Acta Scientific COMPUTER SCIENCES

Volume 4 Issue 7 July 2022

Research Article

DOSSIER: A Toolkit to Extract Data from Digital Life Simulations Using Dose

Argho Maitra[1] and Maurice HT Ling[1,2]*

[1]*School of Data Sciences, Perdana University, Malaysia*

[2]*HOHY Private Limited, Singapore*

***Corresponding Author:** Maurice HT Ling, School of Data Sciences, Perdana University, Malaysia.

Received: May 14, 2022

Published: June 09 , 2022

Abstract

Artificial life, also known as digital organisms, has been useful in testing various evolutionary hypotheses. DOSE is one of the platforms for experimenting with digital organisms and had been used in several studies. However, the internal architecture of DOSE does not allow for easy processing of simulation results despite storing the state of each digital organism and the world for each generation in an SQLite database. To address this problem, we implemented DOSSIER, a Python-based toolkit that can connect with the SQLite database, facilitate data extraction and processing into a standardized table of fitness scores.

Keywords: Artificial Life; Digital Organisms; Cellular Automata; Computational Tool; Python Programming; DOSE

Introduction

Artificial life provides an alternative to conventional biological experimentation. Earlier studies proposed that cellular automata (abstract computer spaces) can house virtual automata; that is, artificial molecules that can code for the molecular logic of life [1]. Such computer-based model organisms are also known as digital organisms (DOs) and have made artificial life a distinct possibility [2]. DOs behave like viruses and bacteria: they can self-replicate, compete, mutate and evolve in controlled environments [3], and can exhibit evolutionary dynamics similar to microbes in chemostats [4]. They have properties that can address the limitations posed by experiments using biological organisms; such as, faster generation rates [5], precise mutation rates and fitness scores [6], high statistical accuracy [7], and reproducible experimentation with a perfect fossil record [8]. They are referred to as model organisms [9] and serve as instances of evolution [10]; for this reason, they have been used to test several evolutionary hypothesis in silico [2,7-9,11]. Some prominent examples of DO platforms are Tierra [12], Avida [13] and Aevol [14].

Digital Organism Simulation Environment (DOSE; https://github.com/mauriceling/dose) is one such ecologically-based DO platform that has been used to test several evolutionary hypotheses *in silico* [11,15]. DOSE presents several advantages over other DO platforms like (a) a biochemically inspired metabolism instead of a Boolean one [16], (b) a set of functions to operate at the organism, population and world level [11], (c) a Turing complete instruction set (the Ragaraja genomic interpreter) to mimic the central dogma [17] and (d) an implementation of custom-designed genomic interpreters to suit hypotheses [18]. The DOSE boilerplate acts as a simulation driver to perform the following: construct a 3D world wherein DOs are mapped and initialized, conduct simulations across generations and produce results using a population_report function [11]. The DOSE output is in the form of a collection of time-stamped series data wherein every single series contains information that corresponds to a particular generation [11]. These outputs contain the status of DOs through generations in each simulation and are stored in a text file and an SQLite database.

However, the strengths of DOSE in examining eco-evolutionary hypotheses does not come without limitations. With the current implementation of DOSE, the user must decide on the implementation of the population_report function to get the desired output. Hence, if the user realizes in hindsight that the desired output has not been generated, there is no possibility to go back and retrieve the results during the simulation as the population_report function cannot be modified during the simulation. The alternative is to modify the population_report function and re-perform the simulations. This method is not only counterproductive and leads to a waste of resource; such as, time and computing power but

also creates a barrier between conducting multiple simulations and performing instant analyses. Since there are no convenient methods for the user to analyze data from the DOSE simulation results recorded in the SQLite database, we present an integrated Python toolkit, DOSSIER, to perform data extraction and further facilitate data analysis.

Description of DOSSIER

Assuming that one or more DOSE simulations had been executed and completed, DOSSIER provides a set of functions to read the SQLite database to generate a standardized table of fitness scores using GenerateFitness function (Figure 1). The standardized table of fitness scores is a Pandas data frame object where each row contains fitness scores for each replicated generation. Hence, if there are 100 DOs per replicate per generation, the standardized table of fitness scores will have 102 columns – first column is the replicate number, second column is the generation count, and third to 102nd column will contain fitness score for each of the 100 DOs.

However, GenerateFitness function is a driver function which requires (a) handle for the DOSE database (the SQLite database), (b) identifier for a specific simulation in the DOSE database, (c) a fitness calculation function to generate the fitness score and its parameters. The handle for the DOSE database is provided by ConnectDB function, which takes either the absolute or relative path to the SQLite database file. The simulations recorded within the database can be listed using Sims function (Table 1), which results in a Pandas data frame of simulation start times (start_time) and simulation names (simulation_name). The simulation start time (start_time) is then used to identify the simulation to be processed. More than one simulation can be processed at the same time by providing a dictionary of simulation start times to replicates mapping, {<start_time>: <replicate>}, as simSet parameter to GenerateFitness function. Finally, the fitness calculation function is a user-defined function that takes a Pandas data frame of simulation results and replicate number, calculates fitness for each DO, and returns a standardized table of fitness scores described above. Other supportive functions are listed in Tables 1 and 2. Several examples on the usage of DOSSIER can be found within dossier_examples folder in DOSE.

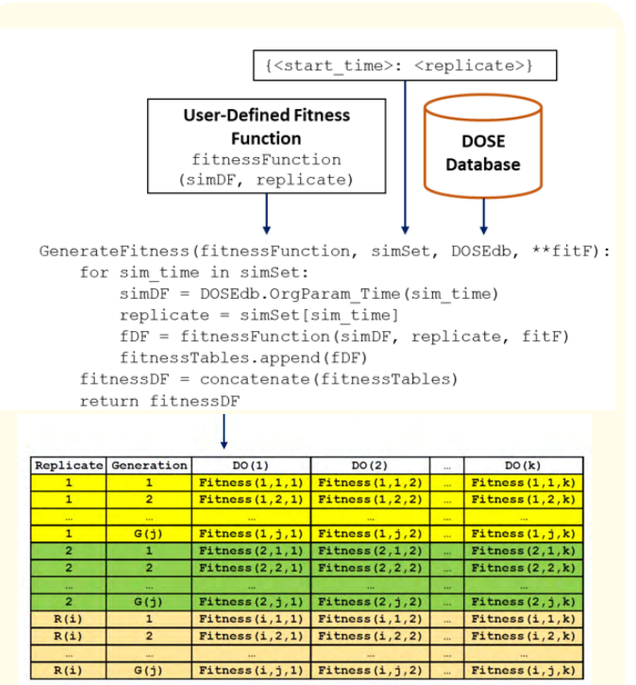

Figure 1: Data Flow from DOSE Database to Pandas Data Frame of Fitness Scores.

Function Name	Description	Returned Data	Comments
Sims()	List available simulation results	Pandas data frame of start_time (simulation start time) simulation_name (simulation name)	No comments.
SimParam_Time (start_time)	List parameters of a given simulation	Pandas data frame of start_time (simulation start time) x (x-axis of ecological cell) y (y-axis of ecological cell) z (z-axis of ecological cell) generation (generation count) key (parameter name) value (parameter value)	start_time will be constant. x, y, z, generation will be None.
SimParam_Name (parameter)	List the values of parameter across simulations		Key will be constant. x, y, z, generation will be None.
WorldParam_ Time (start_time)	List world parameters of a given simulation		start_time will be constant.
WorldParam_ TimeCell (start_time, x, y, z)	List the parameters of a specific ecological cell of a given simulation		start_time, x, y, z will be constant.
WorldParam_ TimeCellName (start_time, x, y, z, parameter)	List the values of a specific parameter of a specific ecological cell of a given simulation		start_time, x, y, z, key will be constant.
MiscParam_ Time (start_time)	List miscellaneous parameters of a given simulation		start_time will be constant. x, y, z, will be None.
MiscParam_ TimeName (start_time, parameter)	List specific world parameters of a given simulation		start_time, key will be constant. x, y, z, will be None.

SimParam_ TimeName (start_time, parameter)	Get the value of one parameter in one simulation	Value of parameter	No comments.
ParamTypes (table, to_list)	List parameters types for a table	If to_list is True, returns a list of parameter types.	No comments.

Table 1: Getter Functions for Simulation, World, and Miscellaneous Data.

Function Name	Description	Returned Data	Comments
OrgParam_Time (start_time)	List organism data for a simulation	Pandas data frame of start_time (simulation start time) pop_name (population name) org_name (organism name) generation (generation count) Key (parameter name) Value (parameter value)	start_time will be constant.
OrgParam_Pop (pop_name)	List organism data for a population		pop_name will be constant.
OrgParam_ TimePop (start_time, pop_name)	List organism data for a population within a simulation		start_time, pop_name will be constant.
OrgParam_ TimePopName (start_time, pop_name, parameter)	List specific organism parameter for a population within a simulation		start_time, pop_name, key will be constant.

Table 2: Getter Functions for Organism Data.

A pre-defined set of fitness calculation function has been implemented:

- MeasureDiameter function measures the diameter of metabolic pathways in an organism
- MeasureDensity function measures the undirected and directed density of metabolic pathways in an organism
- MeasureDiversity function measures the diversity of perception and enzymatic genes in an organism
- MeasureEfficiency function measures the local or global efficiency of metabolic pathways in an organism
- MeasureGeneSpaceUtilization function measures the gene space being utilized in an organism
- MeasureNodes function measures the number of nodes in metabolic pathways in an organism
- MeasureRichClubCoefficient function measures the absence or presence of rich clubs in metabolic pathways in an organism
- MeasureSum function measures the sum of perception or enzymatic genes in an organism
- SubsequenceCounter function counts the number of subsequences in an organism.

Future work includes (a) update the current version of DOSSIER to continually improve its usability by equipping it to incorporate better coverage of algorithms related to evolutionary computation, and (b) allow researchers familiar with DOSE to collaborate and contribute to the codebase by enhancing it to utilize expertise from different subfields of evolutionary biology.

Conclusion

As the execution of a DO simulation in DOSE results in an SQLite database, which stores all the simulation data, we present DOSSIER as a toolkit to facilitate data extraction this resulting SQLite database.

Availability

DOSSIER is part of the DOSE project (https://github.com/mauriceling/dose/) and is licensed under GNU General Public License version 3.0 for academic and non-commercial purposes only.

Conflict of Interest

The authors declare no conflict of interest.

Bibliography

1. Langton CG. "Studying Artificial Life with Cellular Automata". *Physica D: Nonlinear Phenomena* 22.1-3 (1986): 120-149.

2. Lenski RE., *et al*. "Genome Complexity, Robustness and Genetic Interactions in Digital Organisms". *Nature* 400.6745 (1999): 661-664.

3. Wilke CO and Adami C. "The Biology of Digital Organisms". *Trends in Ecology and Evolution* 17.11 (2002): 528-532.

4. Yedid G and Bell G. "Microevolution in an Electronic Microcosm". *The American Naturalist* 157.5 (2001): 465-487.

5. Adami C. "Digital Genetics: Unravelling the Genetic Basis of Evolution". *Nature Reviews Genetics* 7.2 (2006): 109-118.

6. Wilke CO., *et al*. "Evolution of Digital Organisms at High Mutation Rates Leads to Survival of the Flattest". *Nature* 412.6844 (2001): 331-333.

7. Wilke CO and Christoph A. "Interaction Between Directional Epistasis and Average Mutational Effects. *Proceedings of the Royal Society of London Series B: Biological Sciences* 268.1475 (2001): 1469-1474.

8. Grabowski LM., *et al*. "A Case Study of the De Novo Evolution of a Complex Odometric Behavior in Digital Organisms". *Plos One* 8.4 (2013): e60466.

9. Gupta A., *et al*. "Evolution of Genome Size in Asexual Digital Organisms. *Scientific Reports* 6.1 (2016): 25786.

10. Pennock RT. "Models, Simulations, Instantiations, and Evidence: The Case of Digital Evolution. *Journal of Experimental and Theoretical Artificial Intelligence* 19.1 (2007): 29-42.

11. Castillo CF and Ling MH. "Digital Organism Simulation Environment (DOSE): A Library for Ecologically-Based In Silico Experimental Evolution". *Advances in Computer Science : An International Journal* 3.1 (2014): 44-50.

12. Ray TS. Evolution and Optimization of Digital Organisms. Scientific Excellence in Supercomputing: The IBM 1990 Contest Prize Papers (1991).

13. Ofria C and Wilke CO. "Avida: A Software Platform for Research in Computational Evolutionary Biology". *Artificial Life* 10.2 (2004): 191-229.

14. Liard V., *et al*. "The Complexity Ratchet: Stronger than Selection, Stronger than Evolvability, Weaker than Robustness". *Artificial Life* 26.1 (2020): 38-57.

15. Castillo CF and Ling MH. "Digital Organism Simulation Environment (DOSE) Version 1.0.4". *Current STEM, Volume 1* (Nova Science Publishers, Inc.) (2018): 1-106.

16. Ling MH. "An Artificial Life Simulation Library Based on Genetic Algorithm, 3-Character Genetic Code and Biological Hierarchy". *The Python Papers* 7.5 (2012): 1-30.

17. Ling MH. "Ragaraja 1.0: The Genome Interpreter of Digital Organism Simulation Environment (DOSE)". *The Python Papers Source Codes* 4.2 (2012): 1-45.

18. Ang DG and Ling MH. "Sudden and Steep Harsh Environment Results in Over-Compensation in Digital Organisms". *EC Microbiology* 17.7 (2021): 104-113.