Simulation

Simulation: Transactions of the Society for Modeling and Simulation International
89(10) 1173–1183

DOI: 10.1177/0037549713495752
sim.sagepub.com
SAGE SCS

Modeling and simulation of large-scale social networks using parallel discrete event simulation

Bonan Hou[1], Yiping Yao[1,2], Bing Wang[3] and Dongsheng Liao[4]

Abstract

The modeling and simulation of social networks is an important approach to better understanding complex social phenomena, especially when the inner structure has remarkable impact on behavior. With the availability of unprecedented data sets, simulating large-scale social networks of millions, or even billions, of entities has become a new challenge. Current simulation environments for social studies are mostly sequential and may not be efficient when social networks grow to a certain size. In order to facilitate large-scale social network modeling and simulation, this paper proposes a framework named SUPE-Net, which is based on a parallel discrete event simulation environment YH-SUPE for massively parallel architectures. The framework is designed as a layered architecture with utilities for network generation, algorithms and agent-based modeling. Distributed adjacency lists are used for graph modeling and a reaction–diffusion paradigm is adapted to model dynamical processes. Experiments are performed using PageRank and the susceptible–infected–recovered (SIR) model on social networks with millions of entities. The results demonstrate that SUPE-Net has achieved a speedup of 12, and increased the event-processing rate by 11%, with good scalability and effectiveness.

Keywords

Social networks, modeling and simulation, parallel discrete event simulation, PageRank, epidemic model

1. Introduction

With the rapid advent of online social networks (OSNs) and mobile devices, huge numbers of people can stay interconnected at an unprecedented scale. The people and their relationships form a social network, which plays an important role in transforming our lifestyle in the social media era. The network sheds light on the way we communicate, the lifestyle we choose and the opinion leader we follow on Web 2.0. In fact, social networks provide an abstract, explicit view of a complex sociality composed of interacting social factors.[1,2] Therefore, social networks have become an effective modeling paradigm for better understanding many social phenomena arising in different areas of application, including information/virus/rumor/gossip spreading,[3–5] finding the most important opinion leader,[6] influence maximization[7] and privacy protection.[8] These topics have been attracting extensive attention from sociologists, physicists and computer scientists.[9–11]

Social networks modeling and simulation still faces challenges. First, the underlying network structure often reaches terascale or petascale, which means the entire graph structure cannot fit into the main memory of a single computer. Thus parallel and distributed processing is arguably the only choice we have at present. Second, the algorithms of social network analysis (SNA) often exhibit embarrassing-parallel characteristics for fine-grain events, uncertain lookahead and a changing degree of parallelism during execution.[12] The irregular structure, for example scale-free and small-world characteristics, of most social networks also leads to limited parallelism.[13] Finally, the

[1]School of Computer Science, National University of Defense Technology, Changsha, China
[2]School of Information System and Management, National University of Defense Technology, Changsha, China
[3]Software Division, Beijing Institute of Systems Engineering, China
[4]School of Humanities and Social Sciences, National University of Defense Technology, Changsha, China

Corresponding author:
Yiping Yao, School of Computer Science, National University of Defense Technology, 410073, Changsha, China.
Email: ypyao@nudt.edu.cn

Bing Wang, Software Division,
Beijing Institute of Systems Engineering, 100101, Beijing, China.
Email: wangbing.nudt@gmail.com

inherent randomness and intensive communication of social dynamics make scalability a prominent issue. These requirements go well beyond what the mechanisms of statistical analysis or numerical simulation can provide.[14]

It is necessary to construct a scalable, efficient parallel simulation infrastructure to simulate complex social phenomena on large-scale networks. It should satisfy the requirements of the study of social networks, such as reproducing self-organization of complex networks and testing what-if hypotheses. Because of our limited capability to interact with real-world social networks, it is necessary to conduct simulation experiments on large-scale network models with more precision. Compared to mathematical models based on differential equations, computer models improve our capacity for prediction and provide causal explanations.[14] Furthermore, with the increasing scale of social networks under study, a computer model should scale well as more computing resources become available. So we also pay attention to the runtime performance of simulation, especially for large-scale networks.

The central contribution of this paper is that we construct a framework, SUPE-Net, that simulates large-scale social networks.[15] SUPE-Net is based on an object-oriented parallel discrete event simulation (PDES) engine utilizing the massive computing capability of current parallel architectures. The nodes and links of networks can be distributed as simulation entities onto multiple processors within the concurrent object paradigm. The SUPE-Net core is designed as a layered architecture. It includes a PDES engine, dynamics protocol, network modeling, graph analysis algorithms and behavior modeling. This allows us to design and analyze simulations on large-scale social networks and deploy them onto massively parallel architectures. To evaluate the framework, we execute the PageRank algorithm[16] and the susceptible–infected–recovered (SIR) epidemic model on large-scale networks using part of the Tianhe-1A supercomputer.[17] Results indicate that SUPE-Net has achieved a satisfactory speedup as more computing resources are used. We believe that this work will promote the booming research on social networks and their complex social phenomena.

The remainder of this paper is organized as follows. Section 2 introduces related works discussing large-scale network modeling and simulation. Section 3 proposes a framework, SUPE-Net, to simulate large-scale networks. In Section 4 we evaluate the SUPE-Net framework with a series of experiments. We discuss the meaning and limitations of this work in Section 5. Finally, Section 6 closes the paper with conclusions and provides directions for future work.

2. Related works

Recently we have witnessed an explosion of work devoted to large-scale network analysis and simulation. 'Network' as used here refers to the most general concept, which abstracts systems composed of interacting components, and represents them as a graph. 'Nodes' refers to components and 'links' are their relationship or interaction. Examples of such networked systems include, but are not limited to, computer networks,[8,19] knowledge networks,[20] biochemical networks,[21] human protein-interaction networks,[22] complex adaptive supply networks[23] and complex sensor networks.[24]

The focus of the social networks related to this work can be roughly classified into two categories: (a) Social Network Analysis (SNA); and (b) dynamics models and simulation. SNA, a novel branch of computational social sciences, focuses on mining social networks from massive data and capturing the topological characteristics of these networks. For instance, the popularity of OSNs in recent years has resulted in unprecedented large-scale networks: Facebook has more than 721 million unique worldwide Internet users.[25] Graph crawler techniques are widely used to capture the underlying network from the interaction information. Although sampling techniques have tried to capture a part of the network which manifests similar characteristics to the whole network,[26] open datasets available are often at the hundreds-of-millions scale.[25,27] The networks' topological characteristics we are interested in include degree distribution, betweenness centrality,[28] radius plot[29] and triangle counting.[30,31] Some algorithms, such as betweenness centrality computation, are NP-hard problems and difficult to extend to large-scale graphs. Parallel computing paradigms and platforms, like Hadoop/MapReduce[32] (Hadoop is an open-source implementation of MapReduce) and middle ware,[33] were explored to enable large-scale graph analysis. The emerged systems HADI,[34] PEGASUS,[35] Google's Pregel[36] and Parallel BGL[37] distribute the large-scale graph structure onto multi-processors and the computing for each part is coordinated by a global synchronization mechanism. One drawback of these frameworks is that they have a limited modeling capacity which cannot satisfy detailed dynamics modeling requirements. In addition, their synchronization mechanisms are based either on MapReduce or Bulk Synchronous Parallel,[38] which behave similarly to time-stepped simulation execution and thus are not suitable for applications with a near zero lookahead.[39] This will be explained in detail in Section 4.3 in which we discuss our experiments.

The other category of current work focuses on modeling specific social phenomena, such as the spreading dynamics of and the growth or evolution of social networks. Kang et al.[40] studied the structure evolution of large networks and proposed a generative model to explain the dynamic growth process. They also successfully scaled the belief propagation (BP) algorithm on a graph with billions of nodes and edges using the Hadoop platform.[41] Epidemic spreading that models disease or viruses diffusing through physical or social contact is one of the most studied subjects in modeling complex phenomena on large-scale networks. It has been observed that the

structure of the contact network plays a vital role in how fast, and to what extent, the disease finally spreads. Perumalla and Seal successfully executed an epidemic outbreak model in a population of over 800 million individuals using thousands of processor cores with the help of the μsik PDES engine.[42]

In the modeling paradigms for complex networked systems, the papers by Barrett et al.[43] and Bisset et al.[44] introduced an interaction-based approach for modeling the interacting components. The stochastic actor-based model[45] is also suitable for modeling network dynamics and has been applied in simulating control protocols on large sensor networks.[24] In contrast, our approach is based on PDES, thus combining these two features. We use parallel simulation entities to model the autonomous actors and discrete event scheduling to model intensive interactions.

In addition, the scalability techniques present in parallel and distributed simulation, including the scalable time warp[46] (which finds its application in modeling billion-node torus networks)[47] and distributed synchronization,[48,49] provide good references for our work.

3. SUPE-Net: Simulating large-scale social networks

3.1. Mapping the social networks onto multi-processors

An intuitive approach toward simulating large-scale social networks on massively parallel architectures is to distribute the nodes and links across multi-processors. Each processor would manage part of the network data model. The whole simulation would be coordinated by a synchronization mechanism. This mapping concept is illustrated in Figure 1(a).

Adjacency lists and adjacency matrices are the two basic data structures used to represent a graph $G = (V, E)$ on a computer when modeling graph data. In adjacency-list representation, each node has a list that contains the information of its neighboring nodes. Adjacency lists and matrices both have memory and efficiency trade-offs. We prefer the distributed adjacency list data structure in our work for parallel processing. The advantage of adjacency lists is that 1) they can easily be partitioned and independently distributed, which favors scalability; and 2) they are especially efficient for neighborhood-based operations that dynamics algorithms perform on the network. The data structure of a sample network is represented by Figure 1(b). Each node has a global ID and can be uniquely identified. Each node has link information only for its neighbors. Inter-processor communication may be required if the interaction target resides on a different processor.

3.2. Layered architecture

PDES is an important parallel paradigm enabling large-scale social network mapping. The complexities and complicated dynamics of social networks make it difficult to adapt PDES efficiently as the environment for large-scale social network simulations. In this work, scalability, efficiency and flexibility are the chief modeling capacities we are concerned with when designing the simulation framework. The architecture of SUPE-Net (see Figure 2) is designed as three separated layers from bottom to top: 1) the parallel simulation engine; 2) the social networks modeling and simulation layer; and 3) the network algorithms and utilities layer.

3.3. YH-SUPE: Parallel discrete event simulation

Starting at the bottom of the architecture, YH-SUPE[50] provides a high-performance, object-oriented PDES simulation engine. Concurrent entities are basic units in YH-SUPE. They interact with one another through asynchronous event scheduling. Each entity can be referred to by its unique ID, which contains the processor ID, kind ID and a serial number. Each entity maintains some local state and a pending list of time-stamped events that have been scheduled for this entity. The simulation time of an entity is advanced by repeatedly processing the smallest time-stamped event. The global simulation time is coordinated by synchronization algorithms which are as classified either the conservative or the optimistic approach.

Simulation entities can be created on any processor and accessed remotely through event scheduling with the concept of processor virtualization. The simulation engine automatically distributes the entities across available processors according to the users' strategy. Optional decomposition patterns include block, scatter, CommPar and customized. Block decomposition distributes the entities to processors evenly, while scatter operates as a card deal method. CommPar[51] is a community-based multi-level graph partition strategy suitable for social networks with community structure. Users can also customize distribution by specifying it in a parameter file.

In the topology mapping between the simulated network and multi-processors, we first divided the network structure into partitions. Each consisted of a set of nodes and their out-going edges. User-specified partition strategy determines the node's assignment to a partition. Because each node is identified with a global object handler, it is transparent to schedule an event for any other node even if it is on a different partition. Before execution, the engine assigns partitions to worker machines. It is possible to assign one or more partitions onto a worker machine. For example, the job scheduling command '*srun -N 5 -n 10 ./ application*' assigns 10 partitions of the simulation to five workers for parallel processing.

During execution, each worker machine maintains the state of its portion of the network in memory. Nodes interact with each other by scheduling events. When an

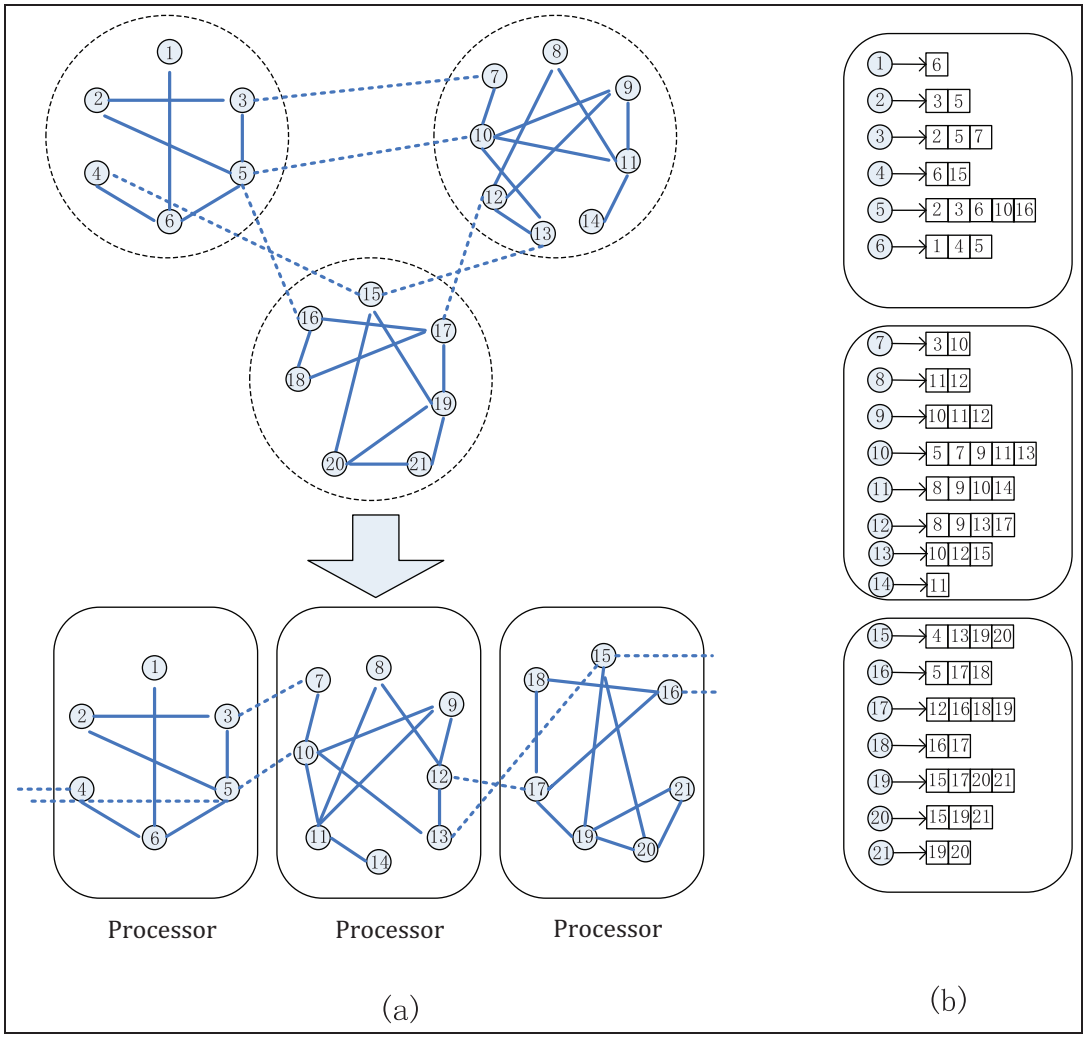

Figure 1. Mapping large-scale networks onto multi-processors: (a) a schematic drawing of the mapping concept, in which a sample network is partitioned and mapped to three processors; (b) the data structure in each processor using a distributed adjacency list.

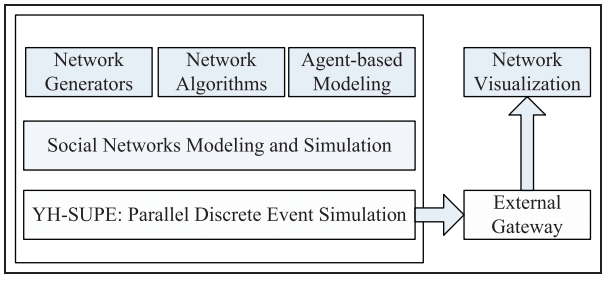

Figure 2. The layered architecture of SUPE-Net.

event is scheduled, the worker first determines whether the target node is on the same worker or on a remote worker. In a remote case, the event scheduling is delivered as a message passing. In a local case, the event is inserted directly into the destination node's pending event queue.

3.4. Social networks modeling and simulation

The social networks modeling and simulation layer provides basic modeling constructs, including node, edge, property and network entities, as illustrated by Figure 3. Each node contains essential characteristics and autonomous behavior. For example, the node can easily find its neighbors with the connected edges through a unified API, regardless of the processor they reside on. The edge represents the link between each pair of nodes. The edge has weight, and can be directed or undirected. Each node and edge can associate with one or more properties. The network entity provides utilities for display and statistics. Different from many other simulators, the network instance in SUPE-Net does not maintain a global network data structure thus avoiding memory exhaustion. Instead, we designed an event scheduling API to dynamically figure out the entire network dynamically at runtime through

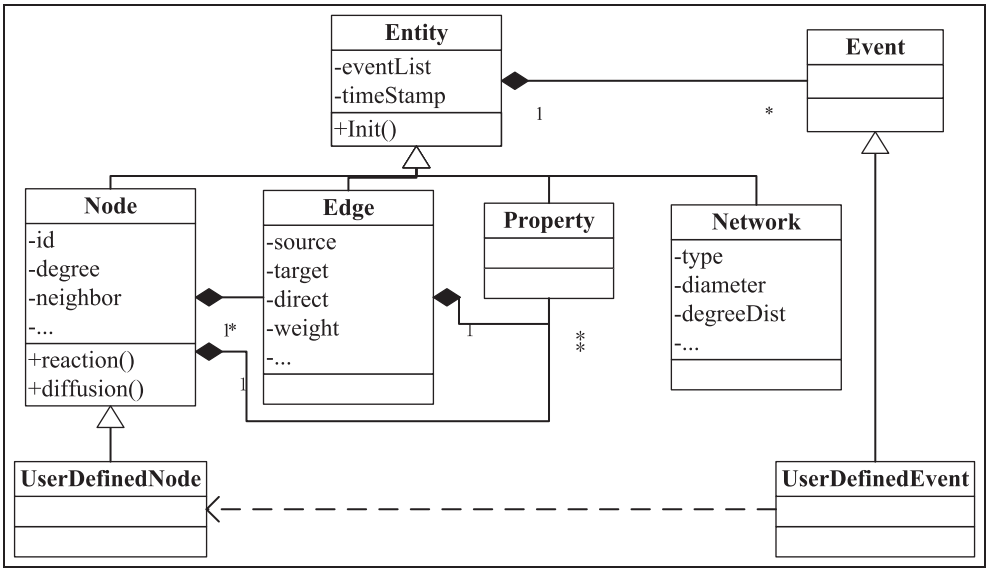

Figure 3. Class hierarchy of basic modeling constructs in SUPE-Net.

an interest management mechanism. These constructs' capability could easily be extended by users through inheritance.

Dynamical processes on social networks are important for better understanding nonlinear systems dynamics, for example the spread of contagion in a contacted population. The inherent dynamical process on social networks involves local reaction and global diffusion. Nodes will react once triggered or infected and then are apt to propagate further to their neighbors. Thus, we adopt a reaction–diffusion paradigm to model different protocols for dynamical processes with discrete events scheduling. Once an entity is triggered, it invokes its reaction events. When certain conditions are satisfied, the entity decides whether or not, and if so, how, to diffuse. This paradigm is especially efficient for spreading processes. The process then cascades as a chain of event scheduling until terminal conditions are satisfied. The event graph of the reaction–diffusion paradigm is shown by Figure 4.

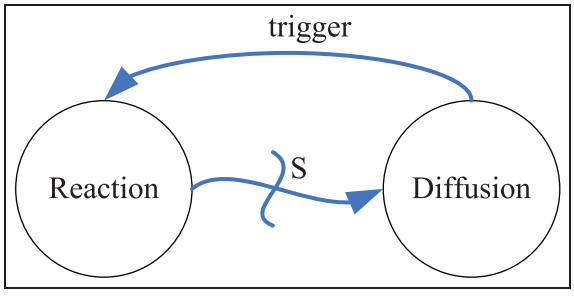

Figure 4. The event graph of the reaction–diffusion paradigm. Once an entity is triggered, a reaction event will be scheduled. The reaction may spawn further diffusion events if a condition is reached.

3.5. Utilities layer

The top utilities layer of the SUPE-Net framework offers fundamental modules for parallel network analysis. They include network generators, algorithms, visualization and compatibility with agent-based modeling.

1. Network generators. This module supports the random, small-world and scale-free network topology generation. Many synthetic algorithms, including the Erdös–Rényi random network, Watts–Strogatz small-world network and Barabási–Albert (BA) scale-free network,[52] can be used to generate networks with specified statistical characteristics. Although these classic models are synthetic, they offer a basis for comparative study and validation. In reality, social networks are often constructed through data mining from social network websites or databases. Social networks may manifest complicated characteristics that cannot be generated by these classic models. For this reason, we provided an IO interface for importing realistic topology from graph files. The file formats supported include Matrix, Graphml and Pajek. During implementation, the network instance is responsible for reading the link information, and telling the target node entity to update its neighbors.
2. Network algorithms. Network algorithms are essential utilities for SNA. Statistical algorithms for computing a network's degree distribution,

centralities, and other characteristics, are supported. Algorithms for community detection (finding the tightly connected groups) and node ranking (finding the most important nodes) are also considered. In these algorithms some optimizations, such as betweenness centrality (which involves calculating the shortest paths between each pair of nodes), are NP-hard. For parallel efficiency, we prefer flooding and local greedy approaches which can be modeled as event-scheduling chains. We use local modularity optimization to determine the community each node belongs to, and we use multiple random walkers on the network to find the most important nodes. The advantage of these approaches is their low complexity and good efficiency on multi-processors.

3. Agent-based modeling. Integrating agent-based modeling and social networks in SUPE-Net facilitates the reproduction of complex social phenomena. Because the social systems often consist of a large number of interacting, autonomous individuals, agent-based modeling enhances the ability to express entity autonomy. Each individual has its own storage (memory) and actions (social behavior). This utility provides a powerful way to manifest the emergence and self-organization social phenomena.
4. Network visualization. Visualization is one of the most useful ways for users to understand and validate social simulation. The network is distributed among multi-processors. Each processor maintains only a portion of the model. Thus no global state feeds the visualization terminal. The external gateway of YH-SUPE provides a special way to extract information from parallel simulation and knows how to locate the simulation entities. Network visualization subscribes the interested entities through the external gateway's API. The state update information is then reflected to the visualization.

4. Performance evaluation

This section will discuss an experimental study performed to evaluate the proposed social network simulation environment. Two representative models, PageRank and SIR epidemic dynamics, were adopted as benchmarks. The performance of SUPE-Net is measured using a variety of metrics.

4.1. Datasets and environment

The underlying social networks used in our experiments include both synthetic scale-free and realistic networks. The BA network generator is used to generate random scale-free networks. It begins with an initial network with m_0 nodes. New nodes are added to the network one at a time. Each new node is connected to m existing nodes through a preferential attachment mechanism. We used $m_0 = 4$, $m = 2$ to generate a series of scale-free networks of different sizes. We also select the movie–actor-collaboration network[52] and a friendship social network from the LiveJournal website.[53] These networks' data are listed in Table 1.

Table 1. The network data used in the experiment.

Network	Nodes	Links
BA-100	100	198
BA-1000	1,000	1,998
BA-10000	10,000	19,998
BA-100000	100,000	199,998
Actor	383,640	1,229,020
LiveJournal	4,846,609	68,475,391

The hardware environment used in the experiment is a part of Linux cluster of the Tianhe-1A supercomputer.[17] Each computing node is equipped with two 2.53 GHz QuadCore Xeon Processors (eight cores) and 8 GB RAM. The computing nodes are interconnected with InfiniBand. The operating system is Kylin Server 3.1, kernel 2.6.18. The GCC version is 4.1.2. The disks are logically shared among all computing nodes.

4.2. PageRank

The PageRank[54] algorithm was designed to rank documents on the World Wide Web for search engines; now this algorithm is widely used to measure the relative importance of linked things. The basic implementation updates each entity's score iteratively by taking into account its neighbors' scores each time. Numerically, PageRank can be computed as

$$G_i = \frac{1-q}{N} + q \sum_{j\ nn\ i} \frac{G_j}{k_j}$$

where q is a damping factor, k_j is the degree of neighbor j and N is the node number of the entire graph.

This iterative algorithm is suitable for time-stepped implementation. All nodes of the networks update their value in one step. They then exchange messages and synchronize during the next step. Here, however, we implemented PageRank with discrete events in order to exhibit the capability of our framework for this time-stepped algorithm. There are two kinds of events: ReceiveMessage and UpdateAndSend. Both are illustrated in Algorithm 1. Although the time increment is set to one to mimic the time-stepped update, these events are not explicitly synchronized during the execution. They may have been

processed out of sequential order, but the time synchronization mechanism maintains their causal order.

Algorithm 1. Discrete event implementation of PageRank.

```
1: Event ReceiveMessage(double value)
2:     sum + =value
3: end Event
4: Event UpdateAndSend()
5:     if simtime > = 1 then
6:         rankscore = 0.15/ NumNodes + 0.85*sum
7:         sum = 0.0
8:     end if
9:     for all individual ∈ Neighbors do
10:        val = rankscore/degree
11:        SCHEDULE_ReceiveMessage(simtime + 1.0,
           individual, val)
12:    end for
13:    SCHEDULE_UpdateAndSend(simtime + 2.0,
       this)
14: endevent
```

We first executed the PageRank algorithm on scale-free networks of different sizes. Figure 5 shows the execution time versus the network size on fixed eight-CPU cores. The nodes of these networks are distributed in block mode. The execution time logarithmically scales with the network size, and is indicated by the nearly straight solid line against the left axis on the log–log graph. We also plot the event processing rate, which is the number of events committed per second, against the right axis in the same figure. With the same computing resources, we can see an obvious break point in the event rate on the BA-1000 network. Then the event rate decreases as the network size increases, which indicates that the computing resource begins falling short for efficiency.

In order to test the scalability of PageRank on SUPE-Net, we ran the model on the BA-100000 and Actor networks with different computing cores. The performance is illustrated by Figure 6. Execution time decreases as more computing cores are used. For the Actor network, execution time achieves 3.23X relative speedup. A turning point appears with the BA-100000 network, since the communication cost exceeds the computation parallelism.

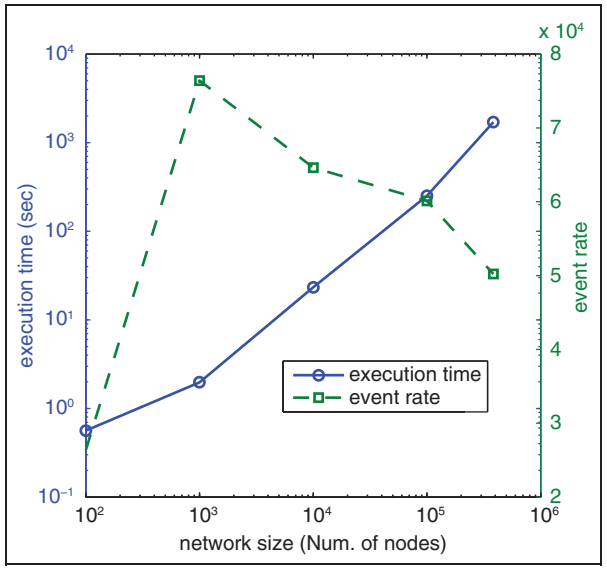

Figure 5. The performance of the PageRank model on networks with differing numbers of nodes (BA-100 to BA-100000, and the Actor network). The plot is double *y*-axes. The left *y*-axis is a log–log plot of the execution time of PageRank. The right *y*-axis indicates the event rate. The PageRank algorithm was simulated using 30 iterations.

4.3. SIR epidemic model

To simulate social dynamics, we applied the SIR model[55] on both the Actor and LiveJournal friendship networks. The SIR model has been widely used to describe both disease and rumor spreading through social contact. The population is classified into susceptible, infected and recovered. Once interacted, an infectious person will infect a susceptible neighbor with probability β. Those infected will recover with rate λ. Based on the basic model, we further take into account the behavior pattern of human interaction, as more and more evidence establishes that social interactions are heavy-tailed.[56] In the experiment, we generate the interval time, which is the incremental time that a person will next interact with or infect others, and which obeys a power law distribution. As a result, there is no

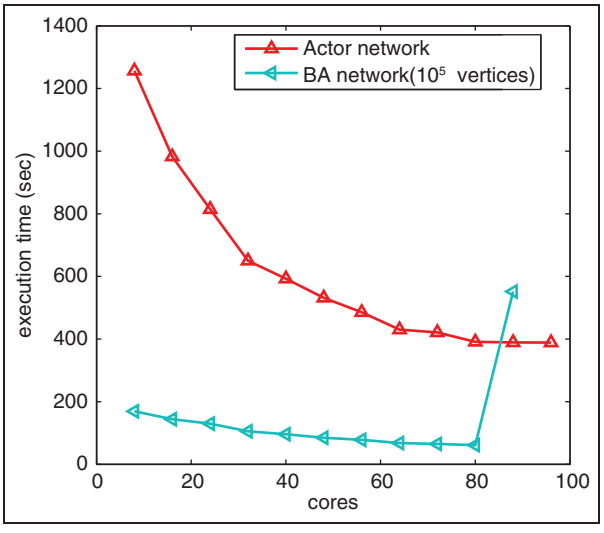

Figure 6. The execution time of the PageRank model on the Actor network and a BA network using different computing cores.

explicit lookahead in the model, which would have a tradeoff between performance and accuracy for time-stepped simulations. Fortunately, this is not a problem for event-driven simulation, especially for those that use optimistic time management.

The pseudo-code of the SIR model is shown in Algorithm 2.

Algorithm 2. Discrete event implementation of SIR.

```
1: Event Infect()
2:     if status = recovered then
3:         return
4:     end if
5:     if status = susceptible then
6:         status = infected
7:     end if
8:     //infectable
9:     for all individual ∈ Neighbors do
10:        if random() < infectRate then
11:            detat = generatePowerLaw(2.0)
12:            SCHEDULE_Infect(simtime + detat,
               individual)
13:        end if
14:        if random() < recoveredRate then
15:            status = recovered
16:        end if
17:    end for
18: end Event
```

We first execute the SIR model on Actor networks using both conservative and optimistic time synchronization algorithms. The conservative algorithm supported by YH-SUPE uses lookahead to avoid deadlock. The optimistic algorithm used in the experiment was Time Warp. The execution time of the simulation on different CPU cores is illustrated by Figure 7. We can see a significant divergence between conservative and optimistic algorithms. Due to zero lookahead, the conservative synchronization works even worse as more computing cores are used. However, the optimistic Time Warp scales well as shown by the inner plot.

We then measure the execution time and event processing rates scaling on different numbers of computing cores. When simulating the SIR model on the LiveJournal network, we use the optimistic Time Warp synchronization algorithm. The experimental results are illustrated in Figure 8. The average execution time and event processing rate are presented on the left and right axes, respectively. Our simulation scales well on SUPE-Net as more computing cores are used. The solid line against the left axis indicates that the execution time decreases as more computing cores are used. It achieves a relative speedup of about 12X. The dotted line against the right axis shows that the event processing rate also increases by 11%. The results demonstrate that SUPE-Net has acceptable efficiency and scalability.

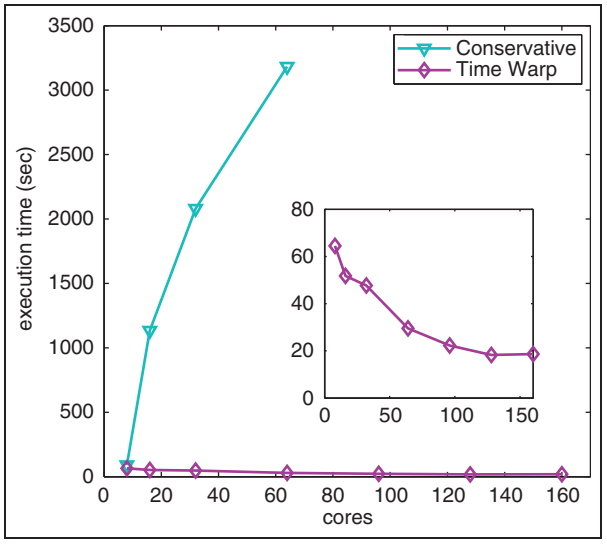

Figure 7. The performance of the SIR simulation on the Actor network using conservative and optimistic time management algorithms. The inset is the same plot for Time Warp as an eye-guide.

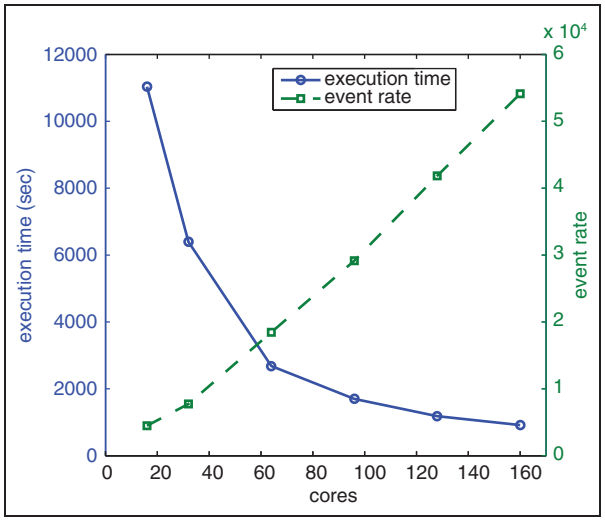

Figure 8. The performance of the SIR simulation on the LiveJournal network using different computing cores, plotted with double *y*-axes. The left *y*-axis is the execution time. Execution time decreases as more computing resources are used. The right *y*-axis is the event rate, indicating that the simulation efficiently processes more events per second using more computing resources.

5. Discussion

With the increasing scale of network structures in many complex systems, the modeling and simulation of large-scale social networks provides a powerful tool for complexity research. From the viewpoint of computation, the

proposed SUPE-Net framework faces the challenges common to large-scale graph processing in the field of computer science, including appropriate graph data modeling and efficient graph algorithm implementation. However, some differences remain between parallel graph processing techniques and parallel simulation. The goal of graph processing is graph query and response, for instance, a social recommendation system must instantly be able to respond to the question 'what are my friends buying?' Social network modeling and simulation focuses more on other capabilities, such as reproducing complex social phenomena, evaluating different policies and testing what-if hypotheses. In light of these differences, we chose PDES as an infrastructure, as it provides a more powerful modeling capacity. The experiments demonstrate that our framework can efficiently support dynamics simulation on large-scale networks. While satisfying performance and scalability requirements, SUPE-Net has a specific application area, with the inevitable limitations on instant graph processing and user interaction.

6. Conclusions and future work

In this study, we explored the modeling of social networks and proposed an experimental framework for social research on the ever-expanding interconnected networks of linked individuals. As the network scale reaches millions or billions, parallel and distributed processing seems to be the only choice we have. To perform this task, we implemented a SUPE-Net framework based on YH-SUPE, a PDES engine. With layered architecture, we provided utilities for network generation, algorithms and agent-based modeling. Further, a reaction–diffusion paradigm was designed to effectively implement a dynamical process using event scheduling chains. Finally, one case study of PageRank and the SIR model on social networks indicates the scalability and effectiveness of SUPE-Net.

With respect to future work, two directions require further research efforts. First, the irregular model structure of social network simulation and suitable performance optimization techniques, such as model partitioning and load balancing, are worth exploring. Second, many networked systems evolve over time, with nodes and links appearing and disappearing at various points on the time scale. This induces profound consequences for the dynamic processes taking place on them. Thus, how to model temporal and spatial networks becomes an important issue.

Funding

This work was funded by the National Natural Science Foundation of China (NSFC; grant numbers 61170048 and 91024030) and the PhD Programs Foundation of Ministry of Education of China (grant number 201243070017).

References

1. Lazer D, Pentland A, Adamic L, et al. SOCIAL SCIENCE: Computational social science. *Sci* 2009; 323: 721–723.
2. Boccaletti S, Latora V, Moreno Y, et al. Complex networks: Structure and dynamics. *Phys Rep: Rev Sec Phys Lett* 2006; 424: 175–308.
3. Barrat A, Barthélemy M and Vespignani A. Dynamical processes on complex networks. *J Stat Phys* 2009; 135: 773–774.
4. Newman MEJ. Spread of epidemic disease on networks. *Phys Rev E* 2002; 66: 016128.
5. Moreno Y, Nekovee M and Pacheco AF. Dynamics of rumor spreading in complex networks. *Phys Rev E* 2004; 69: 066130.
6. Lü L, Zhang Y-C, Yeung C, et al. Leaders in social networks, the Delicious case. *PloS One* 2011; 6(6): e21202.
7. Kempe D, Kleinberg JM and Tardos É. Maximizing the spread of influence through a social network. In: *Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining*, 2003, pp. 137–146.
8. Krishnamurthy B and Wills CE. Characterizing privacy in online social networks. In: *Proceedings of the first workshop on online social networks*, New York, NY, 2008, pp. 37–42.
9. Castellano C, Fortunato S and Loreto V. Statistical physics of social dynamics. *Rev Mod Phys* 2009; 81: 591–646.
10. Christensen C, Albert I, Grenfell B, et al. Disease dynamics in a dynamic social network. *Physica A – Stat Mech Appl* 2010; 389: 2663–2674.
11. Da Fontoura Costa L, Oliveira ON Jr, Travieso G, et al. Analyzing and modeling real-world phenomena with complex networks: A survey of applications. *Adv Phys* 2007; 60(3): 329–412.
12. Lumsdaine A, Gregor D, Hendrickson B, et al. Challenges in parallel graph processing. *Parallel Process Lett* 2007; 17: 5–20.
13. Hruz T, Geisseler S and Schöngens M. Parallelism in simulation and modeling of scale-free complex networks. *Parallel Comput* 2010; 36: 469–485.
14. Brase JM and Brown DL. Modeling, simulation and analysis of complex networked systems: A program plan. Technical Report, Lawrence Livermore National Laboratory, CA, 2009.
15. Hou B, Yao Y, Wang B, et al. SUPE-Net: An efficient parallel simulation environment for large-scale networked social dynamics. In: *Green computing and communications, IEEE/ACM international conference on & international conference on cyber, physical and social computing*, 2010, pp. 628–635.
16. Page L, Brin S, Motwani R, et al. The PageRank citation ranking: Bringing order to the web. Technical Report, Stanford Digital Library Technologies Project, CA, 1998.
17. Yang XJ, Liao XK, Lu K, et al. The TianHe-1A supercomputer: Its hardware and software. *J Comput Sci Tech* 2011; 26: 344–351.
18. Liu J. A primer for real-time simulation of large-scale networks. In: *Proceedings of the 41st annual simulation symposium*, 2008, pp. 85–94.
19. Bononi L, Felice MD, D'Angelo G, et al. MoVES: A framework for parallel and distributed simulation of wireless vehicular ad hoc networks. *Comput Netw* 2008; 52: 155–179.

20. Roth C and Cointet JP. Social and semantic coevolution in knowledge networks. *Soc Networks* 2010; 32: 16–29.
21. Sandmann W. Discrete-time stochastic modeling and simulation of biochemical networks. *Comput Bio and Chem/ Comput & Chem* 2008; 32: 292–297.
22. Bader DA and Madduri K. A graph-theoretic analysis of the human protein-interaction network using multicore parallel algorithms. *Parallel Comput* 2008; 34: 627–639.
23. Li G, Yang H, Sun L, et al. The evolutionary complexity of complex adaptive supply networks: A simulation and case study. *Int J Prod Econ* 2010; 124: 310–330.
24. Cicirelli F, Furfaro A and Nigro L. Exploiting agents for modelling and simulation of coverage control protocols in large sensor networks. *J Syst Softw* 2007; 80: 1817–1832.
25. Backstrom L, Boldi P, Rosa M, et al. Four degrees of separation. In: *Proceedings of the 3rd annual ACM web science conference*, New York, NY, 2012, pp. 33–42.
26. Gjoka M, Kurant M, Butts CT, et al. Walking in Facebook: A case study of unbiased sampling of OSNs. In: *IEEE INFOCOM*, 2010, pp. 2498–2506.
27. Gjoka M, Kurant M, Butts CT, et al. A walk in Facebook: Uniform sampling of users in online social networks. *Comput Research Repos* 2009; abs/0906.0.
28. Bai-Da Z, Jun-Jie W, Yu-Hua T, et al. Betweenness-based algorithm for a partition scale-free graph. *Chinese Phys B* 2011; 20(11): 118903.
29. Kang U, Tsourakakis CE, Appel AP, et al. Radius plots for mining terabyte scale graphs: Algorithms, patterns, and observations. In: *SIAM international conference on data mining*, 2010, pp. 548–558.
30. Becchetti L, Boldi P, Castillo C, et al. Efficient algorithms for large-scale local triangle counting. *ACM T Knowl Disc Data* 2010; 4: 1–28.
31. Becchetti L, Boldi P, Castillo C, et al. Efficient semi-streaming algorithms for local triangle counting in massive graphs. In: *Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining*, 2008, pp. 16–24.
32. Plimpton SJ and Devine KD. MapReduce in MPI for large-scale graph algorithms. *Neuropsy* 2011; 37: 610–632.
33. Macambira TA and Guedes D. A middleware for parallel processing of large graphs. In: *Proceedings of the 8th international workshop on middleware for grids, clouds and e-science*, 2010, pp. 1–6.
34. Kang U, Tsourakakis CE, Appel AP, et al. HADI: Mining radii of large graphs. *ACM T Knowl Disc Data* 2011; 5: 1–24.
35. Kang U, Tsourakakis CE and Faloutsos C. PEGASUS: Mining peta-scale graphs. *Knowl Info Syst* 2011; 27: 303–325.
36. Malewicz G, Austern MH, Bik AJC, et al. Pregel: A system for large-scale graph processing. In: *Symposium on principles of distributed computing*, 2009, pp. 6–146.
37. Gregor D and Lumsdaine A. The parallel BGL: A generic library for distributed graph computations. In: *Proceedings of the parallel object-oriented scientific computing*, 2005, pp. 1–18.
38. Valiant LG. A bridging model for parallel computation. *Commun ACM* 1990; 33: 103–111.
39. Fujimoto R. Parallel discrete event simulation. *Commun ACM* 1990; 33(10): 30–53.
40. Kang U, McGlohon M, Akoglu L, et al. Patterns on the connected components of terabyte-scale graphs. In: *IEEE international conference on data mining*, 2010, pp. 875–880.
41. Kang U, Chau DH and Faloutsos C. Inference of beliefs on billion-scale graphs. *The 2nd workshop on large-scale data mining: Theory and applications*, 2010.
42. Perumalla KS and Seal SK. Discrete event modeling and massively parallel execution of epidemic outbreak phenomena. *Simul* 2012; 88(7): 768–783.
43. Barrett C, Eubank S and Marathe M. Modeling and simulation of large biological, information and socio-technical systems: An interaction based approach. In: *Interactive Computation*. Berlin: Springer, 2006, pp.353–392.
44. Bisset K, Chen J, Kuhlman CJ, et al. Interaction-based HPC modeling of social, biological, and economic contagions over large networks. In: *Proceedings of the winter simulation conference*, 2011, pp. 2938–2952.
45. Snijders TAB, van de Bunt GG and Steglich CEG. Introduction to stochastic actor-based models for network dynamics. *Soc Networks* 2010; 32: 44–60.
46. Bauer DW, Carothers CD and Holder A. Scalable time warp on Blue Gene supercomputers. In: *Workshop on principles of advanced and distributed simulation*, 2009, pp. 35–44.
47. Liu N and Carothers CD. Modeling billion-node torus networks using massively parallel discrete-event simulation. In: *Workshop on principles of advanced and distributed simulation*, 2011, pp. 1–8.
48. Chen D, Theodoropoulos GK, Turner SJ, et al. Large scale agent-based simulation on the grid. *Future Gener Comput Syst* 2008; 24: 658–671.
49. Chen D, Turner SJ, Cai W, et al. Synchronization in federation community networks. *J Parallel Dist Comput* 2010; 70: 144–159.
50. Yao Y and Zhang Y. Solution for analytic simulation based on parallel processing. *J Syst Simul* 2008; 20(24): 6617–6621.
51. Hou B and Yao Y. CommPar: A community-based model partitioning approach for large-scale networked social dynamics simulation. In: *IEEE/ACM 14th international symposium on distributed simulation and real-time applications*, 2010, pp. 7–13.
52. Barabasi AL and Albert R. Emergence of scaling in random networks. *Sci* 1999; 286: 509–512.
53. Backstrom L, Huttenlocher DP, Kleinberg JM, et al. Group formation in large social networks: Membership, growth, and evolution. In: *Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining*, 2006, pp. 44–54.
54. Brin S and Page L. The anatomy of a large-scale hypertextual web search engine. *Comput Networks ISDN* 1998; 30: 107–117.
55. Hethcote HW. The mathematics of infectious diseases. *SIAM Rev* 2000; 42: 599–653.
56. Barabasi AL. The origin of bursts and heavy tails in human dynamics. *Nature* 2005; 435: 207–211.

Author biographies

Bonan Hou is a PhD candidate at the School of Computer Science, National University of Defense Technology, China. His research interests include parallel discrete-event simulation and complex networks.

Yiping Yao is a professor both at the School of Computer Science and the School of Information System and Management, National University of Defense Technology, China. His research interests include modeling and simulation, and parallel and distributed simulations.

Bing Wang is an assistant professor at the Software Division, Beijing Institute of Systems Engineering, China. His research interests include modeling methodology, software engineering and parallel simulation.

Dongsheng Liao is an assistant professor at the School of Humanities and Social Sciences, National University of Defense Technology, China. His research interest is social networks.