A Self-Organizing Model for Decentralized Virtual Environments in Agent-Based Simulation Systems

(Extended Abstract)

Mohammad Al-Zinati, Rym Zalila Wenkstern
University of Texas at Dallas
800 West Campbell Road
Richardson, Texas, USA
{mohammad.al-zinati, rymw}@utdallas.edu

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Multiagent systems; I.6.8 [**Simulation and Modeling**]: Types of Simulation—*self-organizing*; D.2.11 [**Software Engineering**]: Software Architectures

1. INTRODUCTION

For simulations to be meaningful, it is necessary to implement realistic models for both virtual agents and environment. A lot of attention has been given to the definition of accurate models for agents. Unfortunately not much has been done for the definition of virtual environments that mimic the complexity of real-world environments. The reason is twofold: 1) The construction of realistic virtual environments (also called *open* environments) is not a trivial task [3]. Such environments are *inaccessible*, *non-deterministic*, *dynamic* and *continuous*. 2) Realistic simulations involve the execution of a *large-number* of *sensor-based perception* agents in an *open* environment. Unfortunately, limited computational resources make this goal untenable on a single machine.

A few MABS have proposed models for open virtual environments. Most of these models represent the environment as a single massive component that is managed by one control unit. Other models decompose the environment into regions that are also managed by a single control unit [4]. In both cases, centralized control creates a bottleneck and limits the scalability of the simulation. On the other hand, a very limited number of MABS have proposed a partitioned structure of the environment with control units managing specific spatial areas [2]. Unfortunately, these systems do not leverage several of the benefits enabled by decentralized control.

In this paper we propose a model for the execution of large-scale MABS with *open* environments on a single host. In our approach, agents execute their behaviors and are not subjected to any resource management constraints (e.g., aggregation). The open environment has a decentralized structure that is supported by an underlying self-organizing system. During the execution of the simulation, virtual agents are unaware of the partitioned structure of their environment and the self-organization activities occurring at the supporting system layer.

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

2. A SELF-ORGANIZING MODEL FOR VIRTUAL ENVIRONMENTS

The model presented in this section is based on the reference architecture for agent-based autonomic software systems defined by the authors. In the proposed model, a virtual environment is decomposed into partitions called *cells*. As shown in Figure 1, the environment is supported by an underlying self-organizing system that consists of micro- and macro-level entities. At the micro level, specialized agents called *cell controllers* manage cells. A cell controller assumes two critical roles: *cell function management* and *cell performance management*. Cell function management is related to the cell controller's responsibility to: a) autonomously manage environmental information about its cell; b) be aware of the virtual agents located in its defined area; c) inform its neighboring cell controllers of propagating events; and d) provide its local virtual agents with an accurate perception of their surroundings. Cell performance management is related to the cell controller's responsibility for keeping its workload under nominal capacity and meeting performance requirements (e.g., CPU usage). If the performance constraints are not met, the controller has the ability to reorganize its cell by performing one of two actions: splitting its cell and spawning a new controller to manage the new cell; or merging its cell with another cell and releasing its resource[1]. Each controller is required to complete its cell function management at every simulation cycle. Since controllers execute their work concurrently, the cycle time depends on the parallel completion of each controller's work. When the functional work among controllers becomes unbalanced, two problems arise: 1) the simulation cycle time is delayed and 2) computational resources are wasted.

At the macro-level, specialized agents called *coordinators* monitor and guide a collection of cell controllers. A coordinator's main responsibility is to ensure that the simulation performance requirements (i.e., duration of the simulation cycle, resource utilization) for the set of controllers it supervises are met. The definition of coordinators is necessary

[1]In this paper we use the re-organization decisions of splitting and merging cells but these can be replaced with other application-specific reorganization decision.

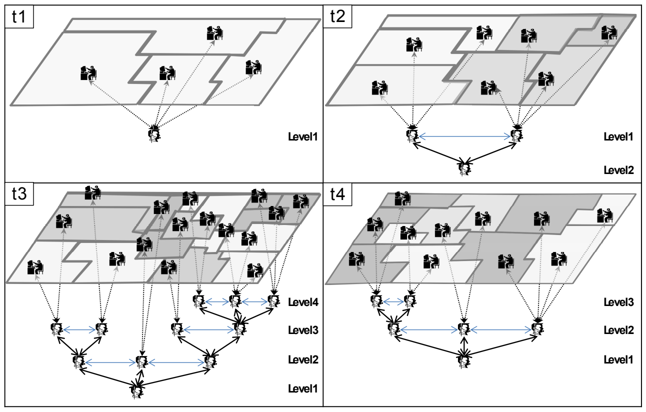

Figure 1: Self-organizing environment

since purely decentralized reorganization can lead to undesired behavior and performance. Nevertheless, it is important to note that coordinators are centers of *knowledge* and *advice* rather than centers of control. From a behavioral perspective, since a coordinator can oversee a limited number of cell controllers, when its load increases or decreases, it either spawns a new coordinator and passes part of its load on to it, or merges its load with another coordinator and destroys itself.

In the self-organizing system, interactions happen in two ways: horizontal and vertical. Horizontal interactions occur between agents of the same type and at the same level. Vertical interactions occur between agents of the same or different type located at different levels; they are either bottom-up or top-down. Note that, since our hierarchy is reversed, the root of the hierarchy is shown at the bottom while the leaves are at the top (see Figure 1).

Controller Interactions. A *horizontal controller-to-controller* interaction occurs when a controller offers a to merge with its neighboring controller to save resources. Depending on the neighboring controller load, this may result in merging the two cells and releasing a controller. A *vertical bottom-up controller-to-coordinator* interaction occurs when a controller is overloaded and requires information about available resources in order to perform a split. A *vertical top-down coordinator-to-controller interaction* occurs when a coordinator wants to advise one of its subordinate controllers to perform a split task. Splitting involves spawning a new controller and reassigning a portion of a cell to the new cell controller.

Coordinator Interactions. *Horizontal coordinator-to-coordinator* interactions occur when a coordinator is in need of additional resources or when it is ready to merge with its neighboring coordinator. A *vertical bottom up coordinator-to-coordinator* interaction occurs when a coordinator wants to inform its parent coordinator about its inability to get assistance from coordinators at the same level. A *vertical top-down coordinator-to-coordinator* interaction occurs when a parent coordinator provides assistance to a child coordinator.

Proactive behavior interactions A controller's proactive behavior may trigger interactions with controllers (e.g., to advise them to merge in order to satisfy broader system goals) or with subordinate coordinators (e.g., to advise them to redistribute their available resources in order to achieve a better resource distribution).

3. EXPERIMENTAL RESULTS

The proposed self-organizing model has been fully implemented and tested using DIVAs, a JAVA-based framework for the development of large-scale agent-based simulation systems [1]. The simulation scenarios were executed on a multicore PC (Intel Core i7 X980 CPU (3.33GHz), 6.00 GB, 64-bit Windows 7). Controllers run on a thread execution pool and coordinators were implemented as daemon threads.

All experiments take place in a virtual city environment consisting of 814 environment objects (e.g., commercial buildings, houses, traffic signals). Situated agents representing humans perceive their surroundings through advanced vision, auditory and olfactory sensors. They execute complex path-finding and collision avoidance algorithms to move within the environment. In addition, agents interact with other agents, plan and deliberate to achieve their goals (e.g., move to location).

The experimental results show the superiority of the proposed self-organizing architecture over non self-organizing decentralized architectures in MABS. The self-organizing virtual environment is scalable (600 agents for a single-cell architecture versus 4000 agents for the self-organizing architecture), performs better (the CPU utilization is less intensive), and the emergence of undesired behavior is controlled.

4. CONCLUSION

In this paper we presented a self-organizing model for decentralized virtual environments in MABS. A virtual environment structure is supported by an underlying software system consisting of specialized agents that re-organize themselves and the environment structure to ensure that the simulation functional and performance requirements are met.

In the current model implementation, cells are split evenly even if agents and objects are not uniformly distributed in the cell area. It may be interesting to implement an algorithm that performs the split based on the agent/object distribution.

5. REFERENCES

[1] M. Al-Zinati, F. Araujo, D. Kuiper, J. Valente, and R. Z. Wenkstern. DIVAs 4.0: A Multi-Agent Based Simulation Framework. In *Proceedings of the 17th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2013)*, pages 105–114, Delft, Netherlands, November 2013.

[2] N. Pelechano, J. Allbeck, and N. Badler. Controlling individual agents in high-density crowd simulation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer animation*, pages 99–108, San Diego, California, USA, August 2007.

[3] S. Russell and P. Norvig. *Artificial Intelligence A modern approach.* Prentice-Hall, Egnlewood Cliffs, Third edition, 2010.

[4] M. Xiaofeng, W. Chaozhong, and Y. Xinping. A multi-agent model for evacuation system under large-scale events. In *Proceedings of the IEEE International Symposium on Computational Intelligence and Design*, pages 557–560, Wuhan, China, October 2008.