Numerical Experiments for testing Demand-Driven Deployment Algorithms

Gwendolyn Chee, Jin Whan Bae and Kathryn D. Huff

Dept. of Nuclear, Plasma and Radiological Engineering, University of Illinois at Urbana-Champaign
gchee2@illinois.edu

INTRODUCTION

For many fuel cycle simulators, it is currently up to the user to define a deployment scheme for each component of the fuel cycle to avoid gaps in the supply chain. This same goal could also be achieved by setting all the facility's capacities to infinity. However, this does not reflect real-world conditions [1]. To address this gap in capability of fuel cycle simulators, the Demand-Driven Cycamore Archetype project (NEUP-FY16-10512) is developing prediction algorithms to give Cyclus demand-driven deployment capabilities. This means that Cyclus will have the capability to deploy supporting fuel cycle facilities to meet front-end and back-end demands of the fuel cycle [1]. The project is a collaboration between University of Illinois Urbana-Champaign and University of South Carolina. This paper will discuss the numerical experiments required to test the various prediction algorithms designed for the project. In particular, the non-optimizing algorithm.

BACKGROUND: CYCLUS

Cyclus [2] is an agent-based extensible framework for modeling flow of material through user-defined nuclear fuel cycles [3]. Cycamore [4] is an additional modules repository in the Cyclus ecosystem that provides basic libraries to represent process physics of various components of the nuclear fuel cycle (ie. mining, fuel enrichment, reactor) [5]. Each library is an archetype.

Cyclus simulations are composed of discrete time steps. Each time step is subdivided into phases. The time step execution phases for Cyclus is illustrated in Figure 1.

BACKGROUND: DEMAND-DRIVEN DEPLOYMENT ALGORITHMS

Nuclear fuel cycle simulation scenarios are described as constrained objective functions. This means that the goal of the simulations are to optimize an objective function with respect to constraints on certain variables. For nuclear fuel cycle simulations, examples for the objective function are percentage energy growth and uranium utilization percentage. Examples for constraints are the availability of new nuclear fuel cycle technology such as specific types of reprocessing. This necessitates demand responsive deployment capabilities to be added to fuel cycle simulation logic. The simulator should have the capabilities to deploy supporting fuel cycle facilities to enable a demand to be met. For example, for a once through fuel cycle with an energy growth demand of 1% per year, the simulator should have the capabilities to optimally deploy supporting facilities such as a mine, enrichment facility and reactor to meet the demand [6].

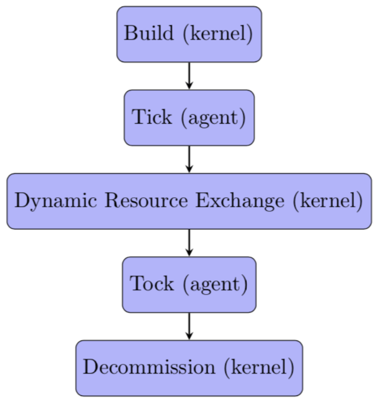

Fig. 1. Each time step in Cyclus follows the five phases in order. Phases labeled kernel have required actions that occur and phases labeled agent are executed by individual agents. Agent refers to specific facilities within the simulation. What happens in "Tick" and "Tock" are unique to each archetype [1].

METHOD: PREDICTION ALGORITHMS

For this project, three prediction algorithm types are considered: non-optimizing methods, deterministic optimization and stochastic optimization. They are listed in level of effectiveness and difficulty of design. These prediction models are currently being developed by the USC team. Essentially, each algorithm will create a supply chain of reactor and supporting fuel facilities. At every time step, the demand for each fuel cycle commodity will be evaluated and the algorithm will make a prediction about future demand, resulting in the deployment or decommissioning of facilities [1].

The non-optimizing method type is the most basic optimization algorithm. It predicts future deployment schedules solely based on historical data. In the tick phase, the difference in supply and demand for each commodity is evaluated. Based on the size of the difference and capacity of the corresponding facility, facilities will be deployed or decommissioned. Methods that are used include autoregressive moving average (ARMA) and autoregressive conditional heteroskedastic (ARCH) methods. Both ARMA and ARCH rely on an autoregressive model which means that the predicted future values of a time series depend on the previous values of that same time series [7].

METHOD: TESTS

In Best Practices of Scientific computing [8], Wilson et al highlights the fact that similar to building experimental apparatus, constructing software requires careful building and validation to ensure their reliability. Furthermore, Wilson et al discusses that because software is commonly reused, it can result in a negative long term effect on the integrity of the group's work if a bug is not found.

An important practice for verification and maintenance of code is to write and run tests [8]. Automated tests ensure that a piece of code is functioning the way it is intended. The most basic test is the unit test which refers to the testing of a single function [8]. For the algorithms, unit tests will be written for each section of code to ensure all components of the code are reliable. Testing may not be perfect at capturing all the bugs, however, it minimizes them.

The goal is for the demand-driven deployment algorithms to be integrated with the Cyclus framework in the long term and used to optimize simulations of transition analyses. Therefore, if the algorithms are not well tested and have logic flaws or bugs, this would result in inaccuracies with the conclusions drawn from the transition analyses simulations. And further compromise any experiments that use the algorithm in the future [8].

Test Example

A simple once through fuel cycle scenario is sufficient to verify the non-optimizing algorithm. The scenario used has only four facilities. Figure 2 depicts the material flow between the four facilities.

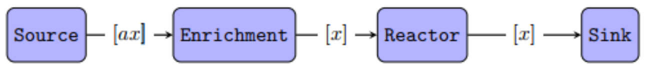

Fig. 2. Simple demand flow of materials. The bracketed values are demands calculated in the algorithm. The Reactor demands *x* amount of fuel which translates to *x* demand from enrichment and *ax* demand from source, taking into account enrichment losses [1].

Reactor Parameters	**Value**	**Units**
Lifetime	3	Timesteps
Power Capacity	1000	MWe
Assembly Size	100	kg
# Assemblies per core	3	
Enrichment Facility parameters	**Value**	**Units**
SWU Capacity	2000	SWU/timestep
Enrichment SWU for 1 assembly	528	SWU
Fuel output	300	kg/timestep

TABLE I. Simple once through nuclear fuel cycle scenario parameters [1]

A unit test is written for each function of the algorithm to test if its output matches the analytical solution for the specified test scenario [1]. To ensure that the deployment and decommissioning of the facilities is appropriate, the tick phase must be working correctly. There are three main types of tests that will ensure this. The first test is that the difference between demand and supply for each commodity is below the capacity of its respective facility for all time steps. The second test is that the correct number of facilities are deployed at each time step. The third test is that the correct number of facilities are decommissioned at each time step.

An example for each type of test described above is for the enrichment facility whose output commodity is fuel. Table I describes the scenario parameters that are relevant for both example tests.

The first test example checks if the difference between enrichment facility fuel supply and reactor fuel demand is within plus-minus the output capacity of one enrichment facility for every time step. For this test, a reactor is deployed at time steps 2 and 3 and a fuel supply of 100kg is given at the initial time step. Since each reactor has 3 assemblies per core, a fuel demand of 300kg is required per reactor at the time step it is deployed. The analytical solution for this test is shown in table II.

Timestep	**Fuel Quantity (kg)**	**Fuel Demand (kg)**	**Difference (kg)**
1	100	0	100
2	400	300	100
3	400	300	100

TABLE II. Analytical solution of the difference between fuel quantity and fuel demand per time step for a test scenario where a reactor is deployed at time step 2 and 3 and an initial fuel quantity of 100kg at time step 1 [1].

The second test example checks if an enrichment facility is deployed when the amount of fuel available is below the fuel demand of the reactor. For this test, a reactor is deployed at time step 2. Since an enrichment facility outputs 300kg of fuel per timestep, one enrichment facility must be deployed at time step 2 to meet the fuel demand. The analytical solution for this test is shown in table III.

Timestep	**Enrichment Facility deployment**
1	0
2	1
3	0

TABLE III. Analytical solution of the number of fuel facilities deployed per time step for a test scenario where a reactor is deployed at time step 2. [1]

Timestep	**Enrichment Facility deployment**
1	0
2	1
3	1
4	1
5	0

TABLE IV. Analytical solution of the number of enrichment facilities deployed per time step for a test scenario where a reactor is deployed at time step 2 and decommissioned at time step 5.

The third test example checks if an enrichment facility is decommissioned when the amount of fuel available is above the fuel demand of the reactor. For this test, a reactor is deployed at time step 2. Since a reactor has a lifetime of 3 timesteps, it is decommissioned at time step 5. Therefore, the enrichment facility must also be decommissioned at time step 5. The analytical solution for this test is shown in table IV.

Each of these tests will be implemented for each commodity in the supply chain. Both the Demand-Driven Deployment algorithms and tests are implemented in the Python language.

ACKNOWLEDGMENTS

This work is supported by U.S. Department of Energy's Nuclear Energy University Program under contract # NEUP-FY16-10512. This is a joint project with University of South Carolina. The USC team comprising of Dr. Robert Flanagan and Dr. Anthony Scopatz is developing the prediction models.

REFERENCES

1. J. W. BAE, G. CHEE, and K. D. HUFF, "Numerical Experiments for Verifying Demand Driven Deployment Algorithms," Graduate Report, University of Illinois at Urbana-Champaign, Urbana, IL (Jan. 2018).
2. R. W. CARLSEN, M. GIDDEN, K. HUFF, A. C. OPOTOWSKY, O. RAKHIMOV, A. M. SCOPATZ, Z. WELCH, and P. WILSON, "Cyclus v1.5.3," *Figshare* (Jun. 2014), http://dx.doi.org/10.6084/m9.figshare.1041745.
3. K. D. HUFF, M. J. GIDDEN, R. W. CARLSEN, R. R. FLANAGAN, M. B. MCGARRY, A. C. OPOTOWSKY, E. A. SCHNEIDER, A. M. SCOPATZ, and P. P. H. WILSON, "Fundamental concepts in the Cyclus nuclear fuel cycle simulation framework," *Advances in Engineering Software*, **94**, 46–59 (Apr. 2016).
4. R. W. CARLSEN, M. GIDDEN, K. HUFF, A. C. OPOTOWSKY, O. RAKHIMOV, A. M. SCOPATZ, and P. WILSON, "Cycamore v1.5.3," *Figshare* (Jun. 2014), http://figshare.com/articles/Cycamore_v1_0_0/1041829.
5. K. D. HUFF, M. FRATONI, and H. R. GREENBERG, "Extensions to the cyclus ecosystem in support of market-driven transition capability," Tech. rep., Lawrence Livermore National Laboratory (LLNL), Livermore, CA (2014).
6. K. D. HUFF, J. W. BAE, K. A. MUMMAH, R. R. FLANAGAN, and A. M. SCOPATZ, "Current Status of Predictive Transition Capability in Fuel Cycle Simulation," in "Proceedings of Global 2017," Seoul, South Korea (Sep. 2017).
7. A. M. SCOPATZ and K. D. HUFF, "Technical Narrative for Demand-Driven Cycamore Archetypes," Technical Report, University of South Carolina and University of Illinois at Urbana-Champaign (2016).
8. G. WILSON, D. A. ARULIAH, C. T. BROWN, N. P. CHUE HONG, M. DAVIS, R. T. GUY, S. H. D. HADDOCK, K. D. HUFF, I. M. MITCHELL, M. D. PLUMBLEY, B. WAUGH, E. P. WHITE, and P. WILSON, "Best Practices for Scientific Computing," *PLoS Biol*, **12**, *1*, e1001745 (Jan. 2014).