Evolutionary Design of Agent-based Simulation Experiments

(Demonstration)

James Decraene, Yew Ti Lee, Fanchao Zeng
Mahinthan Chandramohan, Yong Yong Cheng, Malcolm Yoke Hean Low

Parallel and Distributed Computing Center
School of Computer Engineering
Nanyang Technological University, Singapore
jdecraene@ntu.edu.sg

ABSTRACT

We present CASE (complex adaptive systems evolver), a framework devised to conduct the design of agent-based simulation experiments using evolutionary computation techniques. This framework enables one to optimize complex agent-based systems, to exhibit pre-specified behavior of interest, through the use of multi-objective evolutionary algorithms and cloud computing facilities.

Categories and Subject Descriptors

I.6.5 [**Computing Methodologies**]: Simulation and modeling—*Model Development*; I.2.8 [**Computing Methodologies**]: Artificial intelligence—*Problem Solving, Control Methods, and Search*

General Terms

Performance, Experimentation

Keywords

Design of experiments, agent-based simulation, evolutionary computation

1. INTRODUCTION

Agent-based simulations (ABSs) are increasingly being employed to examine various complex adaptive systems [5]. Nevertheless, the study of such systems using ABSs is a complicated and time-consuming task which is often conducted in an iterative manner. During each iteration, the modeling, design of experiments, execution and analysis of simulations are conducted to *progressively* gain insights in the key factors leading to the emergence of target phenomena.

To facilitate the study of complex agent-based systems, we propose a modular evolutionary framework, coined CASE for "complex adaptive system evolver", to perform the design of experiments using evolutionary computation techniques (a similar approach was recently utilized for materials science and catalysis experiments [2]). Indeed, conventional design of experiments techniques cannot efficiently tackle complex experimental spaces.

We employ Pareto-based multi-objective evolutionary algorithms to automate the modeling and analysis of agent-based simulation models. Moreover, cloud computing is also utilized to assist with the scalability and reliability issues. The latter are commonly met when conducting large-scale experiments using distributed computing facilities.

2. THE CASE FRAMEWORK

An overview of the CASE framework is provided. CASE was implemented in a modular manner (using the Ruby programming language) to accommodate with relative ease the user's specific requirements (e.g. use of different simulation engines or evolutionary algorithms, etc.). CASE is composed of three main components which are distinguished as follows:

1. *The model generator*: This component takes as inputs a base simulation model specified in the eXtended Markup Language and a set of model specification text files. According to these inputs, novel XML simulation models are generated and sent to the simulation engine for execution/evaluation (CASE only supports simulation models specified in XML).

2. *The simulation engine*: The set of XML simulation models is received and executed by the stochastic simulation engine. Each simulation model is replicated a number of times to account for statistical fluctuations (30 repetitions are typically conducted). A set of result files detailing the outcomes of the simulations (in the form of numerical values for instance) are generated. These measurements are used to evaluate the generated models, i.e., these figures are the fitness (or "cost") values utilized by the evolutionary algorithm (EA) to direct the search.

3. *Evolutionary algorithm*: The set of simulation results and associated model specification files are received by the evolutionary algorithm, which in turns, processes the results and produce a new "generation" of model specification files. The generation of these new model

Cite as: Evolutionary Design of Agent-based Simulation Experiments (Demonstration), Decraene J., Lee Y.T., Zeng F., Chandromohan M., Yong Y.C., Low M.Y.H., *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 1321-1322.

specifications is driven by the user-specified search objectives (e.g. maximize/minimize some quantitative values capturing the target system behavior). The algorithm iteratively generates models which would progressively, through the evolutionary search, best exhibit the desired outcome behavior. The model specification files are sent back to the model generator; this completes the search iteration.

The list of evolvable simulation model properties are specified given their XPath, name and numerical values ranges (min,max). In addition to (real) numerical values, it is possible to evolve model property values in the form of enumerable sets (e.g. low, medium, high, etc.) to address model properties that cannot be expressed as numerical values. Finally, it is also possible to evolve the structure of the simulation model (e.g. adding/removing dynamically new agents) [3].

Moreover, the evolutionary search can be conducted under constraints: This optional feature may be utilized to introduce specific considerations when evolving particular model properties. For instance, the user may devise interactions between properties to occur according to some pre-defined conditions. These constraints aim at increasing the plausibility of generated simulation models (e.g. through introducing cost trade-off for specific property values). The specification of such constraints is carried out through the use of a rule-based approach. Finally, constraints can also be introduced through devising additional search objectives (e.g. minimize the value of some evolvable property value).

Communications between the three components are conducted *via* text files for simplicity and flexibility (for instance, this enables the use of PISA evolutionary algorithm modules [1]). Note that the flexible nature of CASE allows one to develop and integrate different simulation engines (using models specified in XML), and evolutionary algorithms.

The experimental settings include: the selected simulation engine, the selected evolutionary algorithm and associated setting (e.g. population size, number of search iterations, mutation probability, set of objectives, etc.), the number of simulation replications, the number of CASE run replications (similarly to ABSs, evolutionary algorithms are stochastic processes, replications of the experimental runs may also be necessary).

3. CLOUD COMPUTING

Cloud computing [4] is a high performance computing (HPC) paradigm which has recently attracted considerable attention. The computing capabilities (i.e., compute and storage clouds) are typically provided as a service *via* Internet. This web approach enables users to access HPC services without requiring expertise in the technology that supports them. The key benefits of cloud computing are reliability (failed operations may automatically be rescheduled), reduced cost (cloud computing infrastructures are provided/managed by a third-party) and scalability (multiple clouds can be aggregated).

The implementation [4] was conducted using the MapReduce programming model:

- *Map*: During the Map phase, the set of simulation models (to be executed) is partitioned into subsets and distributed across multiple compute nodes. The subsets are processed in parallel by the different nodes. The set of intermediate files results resulting from the Map phase are collected and processed during the Reduce phase.
- *Reduce*: Multiple compute nodes process (i.e. evolutionary selection of the most satisfactory/promising candidate models) the intermediate files which are then collated to produce the result data.

CASE may currently submit experiments to the cloud computing facilities hosted at the Parallel and Distributed Computing Center, Nanyang Technological University and Amazon EC2.

4. DEMONSTRATION

The demonstration includes a case study, from the military operations research field [3], examining the protection of a maritime anchorage area against piracy threats. A brief presentation of the employed simulation engine is first performed. Following on from this, the CASE framework is presented in detail. An example experiment is then conducted illustrating the typical usage of CASE.

5. ON GOING-WORK

On-going work focuses on developing further evolutionary optimization techniques such as: multi-objective co-evolution (given two-sided competitive wargame scenarios), niching (to diversify the solution models in the decision space) and the evolution of *nested* simulation structure (to dynamically add/remove agents *and* internal components, e.g. course of actions waypoints).

Acknowledgments

We would like to thank the Defence Research and Technology Office, Ministry of Defence, Singapore, for sponsoring the *Evolutionary Computing Based Methodologies for Modeling, Simulation and Analysis* project (DIRP FY08).

6. REFERENCES

[1] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler. PISA – A Platform and Programming Language Independent Interface for Search Algorithms. In *Proceeding of the Second Evolutionary Multi-Criterion Optimization*, LNCS, pages 494–508. Springer, 2003.

[2] J. Cawse, G. Gazzola, and N. Packard. Efficient discovery and optimization of complex high-throughput experiments. *Catalysis Today*, 159(1):55–63, 2010.

[3] J. Decraene, M. Chandramohan, M. Low, and C. Choo. Evolvable Simulations Applied to Automated Red Teaming: A Preliminary Study. In *Proceedings of the 42th Winter Simulation Conference*, pages 1444–1455, 2010.

[4] J. Decraene, Y. Yong, M. Low, S. Zhou, W. Cai, and C. Choo. Evolving Agent-based Simulations in the Clouds. In *Third International Workshop on Advanced Computational Intelligence*, pages 244–249, 2010.

[5] J. Holland. Studying Complex Adaptive Systems. *Journal of Systems Science and Complexity*, 19(1):1–8, 2006.