

Equation (2) is trivially true. Considering that the model is meant sequentially executed in an imperative language, also Eq. (3) is trivially true. Equation (4) expresses the causal relation between the choice of a node as the recipient of a link and the creation of that link. Equation (5) indicates that in a single step all the targets are chosen before any link is added to the network. Equation (6) states that before processing step $i + 1$, all the links decided in step i must have been created.

In Fig. 1 we present how a program implementing the model could look like in an agent-based pseudo-language. 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 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 Eq. (3); nonetheless, it has little impact on the generated network.

```
1:  class NODE extends Agent
2:      handler ACT
3:          targets ← {}
4:          while |targets| < m do
5:              target = pref-attachment(graph)
6:              if target ∉ targets then
7:                  targets ← targets ∪ {target}
8:              end if
9:          end while
10:         for target in targets do
11:             send(target, "link-to")
12:         end for
13:     end handler
14: end class
15:
16: class ACTIVATOR extends Agent
17:     handler RUN
18:         for step in 1 . . . (n − m0) do
19:             node ← create-node()
20:             send(node, "act")
21:         end for
22:     end handler
23: end class
```

Fig. 1 Agent-based Barabàsi-Albert model

Fig. 2 Alternative implementation of BA nodes

```
1:  class NODE extends Agent
2:      handler ACT
3:          targets ← {}
4:          while |targets| < m do
5:              target = pref-attachment(graph)
6:              if target ∉ targets then
7:                  targets ← targets ∪ {target}
8:                  send(target, "link-to")
9:              end if
10:         end while
11:     end handler
12: end class
```

The problem is that, in general Eq. (6) 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 precedently 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. Barabási and Albert (1999) proved that the latter creates a completely different degree distribution altogether and, effectively, it would not be a BA process anymore.

With the implementation provided in Fig. 1, Eqs. (4) and (5) are always true, even with interruptible handlers. However, in some circumstances a different implementation could be more desirable. In Fig. 2 we show such 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 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 mes-

sage, which is taken care of by another thread on another CPU/core, and proceed with the following choice.

However, in this scenario Eq. (5) 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 Eq. (6) 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.

7 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 Fig. 2, based on the meta-model described in Sect. 8. In environments with limited concurrency, it is completely equivalent to the one of Fig. 1, however, increasing concurrency it could introduce more visible biases. The model was implemented in both the agent-based simulation engines we built, which are described in Sect. 9. 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 Sect. 6, that is to say single-executor, uninterruptible handlers. In this scenario, Eq. (6) 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

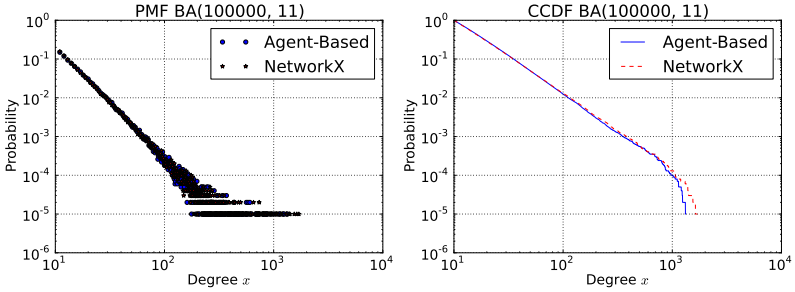
Table 1 Comparison between networks created according to agent-based and traditional Barabàsi-Albert models (denoted by the label "Sequential"). Several networks have been generated with different sizes (n) and number of starting edges per node (m). We compare the clustering coefficient C, the characteristic 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 such small networks

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

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.

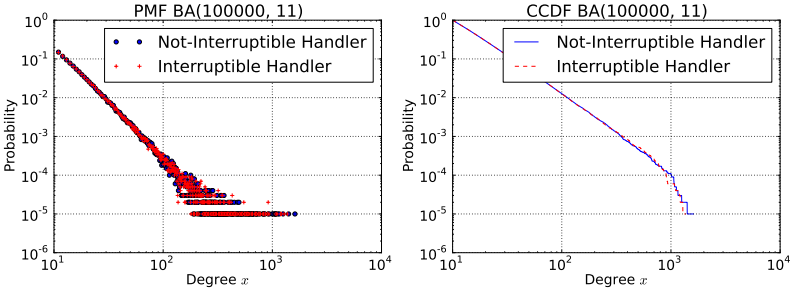
In Table 1 we report some results: we generated networks of different sizes (n) and with different starting number of edges per node (m) with both tools and found out that the clustering coefficient C and the characteristic path length CPL are significantly close one to each other (typically they differ for less than 10 %). We also compared the values with those predicted with the purely analytic methods described in Sect. 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 et al. (2009); their γ coefficients are, as predicted by analytical means, close to 3. In Fig. 3 we report the degree distributions $f(k)$ (Fig. 3a) and the complementary cumulative dis-



(a) PMF (b) CCDF

Fig. 3 Comparison between sequential and agent-based implementation of the BA model, 100000 nodes, 11 edges per node. Panels **a** and **b** represent the degree distributions and complimentary cumulative distribution functions (CCDF) of the networks



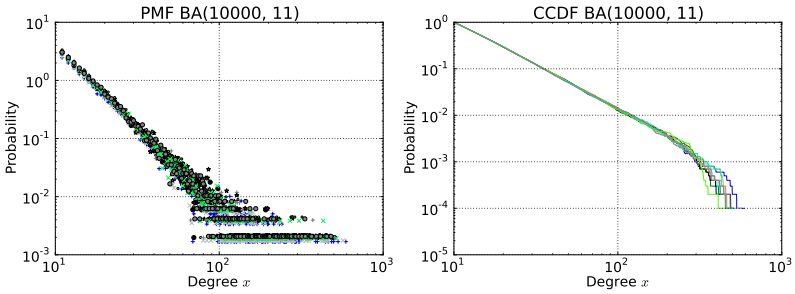
(a) PMF (b) CCDF

Fig. 4 Comparison between agent-based implementations of BA model with and without interruptible handlers, 100000 nodes, 11 edges per node. Panels **a** and **b** represent the degree distributions and complimentary cumulative distribution functions (CCDF) of the networks

tribution functions $S(k) = 1 - \sum f(k)$ (Fig. 3b) 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 Fig. 4 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 by Clauset et al. (2009). 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.

Eventually, we compared multiple networks generated according to the agent-based model: in Fig. 5 we plotted the degree distributions and CCDFs of several



(a) PMF (b) CCDF

Fig. 5 Degree distributions of 10 networks generated with the agent-based simulator (non-interruptible handlers version) and plotted by different symbols and color. Plot **a** represents the degree distribution of the networks, plot **b** their CCDFs

(10) such networks. The distributions are significantly close one to the other and this fact allows us to conclude that the available schedulers are fair enough for our purposes, which implies that actions are not delayed too much nor there is a significant bias in which actions are delayed. Essentially, the number of delayed actions (e.g., the creation of a link) is small, so that they have not a major impact on the simulation results and do not introduce significant biases in the models.

8 An agent-based meta-model for easy adaptation of traditional models

In order to simplify the creation of new agent-based models and 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.

Analyzing the traditional models described in Sect. 3 as well as some other non-generative network processes, such as the infection diffusion models described by Pastor-Satorras and Vespignani (2001), we singled out a meta-model that is suitable to implement said models as agent-based models. The programs presented in Fig. 1 and in Fig. 2 are essentially instances of that meta-model. 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.

In order to run the traditional models in an agent-based scenario, for example to generate the results presented in Sect. 7, it is sufficient to create a software framework that provides at each stop two decision points:

– a selection process determining groups of nodes to be created, "activated" or destroyed, and
– the behavior of the nodes themselves when activated.

A few infrastructural agents are needed in order to support such processes, e.g., (i) the Activator, responsible for the first of the two decision processes, (ii) a Configurator agent, that initially sets the network up in the starting condition of the simulation, (iii) a Clock agent that beats the steps for the activator. The vast majority of

agents in the system are, nonetheless, the nodes themselves. Although in the simplest models they simply wait to be activated, in more typically agent-based models they act pro-actively.

In the rest of this section, we show how the traditional models can be adapted to agent-based models using the meta-model we just described. It is worth noting that all the generation models have explicitly or implicitly a notion of "time", in the sense that in the BA, TL and 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 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, we 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 cause 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.

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.

However, as the traditional models become more complicated, 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 is not more complicated in an agent-based model, because in agent-based models behavior is a first class citizen. 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 also relatively trivial and simply executing the simulation assess their impact; 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
– it introduces itself 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.

9 Engineering the agent-based social network generation system

Some tools for agent-based simulation have been already developed (Minar et al. 1996; North et al. 2007; Tisue and Wilensky 2004; Luke et al. 2005) and we refer to the review made by Lytinen and Railsback (2012) for a comparison of the tools in the generic agent-based modeling context. However, we decided to implement our own agent-based social network generators, since:

– We plan to support both discrete events simulations and continuous simulations. The tools that are available tend to focus on one aspect and lack features pertaining the other one.
– We need a high level of control over the scheduling process and concurrency in general. Although our preliminary results show that average schedulers are fair enough for our purposes, it is also of interest to perform simulations with biased schedulers.
– We do not need many of the features that general purpose agent-based simulation environments offer and occasionally those features may contrast with our goals (i.e., social network specific simulations opposed to generic agent-based simulations) and make the software more complex to use by researchers with different expertises, e.g., in sociology.

We created two different implementations of our system. The first one is built in Java over HDS (Heterogeneous Distributed System) (Poggi 2010), while the other, called PyNetSym (Franchi 2012a), is written in Python and uses a custom simulation engine we created on the top of gevent.[1] Both systems implement the meta-model described in Sect. 8 and have been used to run the experiments described in Sect. 7.

[1] http://gevent.org.

HDS is a software framework that was originally created to simplify the realization of distributed applications by merging the client-server and the peer-to-peer paradigms and by implementing all the interactions among all the software entities of a system through the exchange of messages.

This software framework allows the realization of systems based on two types of software entities: actors and servers, which can be distributed on a (heterogeneous) network of computational nodes (hereafter called runtime nodes). Actors have their own thread of execution and perform tasks interacting, if necessary, with other software entities through synchronous and asynchronous messages. Servers perform tasks on request of other actors.

HDS has been already used for implementing some agent based applications (Bergenti et al. 2010; Bergenti and Poggi 2011) and provides two different ways for the deployment of actors and servers that allow either: (i) to assign a thread to each actor and server; or (ii) to share a thread among a set of actors and servers. The last option is very interesting for developing applications that involve a massive number of agents (in fact, we developed some applications where millions of agents concurrently run in a single HDS node, provided enough RAM is available). Moreover, the possibility to switch between the two strategies is especially useful in order to analyze what occurs with different concurrency models.

Therefore, HDS can be easily used for simulating social networks where individuals are represented by agents implemented on the top of HDS actors and common services (e.g., monitoring, persistence and logging services) are managed by HDS servers. Moreover, the simulation size is scalable by both: (i) distributing the agents on a set of computational nodes; and (ii) managing the execution of the agents of a computational node through a single thread.

On the other hand, PyNetSym does not support yet distribution on multiple computational nodes, although the feature is planned. However, a finer grained control over concurrency issues is possible because execution cannot be preempted. The default setting is that message handlers (i.e., the methods that are called when a given message is received) are never interrupted. We chose this setting because it is quite natural when implementing agent-based variants of probabilistic models. However, if more concurrency is desired, it is possible to explicitly require context switches.

The cooperative multi-tasking engine is created on top of gevent, a networking and concurrency framework that uses coroutines to implement cooperative tasks, called "greenlets" in gevent lingo. Coroutines are a generalization of subroutines allowing multiple entry points for suspending and resuming execution at certain locations (De Moura and Ierusalimschy 2009). Gevent also provides the default greenlet 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. An example of the efficiency of greenlets is shown in Table 2. Further comparisons between greenlet and thread performances are described by Friborg et al. (2009).

10 Conclusions and future work

This work was motivated by the urgent demand for generating realistic social networks in order to use them to perform extensive simulations of processes over net-

Table 2 Greenlet and thread efficiency comparison: 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)
100	0.022	0.002
500	0.110	0.009
1000	0.224	0.018
10000	2.168	0.180
100000	21.85	1.796
1000000	223.3	18.24

works. We believe that there will be a growing need for this kind of simulations because of the pervasive roles social networking systems are assuming in our lives. However we have shown that, in order to derive sound conclusions from simulation of processes over a social network and in order to understand which features influence more significantly the process outcomes, the topological differences of OSNs require the simulations to be run on a very large number of networks.

This, together with the relative complexity of obtaining good network samples from OSNs, led us to the necessity of synthetically generating a varied and ample set of realistic networks, on which the simulations could be run.

We strongly advocate the introduction of more agent-based models in order to perform large scale social network simulations. Among the many strong points of agent-based models, we think that the following are especially relevant for our purposes:

- agent-based models can easily deal with entities with different behaviors and behaviors changing over time,
- they are built simply describing the low level interactions among the parties, i.e., from the description of the people's behavior in the social networking system, taking advantage of the expertise of social scientists,
- they can model complex interactions between the nodes in the system, encompassing learning and goals, but they can also create emergent patterns from simple micro-level interactions
- they do not need simplifying assumptions to allow analytic studies.

However, we planned to introduce advanced agent-based features only gradually, because the effects of some of them can be hard to evaluate in advance. Moreover, when the amounts of "agency" in the model increases, it is easy to introduce unforeseen effects in the simulation. Consequently, we decided to start converting some of the most studied traditional models into agent-based models, so that we can study on familiar example the effects of increased concurrency, that can change the semantics of some operations in subtle ways. Increased concurrency is not only a consequence of agency, but it is also a side effect of distributing simulations on many computational nodes.

Then, we discussed concurrency related issues that arise when mathematical models are transformed into agent-based ones, since actions that in mathematical models are sequential may be executed in parallel and in a different order. We also showed that the schedulers typically found in commonly available concurrency libraries are fair enough to avoid the pathological situations that could change the model behavior.

We created a meta-model to ease the transition of traditional models to agent-based ones and we built two simulation systems using different concurrency technologies, implementing the meta-model. We are now in the position to gather enough results to improve the models further.

As a general comment it should be noted that the question on the extent to which predictions valid for artificially generated social networks also hold for real online social networks is still open and that, in general, this issue is not easily dealt with. However, results look promising and with more sophisticated network models it is likely that simulation is going to be an increasingly important technique for social network related studies.

References

Agha G (1986) Actors: a model of concurrent computation in distributed systems. MIT Press, Cambridge

Albert R, Barabási AL (2002) Statistical mechanics of complex networks. Rev Mod Phys 74:47–74

Barabási AL, Albert R (1999) Emergence of scaling in random networks. Science 286:509–512

Bergenti F, Poggi A (2011) Building distributed and pervasive information management systems with HDS. In: Pallotta V, Soro A, Vargiu E (eds) Advances in distributed agent-based retrieval tools, Studies in computational intelligence. Springer, Berlin

Bergenti F, Franchi E, Poggi A (2010) Using HDS for realizing multiagent applications. In: Proceedings of the third international workshop on languages, methodologies and development tools for multi-agent systems (LADS'010), Lyon, France

Bergenti F, Franchi E, Poggi A (2011) Agent-based social networks for enterprise collaboration. In: 20th international workshops on enabling technologies: infrastructure for collaborative enterprises. IEEE, New York, pp 25–28

Bergenti F, Franchi E, Poggi A (2012) Enhancing social networks with agent and semantic web technologies. In: Brüggemann S, D'Amato C (eds) Collaboration and the semantic web: social networks, knowledge networks, and knowledge resources. IGI Global, Hershey, pp 83–100

Bollobás B, Chung FRK (1988) The diameter of a cycle plus a random matching. SIAM J Discrete Math 1(3):328–333

Bonabeau E (2002) Agent-based modeling: methods and techniques for simulating human systems. Proc Natl Acad Sci USA 99(Suppl 3):7280–7287

Clauset A, Shalizi C, Newman M (2009) Power-law distributions in empirical data. SIAM Rev 51(4): 661–703

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

Davidsen J, Ebel H, Bornholdt S (2002) Emergence of a small world from local interactions: modeling acquaintance networks. Phys Rev Lett 88(12):1–4

De Moura AL, Ierusalimschy R (2009) Revisiting coroutines. ACM Trans Program Lang Syst 31(2): 6:1–6:31

Dymond PW, Cook SA (1980) Hardware complexity and parallel computation. In: IEEE annual symposium on foundations of computer science. IEEE Computer Society, Los Alamitos, pp 360–372

Epstein JM (1999) Agent-based computational models and generative social science. Complexity 4(5): 41–60

Erdős P, Rényi A (1959) On random graphs. Publ Math 6(26):290–297

Franchi E (2012a) A domain specific language approach for agent-based social network modeling. In: International conference on advances in social network analysis and mining (ASONAM 2012), Istanbul, Turkey. IEEE Computer Society, Los Alamitos

Franchi E (2012b) Towards agent-based models for synthetic social network generation. In: Putnik GD, Cruz-Cunha MM (eds) Virtual and networked organizations, emergent technologies and tools, communications in computer and information science, vol 248. Springer, Berlin, pp 18–27
Franchi E, Poggi A (2011) Multi-agent systems and social networks. In: Cruz-Cunha M, Putnik GD, Lopes N, Gonçalves P, EM Miranda (eds) Business social networking: organizational, managerial, and technological dimensions. IGI Global, Hershey
Friborg RM, Bjørndalen J, Vinter B (2009) Three unique implementations of processes for PyCSP. In: Welch PH, Roebbers H, Broenink JF, Barnes FRM, Ritson CG, Sampson AT, Stiles GS, Vinter B (eds) Communicating process architectures 2009—WoTUG-32, Concurrent systems engineering, vol 67. IOS Press, Amsterdam, pp 277–293
Holme P, Kim B (2002) Growing scale-free networks with tunable clustering. Phys Rev E 65(2):2–5
Jackson M (2010) Social and economic networks. Princeton University Press, Princeton
Kumar R, Novak J, Tomkins A (2010) Structure and evolution of online social networks. In: Yu PSS, Han J, Faloutsos C (eds) Link mining: models, algorithms, and applications. Springer, New York, pp 337–357
Li X, Mao W, Zeng D, Wang FY (2008) Agent-based social simulation and modeling in social computing. In: Yang C, Chen H, Chau M, Chang K, Lang SD, Chen P, Hsieh R, Zeng D, Wang FY, Carley K, Mao W, Zhan J (eds) Intelligence and security informatics. Lecture notes in computer science, vol 5075. Springer, Berlin, pp 401–412
Luke S, Cioffi-Revilla C, Panait L, Sullivan K, Balan G (2005) Mason: a multiagent simulation environment. Simulation 81(7):517–527
Lytinen S, Railsback S (2012) 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
Merton RK (1968) The Matthew effect in science. Science 169(3810):56–63
Minar N, Burkhart R, Langton C, Askenazi M (1996) The Swarm simulation system: a toolkit for building multi-agent simulations. Tech rep, Santa Fe Institute, Santa Fe
Newman MEJ (2000) Models of the small world. J Stat Phys 101(3):819–841
Newman MEJ (2003a) Properties of highly clustered networks. Phys Rev E 68:026121
Newman MEJ (2003b) The structure and function of complex networks. SIAM Rev 45(2):167–256
Newman MEJ, Watts DJ (1999) Renormalization group analysis of the small-world network model. Phys Lett A 263:341–346
North M, Howe T, Collier N, Vos J (2007) A declarative model assembly infrastructure for verification and validation. In: Advancing social simulation: the First World Congress. Springer, Berlin, pp 129–140
Pastor-Satorras R, Vespignani A (2001) Epidemic spreading in scale-free networks. Phys Rev Lett 86:3200–3203
Plotkin G (2004) A structural approach to operational semantics. J Log Algebr Program 60–61(0):17–139
Poggi A (2010) HDS: a software framework for the realization of pervasive applications. WSEAS Trans Comput 9(10):1149–1159
Price DDS (1976) A general theory of bibliometric and other cumulative advantage processes. J Am Soc Inf Sci 27(5):292–306
Simon HA (1955) On a class of skew distribution functions. Biometrika 42(3–4):425–440
Snijders TA (2011) Statistical models for social networks. Annu Rev Sociol 37(1):131–153
Szabó G, Alava M, Kertész J (2003) Structural transitions in scale-free networks. Phys Rev E 67(5):1–5
Tisue S, Wilensky U (2004) NetLogo: a simple environment for modeling complexity. In: International conference on complex systems, Boston, MA, pp 16–21
Watts DJ, Strogatz S (1998) Collective dynamics of small-world networks. Nature 393(6684):440–442
Yule GU (1925) A mathematical theory of evolution, based on the conclusions of Dr. J.C. Willis, F.R.S. Philos Trans R Soc Lond, B Contain Pap Biol Character 213(402–410):21–87

Federico Bergenti received a Ph.D. in Information Technologies from the University of Parma (Department of Information Engineering) in 2002. From the same University, he obtained a Laurea degree (M.Sc.) in Electronic Engineering in 1998. From October 2007 he is a permanent researcher in Computer Science at the University of Parma and he is affiliated to the Department of Mathematics. Federico's research activity has been mainly devoted to Artificial Intelligence and Software Engineering, with special regard to multi-agent systems. In the field of Artificial Intelligence he has worked on issues related to agent communication languages and their semantics. In the field of Software Engineering, he initially worked on

architectures for agent-based middleware; then he concentrated on issues related to reusability and strong decoupling in agent-based systems. More recently, he turned his interests towards agent programming languages; he is particularly interested in approaches that rely on constraint (logic) programming.

Enrico Franchi received B.Sc. in Mathematics and Computer Science and M.Sc. in Computer Science from the University of Parma. He is currently enrolled in the Ph.D. course in Information Technologies from the same University under the supervision of Prof. Agostino Poggi. His main interests are related to Multi-Agent and distributed systems, social networks, artificial intelligence and software engineering. He is currently investigating the mutual relationships between social networks and multi-agent systems, with a special regard to simulations and to the creation of distributed online social networks.

Agostino Poggi is full professor of Computer Engineering at the Faculty of Engineering of the University of Parma. He coordinates the Agent and Object Technology Lab and his research focuses on agent and object-oriented technologies and their use to develop distributed and complex systems. He is author of more than a hundred of technical papers in refereed journals and conferences and his scientific contribution has been recognized through the "System Research Foundation Outstanding Scholarly Contribution Award" and the "System Innovation Award". Moreover, he is in the editorial board of the following scientific journals: Software Practice & Experience, International Journal of Hybrid Intelligent Systems, International Journal of Agent-Oriented Software Engineering, International Journal of Multiagent and Grid Systems e International Journal of Software Architecture.