Simulation Modelling Practice and Theory 000 (2017) 1–18

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

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

Flexible asynchronous simulation of iterated prisoner's dilemma based on actor model

Grażyna Skiba[a], Mateusz Starzec[a], Aleksander Byrski[a,*], Katarzyna Rycerz[a], Marek Kisiel-Dorohinicki[a], Wojciech Turek[a], Daniel Krzywicki[a], Tom Lenaerts[b,c], Juan C. Burguillo[d]

[a] AGH University of Science and Technology, Faculty of Computer Science, Electronics and Telecommunications, Department of Computer Science, Krakow, Poland
[b] Université Libre de Bruxelles, Machine Learning Group, Boulevard du Triomphe CP212, Bruxelles 1050, Belgium
[c] Vrije Universiteit Brussel, Artificial Intelligence Lab, Pleinlaan 2, Bruxelles 1050, Belgium
[d] University of Vigo, Escuela de Ingeniería de Telecomunicación, Campus Universitario Lagoas-Marcosende, Vigo 36310, Spain

A R T I C L E   I N F O

Article history:
Available online xxx

Keywords:
Concurrent and distributed simulations
Iterated prisoner's dilemma
Scala
Actor model

A B S T R A C T

The wide range of applications of the Iterated prisoner's dilemma (IPD) game made it a popular subject of study for the research community. As a consequence, numerous experiments have been conducted by researchers along the last decades. However, topics related with scaling simulation leveraging existing HPC infrastructure in the field of IPD did not always play a relevant role in such experimental work. The main contribution of this paper is a new simulation framework, based on asynchronous communication and its implementation oriented to distributed environments. Such framework is based on the modern Akka actor platform, that supports concurrent, distributed and resilient message-driven simulations; which are exemplified over the IPD game as a case study. We also present several interesting results regarding the introduction of asynchrony into the IPD simulation in order to obtain an efficient framework, so the whole simulation becomes scalable when using HPC facilities. The influence of asynchrony on the algorithm itself is also discussed, and the results show that it does not hamper the simulation.

1. Introduction

The history of the prisoner's dilemma (PD) game traces back as far as the 1950s. The two-person PD game models a situation where two individuals must play and decide if they will cooperate or defect to their opponent. Both players can make profit by betraying a cooperative opponent, but they will gain less if both betray each other. Such model abstracts diverse real-world situations in society or Nature, and makes it an interesting research topic, in social and biological sciences, e.g., politics (arms race), economics (market competitions), sports (use of banned substances) and others [1].

* Corresponding author.
E-mail addresses: grazyna.skiba@iisg.agh.edu.pl (G. Skiba), mateusz.starzec@iisg.agh.edu.pl (M. Starzec), olekb@agh.edu.pl (A. Byrski), kzajac@agh.edu.pl (K. Rycerz), doroh@agh.edu.pl (M. Kisiel-Dorohinicki), wojciech.turek@agh.edu.pl (W. Turek), daniel.krzywicki@agh.edu.pl (D. Krzywicki), tlenaert@ulb.ac.be, tlenaert@vub.be (T. Lenaerts), J.C.Burguillo@uvigo.es (J.C. Burguillo).

https://doi.org/10.1016/j.simpat.2017.12.010
1569-190X/© 2017 Elsevier B.V. All rights reserved.

Table 1
Prisoner's dilemma payoffs.

	Cooperate B	Defect B
Cooperate A	A: Reward; B: Reward	A: Sucker; B: Temptation
Defect A	A: Temptation; B: Sucker	A: Punishment; B: Punishment

In real-life, the player interactions modeled by the prisoner's dilemma often happen several times, and this can be simulated by an extension of the classic game, denoted as the iterated prisoner's dilemma (IPD) game. In this version the players play the game multiple times, which gives them the opportunity to penalize the opponent for previous defections [2].

The wide range of applications of the prisoner's dilemma, and its iterated version, has given it a great relevance, and made it a popular subject of study in the Academia. As a result, numerous game variations and experiments have been published in the area. The implementations are often built on a similar bases – a model describing a population of prisoners playing one-on-one consecutive games (see [3–7]).

There are many simulation frameworks supporting simulations like IPD, especially those agent-based [8], but only some of them are actually suited for large scale execution in parallel and distributed infrastructures. Good examples are REPAST HPC [9], FLAME [10], PDES-MAS [11] or Pandora framework [12]. All the mentioned platforms use the standard message passing (MPI) protocol to create processes for parallel and distributed execution, as well as for communication and synchronization mechanisms – a well-established standard used in simulation for many years. The standard approaches employing MPI are implemented in C++, which can offer good performance but is inefficient in terms of rapid experiments development and rather inflexible as a general implementation technique. This situation creates an opportunity to develop novel, dedicated solutions leveraging contemporary, high-level languages and technologies that would ease the development, debugging, deploying and modification processes.

Note that exploring the space of the IPD problem (or doing similar simulations) requires immense computational efforts, when one goes beyond simple modeling and tries to conduct practical experiments. Thus, empirical game-testing analysis (as defined by Wellman [13]) makes researchers either to modify the problem itself (c.f., deviation-preserving reduction [14]) in order to get feasible results in limited time; or to focus on implementing efficient simulation frameworks (parallel/distributed ones) leveraging contemporary high-performance computing infrastructure. The step towards defining such HPC-grade approach and dedicated simulation infrastructure is presented in this work.

In order to develop parallel and distributed simulations, and to enable the modeling of large populations, we need a scalable approach; and we have decided to use the actor model [15,16] as one supporting high-level abstraction of actors. Actors are defined as concurrent processes that use asynchronous messages as the communication mechanism among them. This model has been proven to be efficient and scalable, including multi-core and multi-node scenarios [17]. It has also been used in numerous applications (e.g., actor-based work-flows [18] and in simulating social network dynamics [19]).

The main contribution of this paper is a simulation approach based on asynchronous communication and its implementation dedicated to the distributed environment based on the modern Akka actor platform [20] that supports concurrent, distributed, and resilient message-driven simulations, exemplified over the IPD game. It allows rapid development of various variants of the simulations and their execution on parallel hardware, including supercomputers. Akka provides modern fault tolerance mechanisms in distributed environments in particular persistence (ability to save internal state of actors for recovery) and supervision (the actor that supervises responds to the failures of its subordinates). Akka also supports building scalable simulations by providing various message routing strategies, and it is worth noting that the platform presented here stems from the one developed by Krzywicki et al. [21].

We also present a set of experiments regarding the introduction of asynchrony into the IPD simulation (see, [22,23]). The study of synchronous versus asynchronous updating in the IPD has already been done in the literature (see, e.g. [24]) and it is generally accepted that asynchrony makes games much more realistic. However, in this work we focus on asynchrony introduced to obtain an efficient framework, so the whole simulation becomes scalable when using HPC facilities. Its influence on the algorithm itself is discussed, but it should be the subject of further research.

The rest of the paper is organized as follows. Section. 2 presents the basic concepts in IPD simulations along with a review of the most important issues connected with framework implementations in the scientific literature. Section. 3 discusses the proposed framework for simulating the IPD game using the actor model. Section. 4 describes the configuration of the experiments and the environment details. Section. 5 presents and discuss a significant number of experimental results, and finally; Section. 6 concludes the paper and draws some potential future work.

2. From classic to efficient IPD simulation

The prisoner's dilemma (PD) is a two-person nonzero-sum game, where each player individually selects one of two possible actions - cooperation or defection. Based on their actions, both players get the payoffs presented in Table 1.