Simulation Modelling Practice and Theory 77 (2017) 141–156

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

A hierarchical composite framework of parallel discrete event simulation for modelling complex adaptive systems

Feng Zhu[a,b], Yiping Yao[a], Wenjie Tang[a], Jun Tang[a,c,*]

[a] College of Information System and Management, National University of Defense Technology, Changsha, China
[b] Department of Computing, Imperial College London, London, UK
[c] Department of Telecommunication and System Engineering, Universitat Autònoma de Barcelona, Sabadell, Spain

ARTICLE INFO

Article history:
Received 15 January 2017
Revised 20 May 2017
Accepted 31 May 2017

Keywords:
Complex adaptive systems
Parallel discrete event simulation
Composite modelling framework
Three-level architecture

ABSTRACT

As complex adaptive systems(CAS) continue to grow in scale and complexity, and the need for system adaptability increases, systems modelling has become an essential concern. Parallel discrete event simulation became a preferred choice as logical process world view, which bridges complex system modelling and high-performance computing. To resolve the shortcoming of this world view identified with respect to modularity and scalability. A hierarchical composite modelling framework was proposed, which is a three-level architecture intended to support the composition and integration of sub-models. The bottom layer is simulation model component(SMC), which is not a model but implement some simulation-specific support functionality. The middle layer is logical process model(LP), which describes an agent which can react to the current situation by executing a sequence of SMCs. The top layer is CAS system model, which defines a CAS model consist of several LPs and also the interactions between these LPs. The hierarchical composite modelling process and parallel simulation execution strategy are discussed to support the modelling and simulation of a CAS. In order to verify its effectiveness, a complex social opinion system model is proposed based on this hierarchical composite modelling framework. The experimental results confirms the viability of utilizing multi-level architecture for simulating large scale complex adaptive systems.

1. Introduction

Whether designed to predict the spread of an epidemic, understand the potential impacts of climate change, or model the acoustical signature of a newly designed battle plane, computer simulation is an essential tool. By using simulation models that capture the complex behaviour of the real world, scientists can explore system dynamics that are too costly to test experimentally and too complicated to analyse theoretically.

Complex adaptive systems (CAS) [1] are such systems that have a large number of agents that adapt and interact. CAS problems are pervasiveness, such as encouraging innovation in dynamic economies, providing for sustainable human growth, predicting changes in global trade, understanding markets and so on. Mostly CAS share the features of modularity, adaptation, and evolution [2]. To understand their intricate dynamics it is often beneficial or necessary to use different kinds of

* Corresponding author Jun Tang, Email: tangjun018@gmail.com. Thank you very much!"?>.
E-mail address: tangjun018@gmail.com (J. Tang).

http://dx.doi.org/10.1016/j.simpat.2017.05.010
1569-190X/

models to represent each sub-system [3], then the structures and behaviours could be easy to adjust to change as the systems may continuously reconfigure themselves to adapt to different situations [4]. For example, a social ecological system typically evolves over time to adjust to the new environment. Therefore, to model this type of complex systems, a composite modelling framework is needed, as it greatly enhances the modelling capability.

A composite framework is an architecture or infrastructure intended to support and enable the integration and interoperation of individual sub-models [5]. It may consist of concepts, standards, control mechanisms, interfaces, and processes aimed to facilitate the efficient, and flexible assembly of systems from sub-models in a possible setting [6]. Component-based, standardized, multi-formalism modelling, system theory, and logical process world view are popular approaches for describing composite dynamic systems. Different modelling paradigms are suitable for various kinds of needs. Component-based simulation systems are immensely helpful to users who wish to build complex systems by composing existing software components, thereby shielding them from the underlying complexity of component resources. Discrete-event System Specification(DEVS) supports component-based modelling and simulation by emphasizing the theory of hierarchical modelling. The structure of input and output data is complicated as data for DEVS models are specified in terms of messages each defined in terms of port and value pairs. A standardized modelling approach is the High Level Architecture(HLA), which is aimed at handling interoperability needs with some limited capability for model composability as supported by the Object Model Template. Multi-formalism modelling approach composes different models using a model which handles the differences between modelling formalisms, but common forms of data transformation are inherently supported.

On the other hand, CAS can be modelled by a vast numbers of agents that interact by sending and receiving messages. The agents produce scores of interactive messages, and the actions of agents in a CAS usually depend on the messages they receive. Moreover, the complexity of CAS rises above the handling capacity of monolithic sub-models[7,8]. Therefore, the composite modelling framework is also needed to cope with the requirement for scalable modelling capability and high-performance simulation power.

Parallel discrete event simulation(PDES) became a preferred choice for studying complex systems as logical process worldview, which bridges complex system modelling and high-performance simulation [9]. A PDES program can be viewed as a collection of logical processes(LP). For example, a physical process such as a battle plane or a water station is modelled by a LP, and interactions between physical processes are modelled by scheduling events [10]. Each event contains a timestamp that represents a point in simulation time at which the state of a LP changes. The state variables that capture the state of the system being modelled changes when a corresponding event computation occurs. Recently studies into the parallel simulation of LP-based models has usually attempted to overcome the performance bottleneck. There has been useful work on algorithm optimization, but few effective methods for composite modelling methodology of CAS. Research on modelling CAS using LP paradigm currently suffers from: (1) event scheduling and event processing within a LP couples tightly, so that a LP is hard to achieve flexible composition of sub-models; (2) LP couples tightly, so that it is difficult to enable the efficient, and flexible assembly of a CAS-oriented simulation system from LPs.

To resolve the above problems, this paper proposed a hierarchical composite modelling framework, which is a three-level architecture intended to support the composition and integration of sub-models. The hierarchical composite modelling process and the parallel simulation execution strategy is discussed to support the modelling and simulation of a CAS. To verify its effectiveness, a complex social opinion system model(SOSM) is proposed based on this hierarchical composite modelling framework. The experimental results confirms the viability of utilizing multi-level architecture for simulating large scale complex adaptive systems. The main contributions of this paper are as follows:

(1) Encapsulate a group of action rules of an agent into a SMC. It enables flexible assembly of an agent from SMCs; thus, an agent reacts to the current situation by executing a sequence of SMCs.
(2) Decouple the interaction between agents. An agent only contains the behaviour processing logic. The interaction between agents implements by configuring the interactive relationship in a possible setting.
(3) Mapping an agent to a LP and mapping the interactions between agents to scheduling interactive events between LPs, which enables the high-performance collaborative execution of a CAS model on existing PDES platforms.

The remaining of this paper is organized as follows. Section 2 gives the background and related works. Section 3 introduces the three-level architecture and the formal definition of sub-models in each level. Section 4 depicts the hierarchical composite modelling process for CAS and the execution of a CAS model. Section 5 analyses a case study of social opinion system and illustrates the merits of our framework. Finally, our conclusion will be made with an indication of the future work in Section 6.

2. Background and related work

There has been useful work on simulation platforms of multi-agent modelling for complex systems and complex networks. GAMA [11] aims at providing field experts, modellers, and computer scientists with a complete modelling and simulation development environment for building spatially explicit multi-agent simulations. RepastHPC [12] is a useful and useable agent-based modelling and simulation system explicitly focusing on larger-scale distributed computing platforms. It allows the use of different structures (Networks, Grids) which can be coupled in a same simulation to represent different types of interactions. However, one of the limitations in RepastHPC is no communication is allowed between remote agents.

SARL [13] is a general-purpose agent-oriented programming language. SARL provides a set of agent-oriented first-class abstractions directly at the language level. SARL models can be executed on Janus platform, and it also can be linked with other existing agent platforms and frameworks. Janus [14] integrates and benefits from the new patterns of Object-oriented programming like Inversion of Control, event-driven communication, distributed objects, etc.

About layer architecture of agent-based simulation, D'Angelo et al. [15] proposed a two level simulator where Level 0 simulator and Level 1 simulator communicate with each other through the use of a TCP connection. It is available for simulating large-scale decentralized and heterogeneous scenarios (e.g. mobility models, wireless/wired communications and so on). Holonic structures [16] offer a powerful abstraction for modelling complex systems. Holons are composed of other holons, referred as super-and sub-holons respectively. Holonic MAS represent an attempt to tackle a general problem of how to treat collections of agent as higher-order entities. MaMA-S [17] is a methodological approach for the creation of simulation models of complex and distributed systems. It facilitates the modelling and the simulation of decision-making processes, but synchronization of the simulation models still need to be resolved.

In a logical process world view, a simulation system is carried out by having some LPs each keep track of the state of different parts of the system. Based on this world view, several distinct types of modelling framework have been developed. The mechanisms used to implement a framework may vary in protocols, standards, interface definitions, data translators and so on. Several frameworks for model composition in discrete event simulation are discussed as follow:

DEVS framework is founded on system-theoretic principles including component-based hierarchical modelling [18]. The framework defines two types of models - atomic model and coupled model; atomic models are the basic modelling constructs whereas coupled model represents a group of atomic or coupled modes. The framework supports hierarchical model development through the use of one coupled model as a fundamental component in another coupled model [19]. Based on DEVS, several modelling and simulation platform have been developed, such as DEVSJAVA [20], CELL-DEVS [21] and DEVSNET [22]. However, the hierarchical assembly of DEVS-based system models is implemented by using input/output ports and message couplings; thus the complexity will increase obviously when the system modeller adds a new layer of a system model.

Event Graph [23] modelling paradigm can be used to build discrete-event simulation model, but it does not facilitate composability of sub-models. LEGO [24] was designed to encapsulate an Event Graph as an object using the listener pattern as a means of loosely connecting them. Each LEGO is responsible for the events and state transitions that modify its state variables and produce its state trajectories. Viskit [25] is a graphical editor for creating, editing, and composing discrete event simulation models using Event Graphs and the LEGO framework. DEG [26] extends classical event graphs for component-based models in discrete event simulation. PEG [27] extends classical Event Graph towards a formal specification for LP-based PDES models. However, the Event Graph-based composite modelling frameworks as mentioned above did not refer to hierarchical assembly of different LPs from low-layer components.

Kesaraju [28] proposed a sequential simulation framework that integrates process-driven and event-driven approaches. Modellers can manage the complexity of real-world systems through process-driven orientation while retaining the control logic through event-driven orientation. Rizvi [29] proposed a LP-based simulation model for distributed simulation systems, which provides the internal architecture of each LP and its coordination with the other LPs through some communication protocols.

In summary, the above discrete-event composite modelling frameworks either did not refer to assembly of different LPs from low-layer components or lack of a hierarchical architecture to clearly depict how to compose LPs to build a complex system model.

3. Three-level architecture for CAS model

Combining different sub-models poses a variety of challenges depending on the system being modelled. A real system can be described with a hierarchical style. The System Level where each identified a real system consist of various agents interacting with others. The Agent Level where each identified the agent of the CAS system is associated with a class of possible agent templates. The Component Level where each identified the component of the agent is associated with a class of possible component templates. Partitioning a CAS model into layers, each of which consists of a set of sub-models, is a crucial step in a hierarchical modelling framework. In this paper, a three-level representation is proposed for CAS models, in particular, steps 1 through 3. These steps convert the real CAS to a CAS model as shown in Fig. 1. In a layered architecture, one or more CAS models are built up from lower-level components. The bottom layer is SMC (Simulation Model Component), which is not a model but implement some rule-specific support functionality. The middle layer is LP (Logical Process Model), which defines that an agent can react to the current situation by executing a sequence of SMCs. The top layer is CSM(CAS System Model), which defines that a CAS model consists of several LPs and also the interactions between these LPs.

3.1. Simulation model component

A SMC is a component that can be integrated and used with other components only through well-defined interfaces. SMC defines the details of the component's interface and structure that used to implements the subroutines consist of groups of action rules. The service interfaces of a component define a set of methods. The external service interfaces are provided to be invoked by agent frameworks. A SMC changes the values of the variables that describe the agent state. A SMC can be

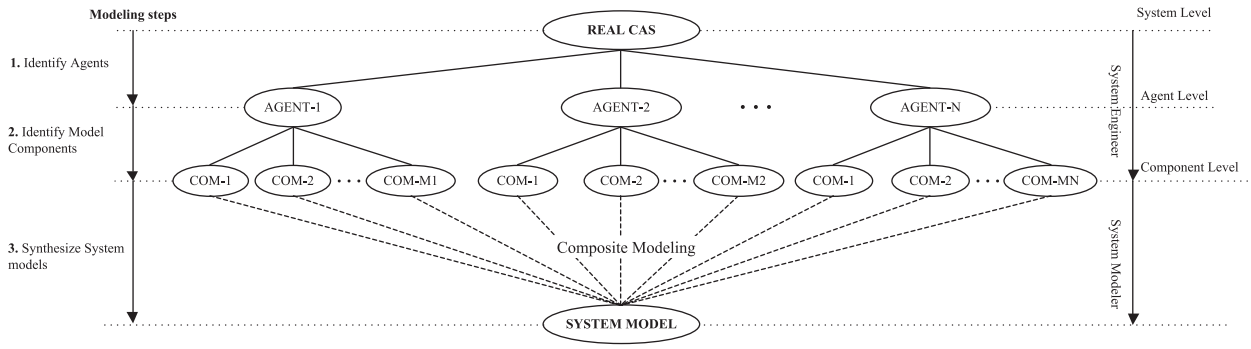

Fig. 1. Three-level representation for CAS.

represented as a nine-tuple.

$$SMC =< Children, T, X, Y, \Delta t, \Gamma, \Omega, \Psi, \Phi > \tag{3-1}$$

Where

Children is the set of low-level SMCs;
T is the set of simulation time;
X is the set of parameters denoting the possible inputs;
Y is the set of parameters denoting the possible outputs;
Δt is the real number that represents the minimal time of SMC iteration;
Γ is the action rule function set of an agent;
$\Omega = \{(t^-, x)|t^- \in T, x \in X\}$ is the input sequence, t^- is the time corresponding to the input sequence;
$\Psi = \{(t^+, y)|t^+ \in T, y \in Y\}$ is the output sequence, t^+ is the time corresponding to the output sequence;
$\Phi = \{f|f_i,\ f_d,\ f_p,\ f_o\}$ is the external service interfaces set, which is used to be invoked by an agent model, where

- $f_i = input(t^-, x_i)$, which is the input interface. It provides inputs for the SMC during its execution. t^- is the current simulation time and x_i represents the inputs;
- $f_d = driven(t^-, x_d)$, which is the dynamic data-driven interface. It provides dynamic inputs for the SMC during its execution. t^- is the current time and x_d represent the dynamic inputs.
- $f_p = process(t^+)$, which is the business processing interface for agent behaivour rules. After its execution, the local simulation time of an agent will advance to t^+.
- $f_o = output(t^+, y_o)$, which is the output interface. It outputs the results after business processing. t^+ is the current simulation time and y_o represents an output sequence after the SMC's execution.

3.2. Logical process model

An agent in CAS can be modelled by a LP *class* which is composed of an initial method, several event process procedures, and a series of variables. Each initial function gives the initial value for the LP and then schedule some necessary events for simulation execution. A LP changes the values of the variables that describe the agent state through processing events. The computation in the event process function is implemented by invoking a series of SMCs. Processing events will change the states of a LP and schedule some new events. A LP model can be represented as a ten-tuple.

$$LP =< I, Q, M, E_\alpha, V, S_\alpha, C, T, O, B > \tag{3-2}$$

Where

I is the initial method;
Q is the set of all possible states that the LP can be in;
M is the set of SMCs that possibly used to be invoked by the event process procedures;
E_α is local event set, which contains the events that are scheduled by the same LP;
$V = \{< e, invoke < m, i, c, b >> |e \in E_\alpha, m \in M, i \in N^+, c \in C, b \in B\}$ is the invoked sequence for SMCs by the event process procedure related to the simulation event e, where i represents the invoking index;
$S_\alpha = \{< e, schedule < e_{next}, c, t, o, b >> |e, e_{next} \in E_\alpha, c \in C, \Delta t \in T, o \in O, b \in B\}$ is local event scheduling relationship which means the simulation event e will schedule event e_{next} in the same LP with incremental time Δt, condition c, priority o, and input parameters b;
$C = \{c_e|c_e : S \to \{0, 1\}, \forall e \in E_\alpha\}$ is condition set for scheduling local events;
$T = \{\Delta t_e|\Delta t_e \in R_0^+, \forall e \in E_\alpha\}$ is incremental time set for scheduling local events;
$O = \{o_e|o_e \in Z_0^+, \forall e \in E_\alpha\}$ is priority set for processing local events;
$B = \{b|b \subseteq Q\}$ is parameter set of denoting the possible inputs.

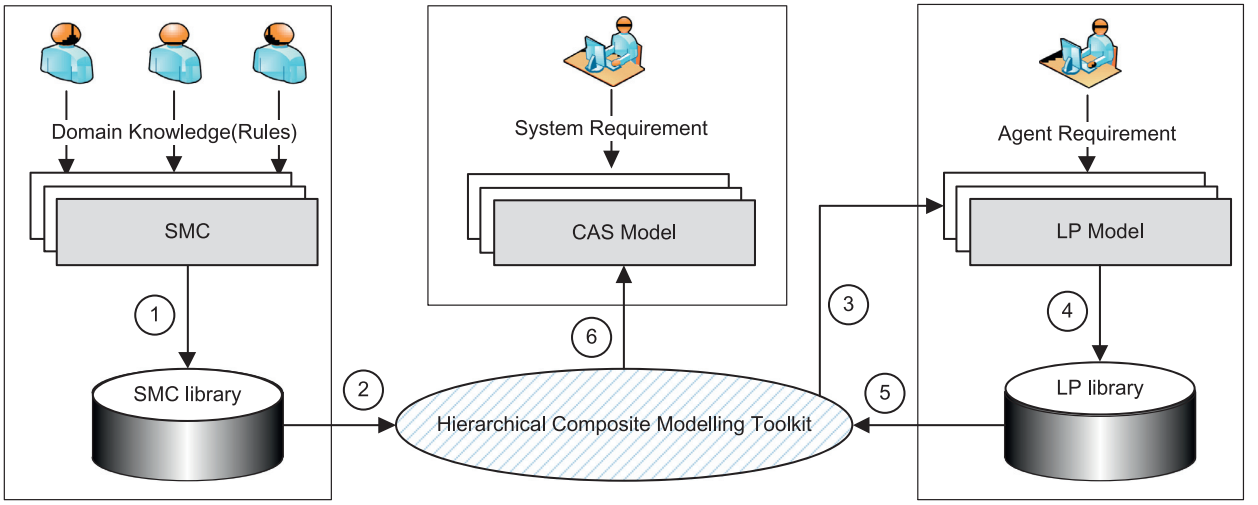

Fig. 2. Hierarchical composite modelling flow for CAS.

3.3. CAS system model

A CAS model can be described as the composition of several LPs. The interaction between LPs is modelled by scheduling events. The CSM layer describes the LPs that composed of the CAS model and the interactive relationship between these LPs, which can be represented as an eight-tuple

$$CSM = < \Omega, E_\beta, S_\beta, L, C, T, O, B > \tag{3-3}$$

Where

Ω is the set of LPs used to assembly of a CSM;

E_β is interactive event set, which contains the events that are scheduled by other LPs;

$S_\beta = \{< e, schedule < e_{netx}, l, c, \Delta t, o, b >> | e, e_{netx} \in E_\beta, l \in L, c \in C, \Delta t \in T, o \in O, b \in B\}$ is the interactive event scheduling relationship which means the simulation event e will schedule event e_{next} belonging to another LP with incremental time Δt, condition c, priority o, and input parameters b;

L is decision-making function set, which can be used to decide the simulation event e_{next} of which LP will be schedule by simulation event e;

$C = \{c_e | c_e : S \to \{0, 1\}, \forall e \in E_\beta\}$ is condition set for scheduling interactive events;

$T = \{\Delta t_e | \Delta t_e \in R_0^+, \forall e \in E_\beta\}$ is incremental time set for scheduling interactive events;

$O = \{o_e | o_e \in Z_0^+, \forall e \in E_\beta\}$ is priority set for processing interactive events;

$B = \{b | b \subseteq Q \cap Q \in \Omega\}$ is parameter set denoting the possible inputs.

4. Hierarchical composite modelling and simulation

The hierarchical composite modelling framework is aimed to provide a CAS model where a collection of SMCs can be coupled to create new and useful capabilities and automate the process as much as possible. Based on the three-level architecture, the hierarchical composite modelling and simulation for CAS includes two parts: hierarchical composite modelling for CAS and parallel simulation execution of a CAS model.

4.1. Hierarchical composite modelling for CAS

Hierarchy and decoupling are useful approaches to composing a CAS model. In this paper, assembly of a CAS model can be divided into the following steps(see Fig. 2):

- Step(1): SMCs that developed by domain experts are added into the SMC library. The SMC should be built in agreement with the formal definition discussed in Section 3.1. Structured descriptions can be used to guide the processes of selecting SMCs for a particular purpose and determining if a set of SMCs can be composed [30], thus the detail of a SMC description shown in Table 1 should be given.
- Step(2–4): Required SMCs are selected to construct LPs, and the formal definition of LP models discussed in Section 3.2 cannot be violated. In practice, it is often assumed that any SMC that had been placed in the library is valid, but the validity of SMCs does not imply that a composition of them can be considered to be valid. Here validating a composition of SMCs can use traditional validation methods, such as comparing output data to baseline data and can

Table 1
Knowledge description of a SMC.

Content	Description detial
ID	Unique identification of a SMC.
Domain	Application domains should be considered when characterizing the context for a SMC, for example, hydromechanics, aerodynamics, mechanical control, and so on.
Purpose	SMC can be considered as replaceable building blocks of an agent. It includes the problem that the SMC solves.
History	History is a characteristic of a SMC that reflects its stability and maturity. When a SMC is first introduced, there is a high risk associated with its usage.
Parameter	This records the parameter description of the external service interfaces, including *name, type, unit, identification* and *funtion*, especially for user defined parameter types. The parameters are passed to the SMC when an agent schedules it.

Table 2
Knowledge description of a LP model.

Content	Description detial
Class	Each *agent* is associated with a class of possible LP templates.
Domain	Application domains should be considered when characterizing the context for a LP, for example, behaviour control agent, motion simulation agent, target selection agent, and so on.
History	LP belongs to the middle-level system resource. The history of a LP is a characteristic of the LP that reflects its stability and maturity. When a LP is first introduced, there is a high risk associated with its usage.
Initial	Initial data description is used for the initialization of a LP. At the beginning of simulation execution, the initial method of each LP is scheduled to set the initial value for the variables of each LP.
Subscribe	Subscribe is a one-way relation between LPs and it is for declaring the interaction data interests in LPs. The list of attributes that a subscriber LP requires to be notified about when a LP is generated or updated during simulation execution.
Publish	A publish association between two instances of LPs indicates that one LP is capable of generating interactive data which is interested by the other LP. The list of the interested attributes of the LP will be produced and updated during simulation execution.
Rules	A set of SMCs composes the behaviour rules. An agent modelled by a LP can react to the current situation by executing a sequence of rules.

also exploit the structure of the composition, such as automatically analyzing the domains of validity for each SMC with the data they are receiving from other SMCs in the compositon [31]. To facilitate to select the LPs and configure the interaction between different LPs, we define the knowledge description of each LP in Table 2. Here interaction is the ability of distinct LPs to share semantically compatible information.

- Step(5-6): Compose different LPs into a CAS model which is described by the formal definition discussed in Section 3.3. A event scheduling scheme was used to deliver the interactions among LPs. It requires that the modellers involved employ compatible semantics and standard interpretations of the information they exchange.
- Step(6): Generate object-oriented code: for each *LP*, we build a *class* in C++ object-oriented language. Each internal method of this class corresponds to an event processing function. The class of *LP* also contains a set of internal variables: there are several variables for the initial method of *LP*; there are also several variables for each event processing procedure of *LP*. All these variables are persistent meaning they maintain their values across the execution of the different functions.

Fig. 3 illustrates the same LP is reused in different CAS models. In this case, CSM_1 (CAS Model 1) and CSM_2 (CAS Model 2) both contain the LP model *AgentC*. In CSM_1, *AgentC* schedules the event of *AgentB*. In CSM_2, *AgentC* schedules the event of *AgentE*. The *ConditionSet* and [*paralist*] are derived from the computational results of SMCs execution. The C++ code of event processing function *EventHandle*() can be generated automatically according to the configuration of *AgentC* and the interaction between *AgentC* and *AgentB* in CSM_1. Meanwhile, the C++ code of event processing function *EventHandle*() can be generated automatically according to the configuration of *AgentC* and the interaction between *AgentC* and *AgentE* in CSM_2. Compare with the generated codes of different event processing functions *EventHandle*() in *AgentC*, the event processing and local event scheduling in *AgentC* is the same, only the interactive event scheduling in *AgentC* is different. That means with our composite modelling framework, event scheduling and event processing decouples in a LP model. A LP can be flexible composed from a set of SMCs and a set of interactions, and the behaviour(a set of SMCs) of the specific LP can be reused in different CAS models.

4.2. Parallel simulation execution of CAS model

In the logical process world view, the simulated system is partitioned into a set of LPs that communicate with each other by sending and receiving time-stamped events. A CAS model composed of several types of LPs can be grouped into several

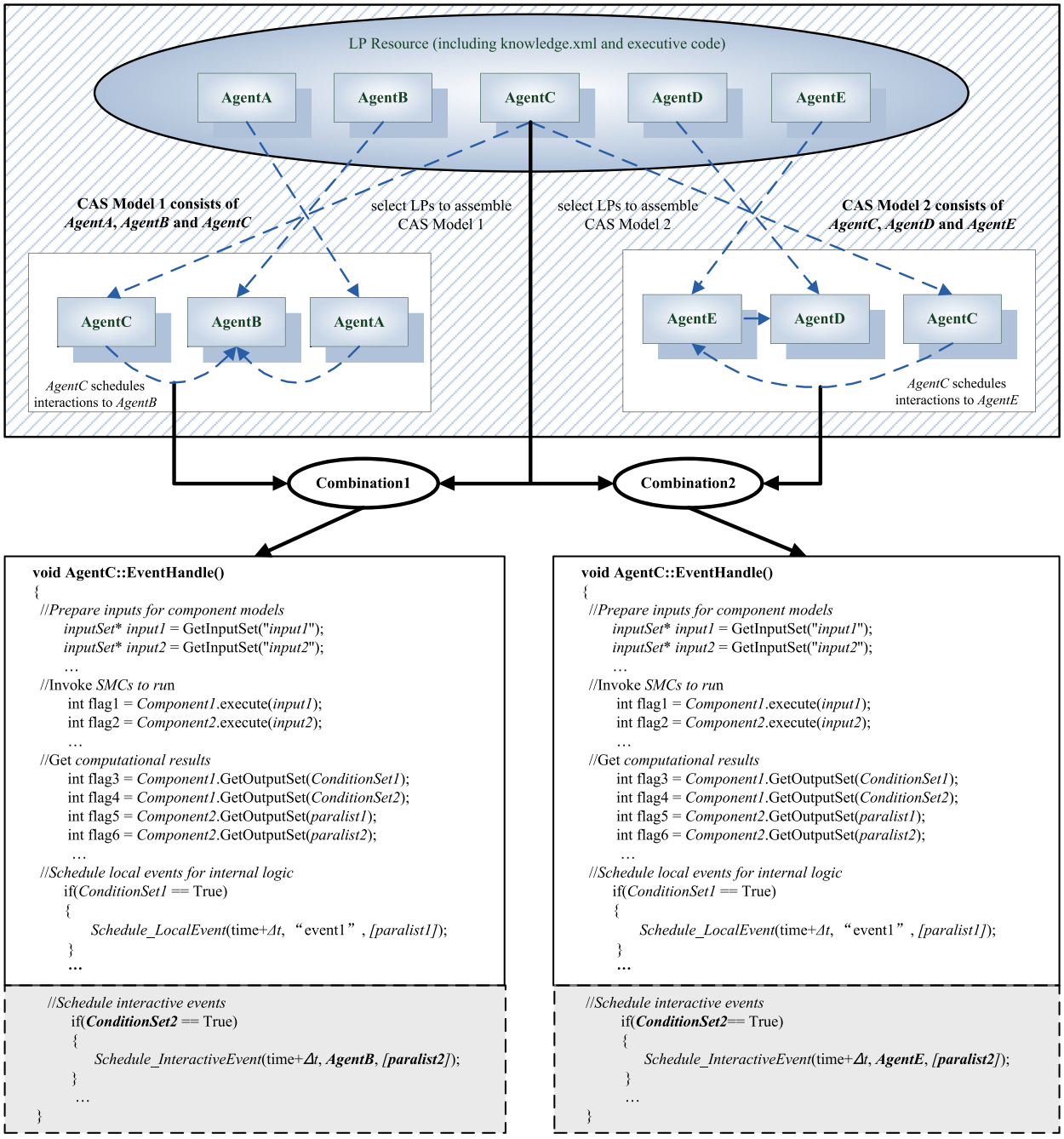

Fig. 3. A simple example of hierarchical composite CAS model.

partitions for parallel execution(see Fig. 4). The *InitialDataSet* obtained by parsing a lot of *SampleFiles* is used to initialize the states of these LPs. During the simulation execution, the LPs are grouped into several partitions, each of which is mapped onto a CPU core. To ensure that all of the events on different CPU cores are performed in accordance with time stamp order, events are required to execute by LPs such that local causality constraint requirement must not be violated [32]. To do that, each LP must know which events are safe to process. Since all LPs do not have a consistent view of the entire system, LPs must exchange simulation time to synchronize with each other. Synchronization refers to the coordination of events that are running simultaneously on different CPU cores. Nowadays, most PDES platforms provide two classes of time management algorithms to synchronize simulation execution [33,34], which are conservative and optimistic. Conservative algorithm allows for the simultaneous execution of events only when the CAS model can guarantee that the events are not causal invalid, but the strict ordering constraint can lead to deadlocks that must be discovered and broken. Optimistic algorithm relaxes the

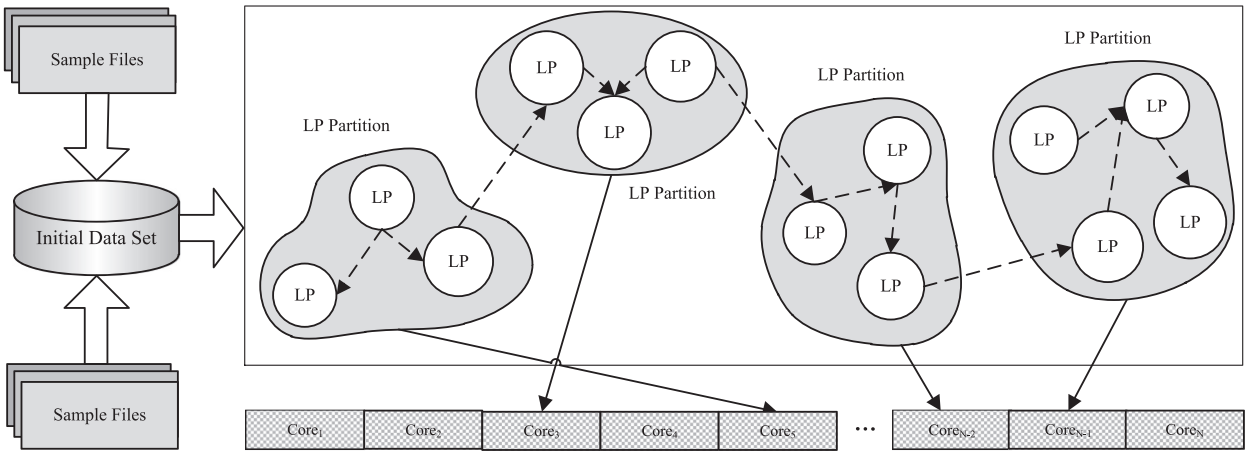

Fig. 4. Mapping LP partitions into multi-core computing platform.

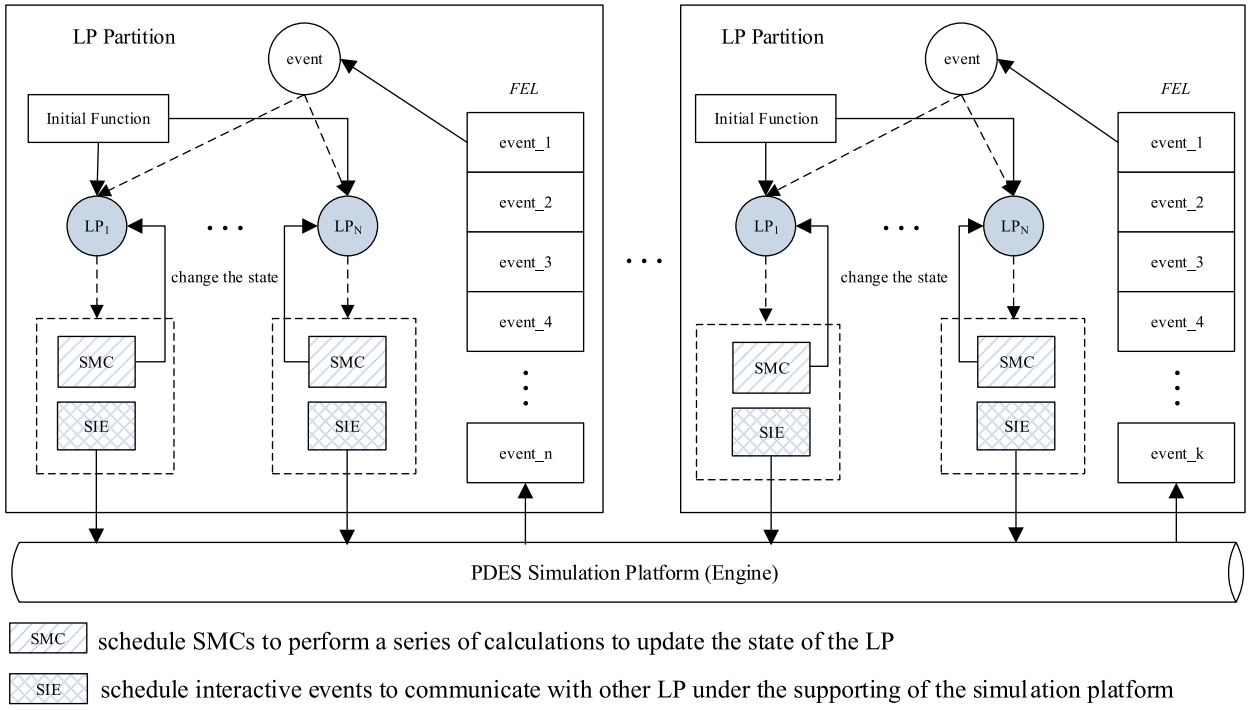

Fig. 5. Parallel simulation execution of CAS model.

strict enforcement of event stamp order. When causal violations occur, some recovery or repair mechanism may be used to restore the simulation to an earlier state. Here we do not work on the optimization for time management algorithms, but focus on how to execute the hierarchical CAS model based on PDES simulation platform.

During the simulation execution of a CAS model, event processing and event scheduling of different LPs drives the evolution of the model (see Fig. 5). In the beginning, the simulation platform first processing the initial $event_0$, which is usually defined by an initial method. The initial method is responsible for the initialization of the state variables of the LPs. During the execution, each LP partition will build a *FEL*(Future Event List) to receive the event messages from LPs. At the same time, the simulation platform will process the event with minimum time order from the *FEL*. Each event message corresponds to an event processing procedure which belongs to a specific LP. The execution of an event processing procedure of a LP includes two different routines:

(1) Schedule SMCs to perform a series of calculations to update the state of the LP. Fig. 6 shows the execution flow of a SMC, which is described in the following: 1) After the input sequence prepared, calling input interface $input(t, x_i)$ to provide input data for a SMC. 2) If dynamic data is given, calling dynamic data-driven interface $driven(t, x_d)$. The

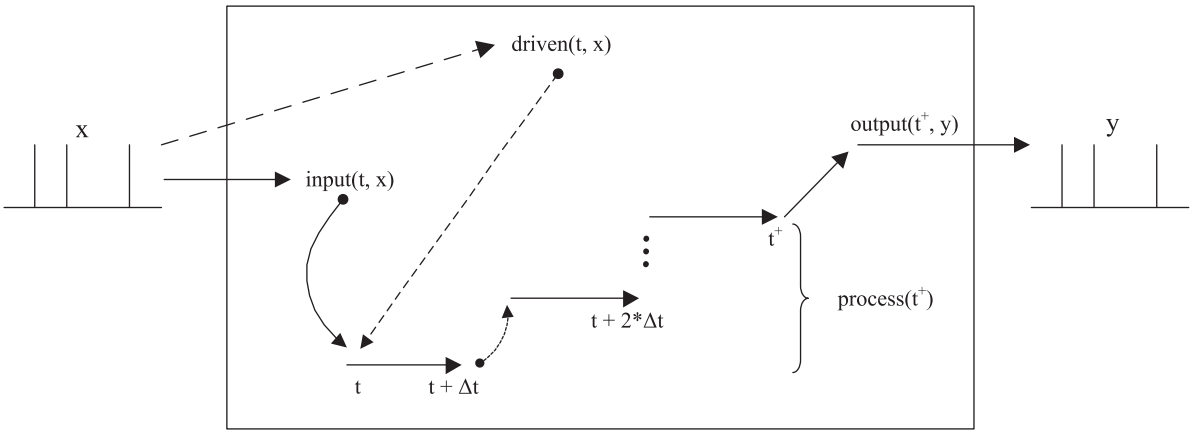

Fig. 6. Execution flow of a SMC.

dotted lines represent the dynamic data is not always needed for computation each iteration. 3) The core of a SMC is in the business processing interface $process(t^+)$, which implements some important action operations defined by the SMC. If the local simulation time $t \in T$ satisfies $t^+ < t + \Delta t$, the SMC will not be scheduled until the next input arrives. Otherwise, the SMC advances the local simulation time in an iterative manner. 4) Call output interface $output(t^+, y_o)$ to output the results after business processing.

(2) Schedule interactive events to communicate with other LP under the supporting of the simulation platform, which notify the relative LP to process the event at what time and with what parameters. Scheduling interactive events uses standard socket ports and Internet communication protocols [e.g., Transmission Control Protocol/Internet Protocol (TCP/IP). It is not an extended version of some other existing message passing protocols, such as Message Passing Interface (MPI), etc.

Priority constrain principle: The execution priority of a SMC should be higher than that of the event processing function which invokes the SMC, i.e., if an event processing function start a thread to run the SMC, the performance of the event processing function will be stopped until the SMC-thread finish its execution.

Proof: Simulation execution with the Priority Constrain Principle is able to guarantee the correct execution of a CAS model.

Let ξ represents a LP, and e denotes a event processing function of ξ, which invokes several SMCs. Let Θ denotes the SMC set of e. The execution of the SMCs will probably change the state of ξ. Meanwhile, after the SMCs invoked by e finish the execution, the corresponding events including local events and interactive events will be scheduled. Thus, there may be two kinds of data dependences, the state of the LPs may be caused error.

(1) The dependence between the SMCs.
Let $\exists A, B \in \Theta(A \xrightarrow{a} B)$ represents the execution of SMC A should prior to B, where a denotes the dependent variable between A and B.

$$\exists smc, smc^+ \in \Theta(smc \xrightarrow{a} smc^+) \tag{4-1}$$

(2) The dependence between the SMC and the event scheduling routine.
Let $\exists A \in \Theta, \exists E \in (S_\alpha^\xi \vee S_\beta^\xi)(A \xrightarrow{a} E)$ represents the execution of SMC A should prior to the execution of event scheduling routine E, where a denotes the dependent variable between A and E.

$$\exists smc \in \Theta, \exists sie \in (S_\alpha^\xi \vee S_\beta^\xi)((smc \xrightarrow{a} sie) \vee (sie \xrightarrow{a} smc)) \tag{4-2}$$

Hence, if the execution of the event processing function is error, one of the followings should appear:

(1) The execution of the SMCs violates the dependence (4-1). Let $\exists A, B \in \Theta(A \Rightarrow B)$ represents the execution of SMC A prior to B in actual execution.

$$\exists smc, smc^+ \in \Theta((smc \xrightarrow{a} smc^+) \wedge (smc^+ \Longrightarrow smc)) \tag{4-3}$$

However, this violates the principle that the execution of SMCs should comply with the input/output dependence.

(2) The execution of the SMCs and the event scheduling routine violate the dependence (4-2). According to Fig. 5, the execution of SMC is always preceded by the events scheduling routine in the same event processing function. Let $\exists A \in \Theta, \exists E \in (S_\alpha^\xi \vee S_\beta^\xi)(B \Longrightarrow A)$ represents the event scheduling routine E finish its execution prior to the execution of SMC A in the actual performance,

$$\exists smc \in \Theta, \exists sie \in (S_\alpha^\xi \vee S_\beta^\xi)((smc \xrightarrow{a} sie) \wedge (sie \Longrightarrow smc)) \tag{4-4}$$

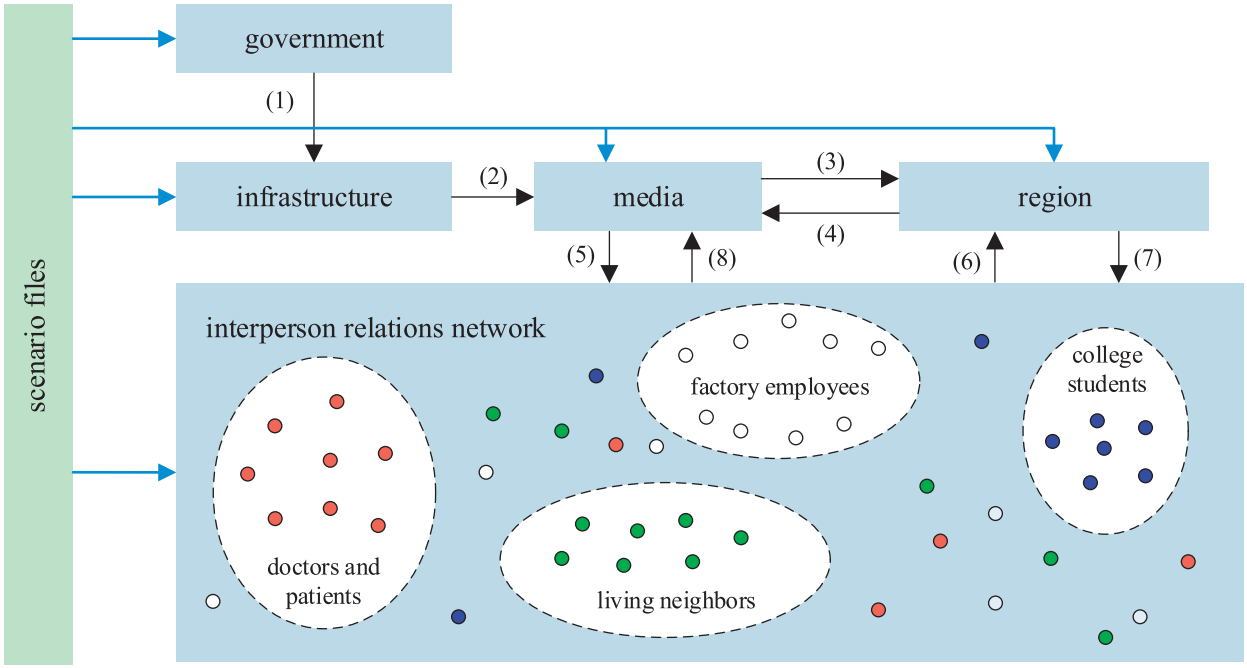

Fig. 7. Agents and interactions in SOSM.

Some event scheduling routine uses a variable a before that the SMC change it to a^+

$$\exists smc \in \Theta, \exists sie \in (S_\alpha^\xi \vee S_\beta^\xi)(\xrightarrow{a} sie \Longrightarrow \ldots \Longrightarrow smc \xrightarrow{a^+}) \tag{4-5}$$

Because the execution of SMC prior to the events scheduling routine, the reason of (4-5) may be that there is at least one SMC starting multiple threads to run, and the priority of these threads is lower than that of event processing function. Thus, it violates Priority Constrain Principle. Therefore, depending on reduction to absurdity, simulation execution with Priority Constrain Principle can guarantee the correct execution of LPs.

5. Case study: social opinion system

Social opinion plays a significant role in the political sphere [35]. These have registered the distribution of opinions on a wide variety of issues, have explored the impact of special interest groups on election outcomes and have contributed to our knowledge about the effects of government propaganda and policy [36]. Social opinion system is a typical complex adaptive system, with a significant overall emergence. The social opinion system model uses the multi-agent modelling method to describe the dynamic behaviour of the national critical infrastructure, and to model the basic behaviours and interactions of individuals, organizations and institutions in society. Through this model, we can study the impact of the destruction of national critical infrastructure on political, economic and social life, the impact on the emergence and dissemination of public opinion, and the impact on international political ecology. In our case study, we study the self-organization process and the emergency of social public opinions by modelling the behaviour of a single entity, the interaction between these entities, and the interaction between entities and environments.

The social opinion simulation model(SOSM) consists of the following LPs (see Fig. 7): (1) *government*, which takes measures to solve the destruction of infrastructures after receiving the opinion-decision results. (2) *infrastructure*, which is the critical resources that support the normal life in the city, e.g. the water station or the power station. (3) *region*, which is an abstraction of the city structure. The city can be divided into several regions, such as the hospital, the college, the factory and the living area. They change their supply requirements according to the scheduled stimulus events, which affects the people in the corresponding living area. (4) *media*, which publish the information such as the destruction of an infrastructure and the measures taken by the *government*. It should be noted that different *medias* influence different *individuals*. (5) *individual*, which is the major research object in SOSM. *individuals* directly releases his/her opinion after the comprehensive analysis of the destruction of the *infrastructure*, the opinions from other *individuals*, and the measures taken by the *government*.

The interactions occur among the above LPs shown in Fig. 7: (1) *government* interacts with *infrastructure* when the external stimulus event is scheduled according to the simulation scenario; (2) *infrastructure* interacts with *media* when the *infrastructure* is destroyed; (3-5) *media* publishes the stimulus events. According to the region of *individuals*, different *indi-*

viduals receive events from different *media*; (6-7) the interactions occur among the *individuals* in the same region; (8) *media* periodically counts the political opinion tendencies of *individuals*.

Interpersonal relationships usually occur between two individuals with more common factors. Often *individuals* have similar characteristics of age, gender, educational level, values, political attitudes and geographical location. These common features are the factors that need to be considered by the interpersonal network. In the interpersonal network model, we will consider the common factor as the "common" weight of the relationship network between individuals. In the process of generating interpersonal networks, each common factor of the individual is quantified. For example, for a culture that includes a set of factors, we sort each category and set an integer value to quantify it. For age, we quantify the age values directly. For geography, we use coordinates. The common weight between two individuals i and j is calculated using the following formula:

$$\omega(i, j) = \sum_{k=1}^{m} \alpha_k \rho_k |y_{ik} - y_{jk}| \tag{5-1}$$

Where m denotes the number of factors; k denotes the label of a factor; α_k represents the weight of the kth factor ($0 < \alpha_k < 1$); y_{ik} represents the kth factor of the individual i; ρ_k denotes the normalized distance between a certain factor k of the entities i and j, such as the normalized distance of the geographical position.

According to the goals of *individuals*, SOSM simulates the evolution of political tendency by scheduling LPs' execution. In the evolutionary process, *individuals*' opinions will aggregate according to their tendencies . At the same time, the aggregation of public opinion has dynamic characteristics, and the occurrence of certain stimulus events leads to the split of aggregation. In this way, through the stimulation of the external stimulus events to the *individuals* and the interaction between the *individuals*, the specific trend of public opinion of a region is simulated.

5.1. Hierarchical composite modelling

In SOSM, the opinion decision of the *individual* is determined by three kinds of SMCs : (1)*development_index_component*, when the individual's *development_index* decreases, it is more likely to oppose *Tendency_A*, and when the individual's *development_index* rises, it is more likely to support *Tendency_A*, but ultimately opinion also consider other indicators. (2) *security_index_component*, which computes the *security_index* when a danger seems to a real probability, such as the destruction of a water station or a power station. When the *security_index* of *individual* decrease, he/she may support *Tendency_C*, otherwise he/she will oppose *Tendency_C*; (3) *decision_making_component*, which comprehensively analyse the individuals' decisions of *government*'s measure depending on the results of *security_index_component* and *development_index_component*. The decision-making of an individual is also influenced by the opinions of other individuals; thus the individual will put the opinions of other individuals together to consider the situation.

To illustrate the flexible assembly of SOSM, we assemble the SMCs developed by domain experts to construct the LP (*individual*), and then integrate the above LPs to build the SOSM for different simulation requirements. In LP design mode(see Fig. 8(a)), a new LP is created by drawing the Event Graph with several message-ports and filling in parameters, so that the simulation modeller need not be a sophisticated programmer. The initial event (oval-shape with tab Init) and several events (circular-shape with tab LE and SE) constitute the LP, *InidividualEntity*. Three circular-shapes in the LP graphical editor respectively represent *development_index_component*, *security_index_component*, and *decision_making_component*. The quadrilateral-shape and the triangle-shape represent the input and output port respectively, which are utilized to depict the event scheduling channel with other LPs. These events are connected with bidirectional active lines which represent the local event scheduling relationship in the same LP. In LP composition mode(see Fig. 8(b)), LPs are hooked together to create a complex system model. A rectangular block with a small picture represents a LP. The LPs can be dragged and dropped from the left LP resource panel to construct the SOSM. They are connected with bidirectional active lines which represent the interactive event scheduling relationship between different LPs.

5.2. Parallel simulation execution

To illustrate the scalability of SOSM, the LPs used to assemble SOSM including the *government*, the *infrastructures*, the *regions*, the *medias* and the *individuals*, are divided into several partitions, and each LP partition will be dispatched to a CPU core. The underline simulation platform is SUPE [8]. All experiments will run on a single computing node of a Linux Cluster. The node is equipped with 2.93 GHz Intel Xeon Processors X5670, with 6 cores. The operating system is Redhat Enterprise Server 5.5, with kernel 2.6.18. The experimental configuration is shown in Table 3. The experiment set up a simulation time corresponds to the physical time of 10 min to study the actual system of public opinion changes within the next 10 days, so each simulation to promote the logical simulation time of 1440. The *strangers* means the *indivials* in different *environments*. Interaction occur between *strangers* at a certain probability.

5.2.1. Emergence of SOSM

The SOSM consists of the following sub-models: the national infrastructure models and the social group behaviour models. Among them, the national infrastructure models include communication network model, power network model, traffic

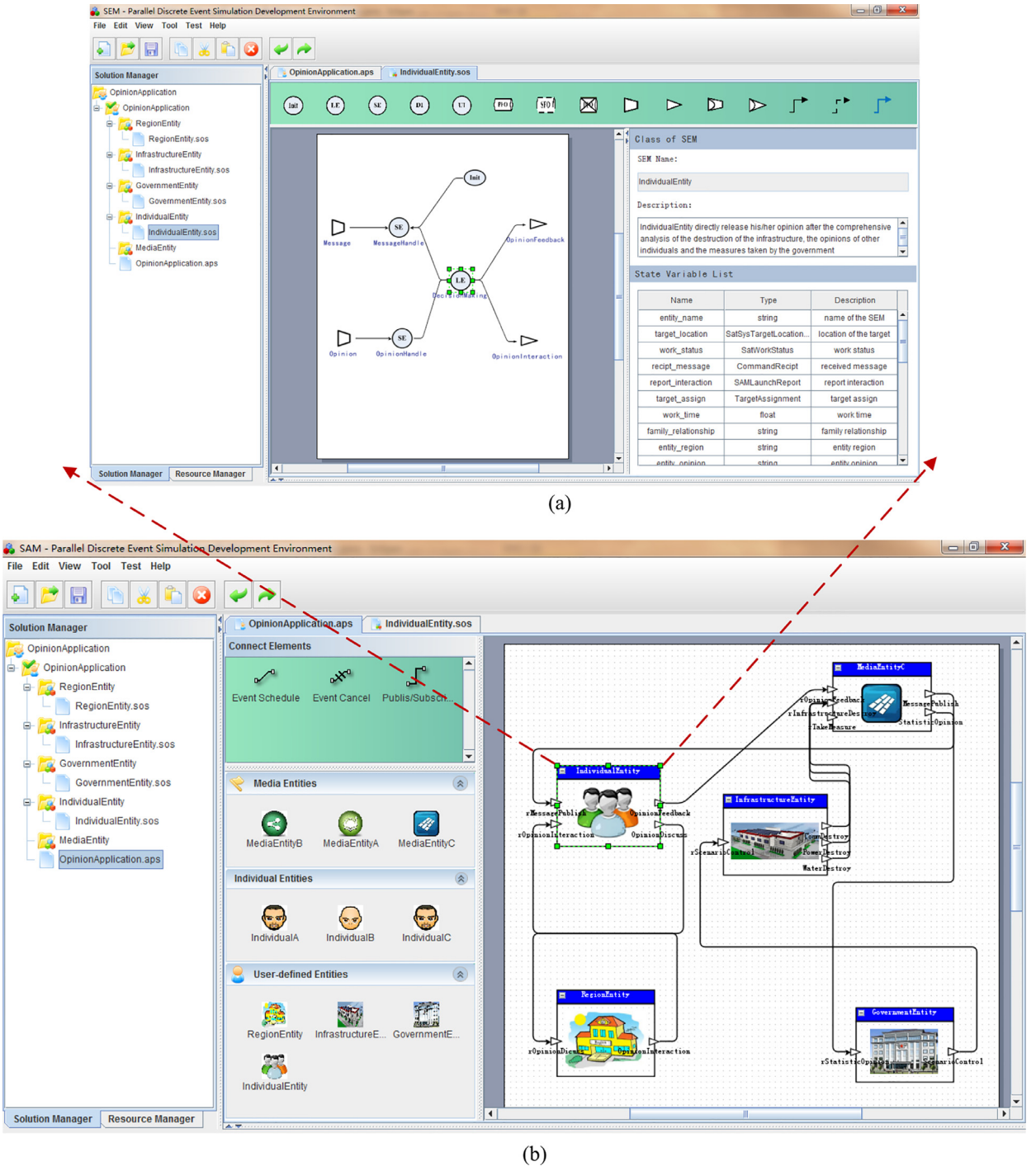

Fig. 8. Hierarchical composite modelling SOSM: (a) composite the three SMCs to construct the LP *Individual*; (b) composite the LPs to build the SOSM.

network model, natural gas network model and so on. The social behaviour models include agent model and decision algorithm (individual behaviour), interpersonal network model and abstract social layer , urban layer model.

The SOSM can be used to describe the emerging behaviour of individuals in a hotspot region, such as political trends, spread of rumours, individual behavioural motivations and decision preferences. This case analyses the impact of certain types of crisis events (such as damage to infrastructure) on specific public opinion trends in hotspots. In the virtual social environment of the hotspot area, the individual model(agent) is constructed, and the behaviour of the group is driven by the constant stimulation of the individuals, as well as the individual evolution and the interaction between the individuals.

Table 3
Simulation scenario configuration.

Experimental parameters	Description/Value
Number of *individuals*	[1000, 32,000]
Number of *cities*	10
Number of *infrastructures*	50
Number of *medias*	50
Number of *environments*	1000
Number of stimuli events	10
Probability of interaction among *strangers*	[5%, 20%]
Number of LP partitions	[1, 6]
Time management algorithm	Conservative
Lookahead	10
Simulation run time	1440

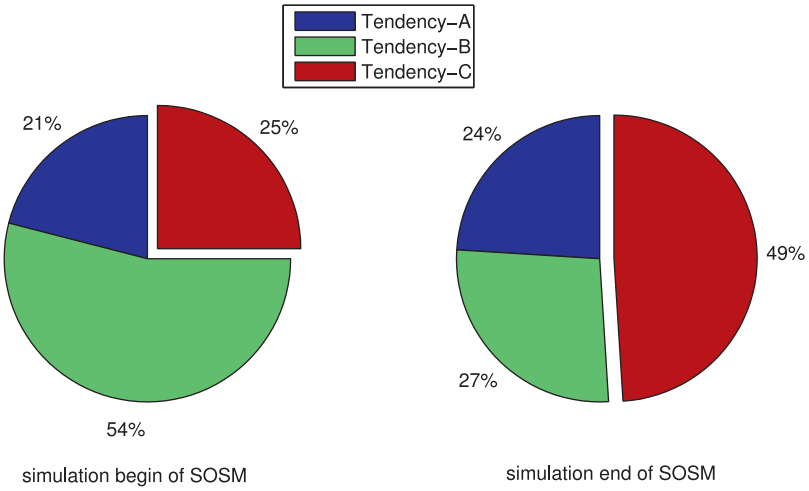

Fig. 9. Percentage of public opinions after simulation execution.

Supposing that there are three kinds of public opinions, *Tendency_A*, *Tendency_B* and *Tendency_C*, *individuals* are most concerned about their own *development*. When a positive event of this type occurs, *individuals* tend to support *Tendency_A* with a certain probability, when such a negative event of this type occurs, *individuals* are more likely to oppose *Tendency_A*. At the same time, *individuals* are also concerned about their own *security*. When a positive event of such type occurs, *individuals* have a certain probability of supporting *Tendency_C*. When such negative events occur, *individuals* are more likely to oppose the *Tendency_C*. Of course, *development* events do not necessarily lead to *Tendency_A*. For example, maybe some positive events of *development* occur, but contrary to a reason(this is often related to the reasonable factors of age, gender, culture degree, occupation, economic status, etc.), so that an *individual* may oppose the *Tendency_A*, and tend to other tendencies. The simulation experiment was designed like that after 100 units of simulation time applied a positive *security* events (i.e. 100, 200, 300, 400, 500, a total of 5 times), while randomly generate the *development* events, the simulation results are shown in Fig. 9. We can see that when *individuals* are stimulated by external *security* events, the overall trend of public opinions will be beneficial to the direction of the stimulus events.

5.2.2. Performance evaluation

First of all, we execute the SOSM in a sequential setup(simulation execution on 1 CPU core). Fig. 10 reports the simulation execution time with different configurations of probability of interaction between *strangers*. The number of *individuals* has been set in the range[1000, 32,000]. For each configuration, the execution time has been adjusted to maintain a fixed density of *individuals*. The figure shows that the interactions between *strangers* strongly affects the scalability of the SOSM.

From Fig. 10, we can see that the sequential setup shows a limited scalability. In this case, we use more than two of the available CPU cores to process the SOSM evolution. The set of LPs was partitioned among the CPU cores and SUPE platform was used to deliver the interactions among LPs. Fig. 11 shows the performance for executing SOSM with different CPU cores. We can see that the execution time slight increases when the number of CPU cores increases from 5 to 6 in the scenarios with 32,000. That is because SUPE platform will generate a LP *manager* during the simulation execution, which is used to create, manage, and delete LPs. The LP *manager* will take up the multi-core computing resources. Resource competition leads to a decline in the efficiency of simulation execution with larger scale. Apart from this factor, we can draw the trend of SOSM can adapt to the growth of the number of CPU cores.

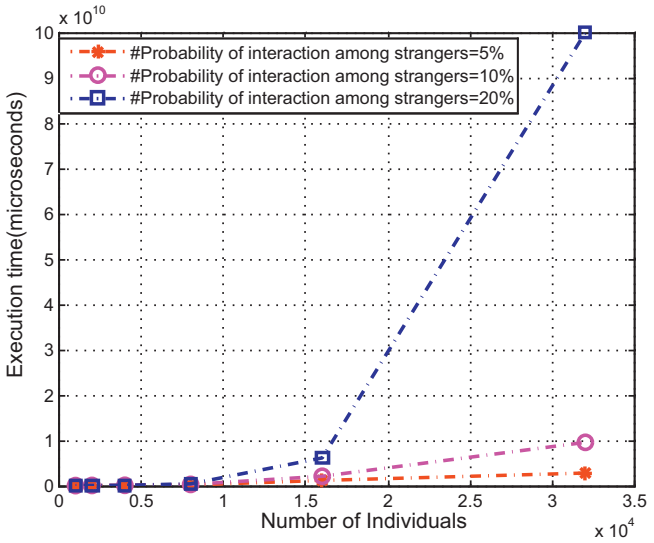

Fig. 10. Scalability evaluation: increasing number of *individuals*, sequential execution, different probabilities of interaction between *strangers*.

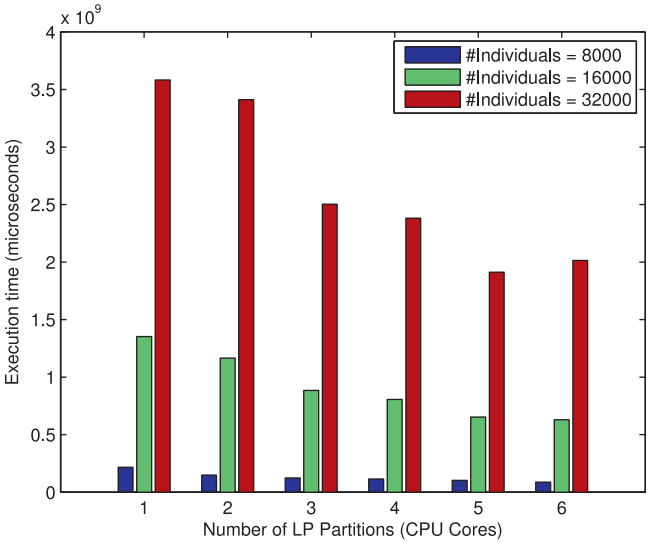

Fig. 11. Scalability evaluation: increasing number of LP partitions, parallel execution, different number of individuals.

Table 4 shows the experimental results that can be obtained by different configurations of *individuals* and LP partitions. The result indicates that SOSM developed based on the hierarchical composite framework can well support higher numbers of *individuals*. As far as we know, every parallel execution is limited by the characteristics of the simulated model. In fact, the SOSM is characterized by a huge amount of interactions among multiple agents. Fig. 12 shows that the number of processed interactive events is dominated by an exponential dependence with the increasing number of *individuals*. With this kind of simulated model, a linear speedup for parallel execution can not be expected.

6. Conclusions and future work

Parallel discrete event simulation provides an important experimental approach to study the sophisticated issues about CAS. To support decomposition and modularity for modelling CAS, we proposed the three-level hierarchical architecture and gave the formal definition of each layer including SMC, LP and CSM. Then The hierarchical composite modelling process and the parallel simulation execution strategy is discussed to support the modelling and simulation of a CAS. A complex social opinion system model named SOSM is proposed based on this hierarchical composite modelling framework. The experimental results confirms the viability of utilizing multi-level architecture for simulating large scale complex adaptive systems.

Table 4
Experimental results of parallel executing SOSM ($\times 10^3$ microseconds).

CPU cores	Execution Time with different number of *individuals*						
	1000	2000	4000	8000	16,000	25,000	32,000
1	18,563	38,492	78,286	215,804	1,352,914	2,171,175	3,582,011
2	11,323	24,175	50,115	147,979	1,165,841	1,512,753	3,411,439
3	7360	15,132	32,623	123,957	884,009	127,935	2,502,244
4	5602	12,436	26,459	114,849	805,306	907,095	2,381,630
5	4729	10,636	22,726	102,093	652,626	**882,890**	**1,911,520**
6	4291	9726	20,484	86,942	629,262	**884,550**	**2,014,301**

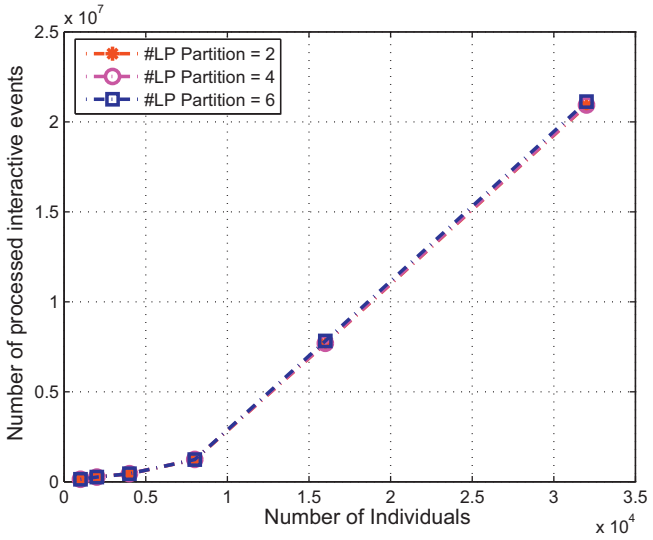

Fig. 12. Interaction measurement: with the increasing of individuals, parallel execution, different number of LP partitions.

As to our future work, we plan to study the validation of a composition of different SMCs. That's because it is often assumed that any SMC that had been placed in the repository is valid in practice, but the validity of SMCs does not imply that a composition of them can be assumed to be valid.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China (No. 71601181).

References

[1] J.S. Lansing, Complex adaptive systems, Annu. Rev. Anthropol. (2003) 183–204.
[2] J.H. Holland, Studying complex adaptive systems, J. Syst. Sci. Complexity 19 (1) (2006) 1–8.
[3] J. Tang, M.A. Piera, T. Guasch, Coloured Petri net-based traffic collision avoidance system encounter model for the analysis of potential induced collisions, Trans. Res. part C: Emerg. Technol. 67 (2016) 357–377.
[4] X. Hu, B.P. Zeigler, S. Mittal, Variable structure in devs component-based modeling and simulation, Simul.: Trans. Soc. Model. Simul. Int. 80 (2) (2005) 91–102.
[5] H.S. Sarjoughian, Model composability, in: Proceedings of the 2006 Winter Simulation Conference, 2006, pp. 149–158.
[6] M.D. Petty, J. Kim, S.E. Barbosa, Software frameworks for model composition, Model. Simul. Eng. 2014 (1) (2014) 1–18.
[7] K.S. Perumalla, S.K. Seal, Discrete event modeling and massively parallel execution of epidemic outbreak phenomena, Simul.: Trans. Soc. Model. Simul. Int. 88 (7) (2012) 768–783.
[8] B. Hou, Y. Yao, B. Wang, Modeling and simulation of large-scale social networks using parallel discrete event simulation, Simul.: Trans. Soc. Model. Simul. Int. 89 (10) (2013) 1173–1183.
[9] J. Liu, J.J. Cochran, L.A. Cox, Parallel discrete-event simulaiton, John Wiley & Sons, Inc., Hoboken, NJ, 2011.
[10] R.M. Fujimoto, Research challenges in parallel and distributed simulation, Acm Trans. Model. Comput.Simul. 26 (4) (2016) 1–29.
[11] P. Taillandier, D. Vo, E. Amouroux, A. Drogoul, Gama: A simulation platform that integrates geographical information data, agent-based modeling and multi-scale control, in: Principles and Practice of Multi-Agent Systems: International Conference on Principles and Practice of Multi-Agent Systems, Springer Berlin Heidelberg, 2012, pp. 242–258.
[12] N. Collier, M. North, Repasthpc: a platform for large-scale agent-based modeling, Large-Scale Computing Techniques for Complex System Simulations (2011) 81–110.
[13] S. Rodriguez, N. Gaud, S. Galland, Sarl: a general-purpose agent-oriented programming language, in: International Conference on Intelligent Agent Technology, Warsaw, Poland, 2014, pp. 156–163.
[14] S. Galland, N. Gaud, S. Rodriguez, Janus: Another yet general-purpose multiagent platform, Seventh AOSE Technical Forum, Paris, 2010.
[15] G. D'Angelo, S. Ferretti, V. Vittorio, Multi-level simulation of internet of things on smart territories, Simul. Modell. Pract. Theory 73 (2017) 3–21.

[16] S. Rodriguez, V. Hilaire, N. Gaud, S. Galland, A. Koukan, Self-organizing Software: From Natural to Artificial Adaptation, Springer Berlin Heidelberg, pp. 238–263.
[17] S. Galland, F. Grimaud, P. Beaune, J.P. Campagne, Supply Chain Optimisation: Product/Process Design, Facility Location and Flow Control, Springer Berlin Heidelberg, pp. 277–288.
[18] B.P. Zeigler, H. Praehofer, T.G. Kim, Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems, Academic Press, Hoboken, NJ, 2000.
[19] S. Palaniappan, H.S. Sarjoughian, Application of the devs framework in construction simulation, in: Proceedings of the 2006 Winter Simulation Conference, 2006, pp. 2077–2086.
[20] J. Mather, The DEVSJAVA Simulation Viewer: A Modeler GUI that Visualizes the Structure and Behavior of Hierarchical DEVS Models, The University of Arizona, Arizona, USA, 2003.
[21] G.A. Wainer, Modeling and simulation of complex systems with cell-devs, in: Proceedings of the 2004 Winter Simulation Conference, 2004, pp. 49–60.
[22] B. Cobanoglu, A. Zengin, F. Celik, Implementation of devs based distributed network simulator for large-scale networks, Int. J. Simul. Modell. 13 (2) (2014) 147–158.
[23] L. Schruben, Simulation modeling with event graphs, Commun. ACM 26 (11) (1983) 957–963.
[24] A. Buss, P.J. Sanchez, Modeling very large scale systems: building complex models with legos, in: Proceedings of the 2002 Winter Simulation Conference, 2002, pp. 732–737.
[25] A. Buss, Composability and component-based dicrete event simulation, in: Proceedings of the 2007 Winter Simulation Conference, 2007, pp. 694–702.
[26] J. Lara, Distributed event graphs:formalizing component-based modelling and simulation, Electron. Notes Theor. Comput. Sci. 127 (2005) (2005) 145–162.
[27] B. Wang, B. Deng, F. Xing, Partitioned event graph: formalizing lp-based modelling of parallel discrete-event simulation, Math. Comput. Model. Dyn. Syst. 21 (2) (2014) 153–179.
[28] V. Kesaraju, F. Ciarallo, Integrated simulation combining process-driven and event-driven models, J. Simul. 2012 (6) (2012) 9–20.
[29] S.S. Rizvi, A logical process simulation model for conservative distributed simulation systems, Int. J. Simul. Modell. 12 (2) (2013) 69–81.
[30] K.L. Morse, M.D. Petty, P.F. Reynolds, Findings and recommendations from the 2003 composable mission space environments workshop, in: Proceedings of the Simulation Interoperability Workshop, Arlington, USA, 2004, pp. 313–323.
[31] M.D. Petty, Verification, validation, and accreditation, in: Modeling and Simulation Fundamental: Theoretical Underpinnings and Practical Domains, Hoboken, USA, 2010, p. 325C372.
[32] R.M. Fujimoto, Parallel and Distributed Simulation Systems, John Wiley & Sons, Inc., Hoboken, NJ, 2000.
[33] D.M. Nicol, The cost of conservative synchronization in parallel discrete event simulations, J. ACM 40 (2) (1990) 304–333.
[34] D.M. Nicol, J. Liu, Composite synchronization in parallel discrete-event simulation, IEEE Trans. Parallel Distrib. Syst. 13 (5) (2002) 433–446.
[35] W. Buckley, Society as a complex adaptive system, Emergence Complexity Organiz. 10 (3) (1998).
[36] J.H. Millerand, S.E. Page, Complex Adaptive Systems: An Introduction to Computational Models of Social Life, Princeton University Press, 2008.