Numerical Experiments for Verifying Demand Driven Deployment Algorithms

Non-Optimizing Algorithm

Jin Whan Bae, Gwendolyn Chee, Kathryn Huff

August 14, 2018

1 Project Objective

The Demand-Driven Cycamore Archetype project (NEUP-FY16-10512) aims to develop CYCAMORE's demand-driven deployment capabilities. The developed algorithm will be in the form of a CYCLUS `Institution` agent, and will deploy `Facilities` to meet the front-end and back-end demands of the fuel cycle.

2 Motivation

The current CYCLUS fuel cycle simulation ~~code~~ framework relies on the user to define a deployment scheme or set the supporting ~~f~~F`acilities` capacities to infinity to ensure that there's no gap in the nuclear fuel cycle supply chain. These user-defined assumptions are not an accurate reflection of the real world.

3 Method

The project objective is met by developing three types of predictive algorithms: non-optimizing, deterministic-optimizing and stochastic-optimizing. Each algorithm aims to improve on the previous to provide more accurate prediction results.

Meanwhile,

The prediction algorithms are being developed by the team at University of South Carolina. While the numerical experiments are being designed by the team at University of Illinois at Urbana-Champaign.

Institution

This report will focus on the non-optimizing algorithm. It lists capability requirements of the non-optimizing case of the new Cyclus `institute` for demand-driven deployment of fuel cycle facilities. It also discusses the tests to check correct implementation of the capabilities, using a sample fuel cycle with well-defined facility parameters.

4 Archetype Requirements

algorithms

Subsections 4.1 to 4.4 state the requirements that apply to all three predictive algorithms. Expectations for the non-optimizing algorithm are different ~~than for~~ the deterministic-optimizing and stochastic-optimizing. Therefore, in subsections 4.5 to 4.6, the requirements unique to the non-optimizing algorithm are specified.

from expectations for

4.1 User Configuration

The archetype should allow the user to define the following parameters:

1. A commodity whose demand drives deployment
2. Initial amount of that demand
3. Rate of growth or decline of that demand
4. The facilities in the simulation able to meet that demand
5. Algorithm type: non-optimizing, deterministic optimizing or stochastic optimizing

4.2 Create a supply chain

The archetype should be able to access the user defined parameters for the facilities in the simulation and evaluate if the commodities supplied by these facilities produce an appropriate supply chain to meet the demand of the commodity whose demand drives deployment. If not, the archetype should inform the user by throwing an error.

4.3 High-level Functionality

If the user chooses a positive demand growth equation for power and the facilities present in the simulation to include reactors and supporting fuel facilities. The demand for power will trigger reactor deployment which will cause an increase in the demand for fuel, which will in turn trigger fuel facility deployment.

??

diagram appropriate here

4.4 Growth or Decline Rate

The archetype should give the user options for different types of growth ~~or decline~~ rate curves. Possible examples could be linear, exponential or piece-wise.

4.5 Facility Deployment and Decommissioning

The non-optimizing algorithm's deployment and decommissioning capabilities should be based on previous demand and supply values. At each time step, the algorithm should evaluate the demand for each commodity against its corresponding supply. If there is a shortage or surplus, the algorithm should deploy new facilities or decommission existing facilities. With an exception to the situation where decommissioning of a facility due to surplus of a commodity will result in the shortage of the commodity. This is a reflection of reality, since it is acceptable to have some surplus in storage as opposed to shortage of a commodity that could result in non-maximum capacity of a nuclear reactor.

??

I think capacity should be "availability" perhaps. Without enough fuel, the reactor shuts down. It doesn't just run at 80%.

4.6 Dealing with volatility

A comprehensive fuel cycle simulator must have predictive capabilities which can deploy fuel cycle support facilities intelligently even in the face of volatile dynamics. ~~In the situation where the commodity whose demand drives deployment changes in a volatile manner~~ If demand for the deployment-driving commodity is volatile, the archetype should ~~be able to~~ recognize this. ~~and not~~ If it fails, the archetype could pathologically deploy and decommission the same facility across short time periods due to volatile changes in commodity demand.

5 Simulation parameter for Test Scenarios

Simple ~~parameters are given to~~ fuel cycle facilities ~~for~~ populate the numerical testing of the algorithm. Only `source` and `reactor` facilities are used in the test scenarios.

Table 1 provides basic parameters for each test scenario. Table 2 provides the parameters for the S `source`, R `reactors` and S `sink` Facilities in the test scenarios.

Table 1: Basic Test Parameters

Test Scenario Parameters	**Value**	**Units**
Duration	15	timesteps
Timestep	1	month
Start Month	1	month
Start Year	2000	year

Table 2: Source, Reactor and Sink Parameters

Source Parameters	**Value**	**Units**
Throughput	1	kg
Output Commodity	fuel	kg
Reactor Parameters	**Value**	**Units**
Cycle Time	1	timesteps
Refuel Time	0	timesteps
Lifetime	1	timesteps
Power Capacity	1	MWe
Assembly Size	1	kg
# assemblies per core	1	
# assemblies per batch	1	
Input Commodity	fuel	kg
Output Commodity	power	MW
Sink Parameters	**Value**	**Units**
Throughput	1	kg
Input Commodity	spent uox	kg

6 Numerical Tests for the Non-optimizing prediction method

The non-optimizing prediction method is tested by comparing its output for various scenarios against their analytical solutions . In this section, the tests that must be met is described based on the parameters defined in table 1 and 2 and analytical solution of a defined simple scenario. Unit test examples are included in Appendix B.

trajectory? ? ?

The tests are in forms of `[alphabet]-[traj]-[num]`. `Alphabet` refers to the supply chain of the test scenario, `traj` the demand change in time, and `num` for tests with the same `alphabet` and `traj`.

I think you mean the test naming convention is in this form.

unclear what this means.

- **A** - source → `End Demand Commodity`
- **B** - source → reactor → `End Demand Commodity`

would be helpful to use a real example, for human understanding (and debugging) (eg. fresh fuel?)

Note that for test C, the `End Demand Commodity` is power generated from the reactor. The C-tests additionally check if the sink is deployed to meet spent fuel disposal demand.

The prediction algorithm for the non-optimizing method has three user-defined input parameters. The aim of the various test scenarios ~~are~~ is to check if the non-optimizing method archetype will deploy or decommission facilities correctly when there is a variation in the combination of the three input parameters. The input parameters are:

1. Initial demand value
2. Number of initial facilities (initial supply)
3. Growth rate of initial demand

The growth in demand is governed by the Equation 1.

$$D_f(timestep) = D_i(1+g)^{(\frac{timestep}{12})} \tag{1}$$

Where D_f is demand of resource at specific time step, D_i is initial demand and g is growth rate.

`Source` and R`reactor` facilities are used in the test scenarios. Test scenarios A1 to A4 and B1 to B2 only have a `source` facility and test scenarios A5 to A7 have both S`source` and R`reactor` facilities. For each test scenario, there is one table that states the test scenario's input parameters and another table that states the exact analytical solution. The analytical solution

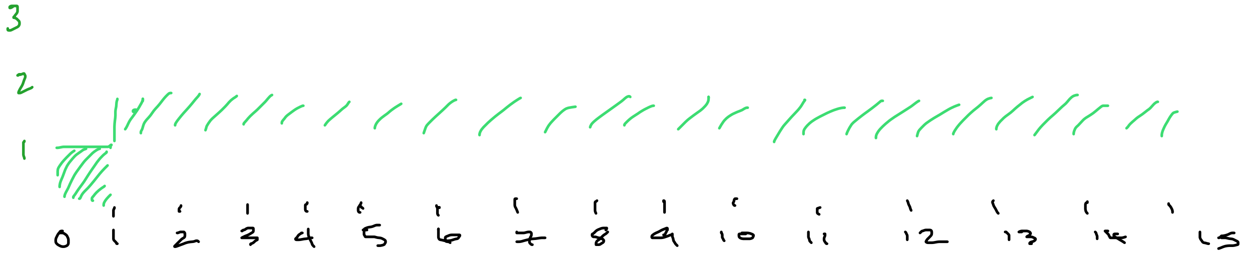

table does not include deployment of the initial facility that is stated in the first table.

Additionally, we created base tests for each A-type test scenario that passes when the supply meets the demand within a given facility number tolerance. In other words, when the supply exceeds the demand by the specified tolerance quantity, the test still passes. For this report, the tolerance is set to one facility. For example in test A-1, the expected total number facilities deployed is 1, and since the facility over-prediction tolerance is 1, the acceptable range of total number of facilities (x) deployed in the entire test scenario is $1 < x < 2$. If the total number of facilities deployed is within this range, the base case test will pass.

6.1 Test A-const-1

In test A-const-1, only a `source` facility is present in the test scenario. Table 3 shows the input parameters of the source facility in the test scenario. Table 4 shows the expected analytical solution based on the test scenario. Table 5 shows the accepted range of total number of facilities deployed over the test scenario which will pass the base test, which factors in the facility over-prediction tolerance of 1.

Table 3: Test A-const-1 Scenario Input Parameters

Source Parameter	**Value**	**Units**
Initial demand	1	kg
Initial facilities	0	#
Growth Rate	0	

Table 4: Test A-const-1 Analytical Solution

Time Step	**No. of Source Facilities Deployed**
1	1
2 to 15	0

perhaps a plot would communicate this more clearly.

Table 5: Test A-const-1 Base Test Acceptance

Acceptable total No. of Source Facilities Deployed + tolerance
1 < x < 2

should be ≤ ?

6.2 Test A-const-2

In test A-const-2, only a S`source` facility is present in the test scenario. Table 6 shows the input parameters of the source facility in the test scenario. Table 7 shows the expected analytical solution based on the test scenario.Table 8 shows the accepted range of total number of facilities deployed over the test scenario which will pass the base test, which factors in the facility over-prediction tolerance of 1.

Table 6: Test A-const-2 Scenario Input Parameters

Source Parameter	Value	Units
Initial demand	2	kg
Initial facilities	1	#
Growth Rate	0	

Table 7: Test A-const-2 Analytical Solution

Time Step	No. of Source Facilities Deployed
1	1
2 to 15	0

maybe a plot?

Table 8: Test A-const-2 Base Test Acceptance

Acceptable total No. of Source Facilities Deployed + tolerance
$2 < x < 3$

not ≤ ?

6.3 Test A-grow-1

In test A-grow-1, only a S`source` facility is present in the test scenario. Table 9 shows the input parameters of the source facility in the test scenario. Table 10 shows the expected analytical solution based on the test scenario. Table 11 shows the accepted range of total number of facilities deployed over the test scenario which will pass the base test, which factors in the facility over-prediction tolerance of 1.

Table 9: Test A-grow-1 Scenario Input Parameters

Source Parameter	Value	Units
Initial demand	1	kg
Initial facilities	0	#
Growth Rate	1	

Table 10: Test A-grow-1 Analytical Solution

plot?

Time Step	No. of Source Facilities Deployed
1	2
2 to 12	0
13	1
14 to 15	0

Table 11: Test A-grow-1 Base Test Acceptance

Acceptable total No. of Source Facilities Deployed + tolerance
$3 < x < 4$

≤

6.4 Test A-grow-2

In test A-grow-2, only a `source` facility is present in the test scenario. Table 12 shows the input parameters of the source facility in the test scenario. Table 13 shows the expected analytical solution based on the test scenario. Table 14 shows the accepted range of total number of facilities deployed over the test scenario which will pass the base test, which factors in the facility over-prediction tolerance of 1.

Table 12: Test A-grow-2 Scenario Input Parameters

Source Parameter	Value	Units
Initial demand	1	kg
Initial facilities	1	#
Growth Rate	1	

Table 13: Test A-grow-2 Analytical Solution

Time Step	**No. of Source Facilities Deployed**
1	1
2 to 12	0
13	1
14 to 15	0

Table 14: Test A-grow-2 Base Test Acceptance

Acceptable total No. of Source Facilities Deployed + tolerance
3 < x < 4

plot?

≤

6.5 Test B-const-1

In test B-const-1, both a `source` and `reactor` facility is present in the test scenario. Table 15 shows the input parameters of the source facility in the test scenario. Table 16 shows the expected analytical solution based on the test scenario. Table 17 shows the accepted range of total number of facilities deployed over the test scenario which will pass the base test, which factors in the facility over-prediction tolerance of 1.

Table 15: Test B-const-1 Scenario Input Parameters

Source Parameter	**Value**	**Units**
Initial demand	1	kg
Initial facilities	0	#
Growth Rate	0	
Reactor Parameter	**Value**	**Units**
Initial demand	1	MW
Initial facilities	0	#
Growth Rate	0	

Table 16: Test B-const-1 Analytical Solution

Time Step	No. of Source Facilities Deployed	No. of Reactor Facilities Deployed
1	1	1
2 to 15	0	0

plot?

Table 17: Test B-const-1 Base Test Acceptance

Acceptable total No. of Source Facilities Deployed + tolerance	Acceptable total No. of Reactor Facilities Deployed + tolerance
$1 < x < 2$	$1 < x < 2$

6.6 Test B-const-2

In test B-const-2, both a `source` and `reactor` facility is present in the test scenario. Table 18 shows the input parameters of the source facility in the test scenario. Table 19 shows the expected analytical solution based on the test scenario. Table 20 shows the accepted range of total number of facilities deployed over the test scenario which will pass the base test, which factors in the facility over-prediction tolerance of 1.

Table 18: Test B-const-2 Scenario Input Parameters

Source Parameter	Value	Units
Initial demand	1	kg
Initial facilities	1	#
Growth Rate	0	
Reactor Parameter	**Value**	**Units**
Initial demand	1	MW
Initial facilities	1	#
Growth Rate	0	

Table 19: Test B-const-2 Analytical Solution

Time Step	No. of Source Facilities Deployed	No. of Reactor Facilities Deployed
1	1	1
2 to 15	0	0

plot?

Table 20: Test B-const-2 Base Test Acceptance

Acceptable total No. of Source Facilities Deployed + tolerance	**Acceptable total No. of Reactor Facilities Deployed + tolerance**
$1 < x < 2$	$1 < x < 2$

6.7 Test B-grow-1

In test B-grow-1, both a `source` and `reactor` facility is present in the test scenario. Table 21 shows the input parameters of the source facility in the test scenario. Table **??** shows the expected analytical solution based on the test scenario. Table 23 shows the accepted range of total number of facilities deployed over the test scenario which will pass the base test, which factors in the facility over-prediction tolerance of 1.

Table 21: Test B-grow-1 Scenario Input Parameters

Source Parameter	**Value**	**Units**
Initial demand	1	kg
Initial facilities	0	#
Growth Rate	1	
Reactor Parameter	**Value**	**Units**
Initial demand	1	MW
Initial facilities	0	#
Growth Rate	1	

Table 22: Test B-grow-1 Analytical Solution

Time Step	**No. of Source Facilities Deployed**	**No. of Reactor Facilities Deployed**
1	2	2
2 to 12	0	0
13	1	1
14 to 15	0	0

plot?

Table 23: Test B-grow-1 Base Test Acceptance

Acceptable total No. of Source Facilities Deployed + tolerance	**Acceptable total No. of Reactor Facilities Deployed + tolerance**
$3 < x < 4$	$3 < x < 4$

6.8 Test A-const-3

In test A-const-3, only a `source` facility is present in the test scenario. Table 24 shows the input parameters of the source facility in the test scenario. Table 25 shows the expected analytical solution based on the test scenario.

Table 24: Test A-const-3 Scenario Input Parameters

Source Parameter	Value	Units
Initial demand	0	kg
Initial facilities	1	#
Growth Rate	0	

Table 25: Test A-const-3 Analytical Solution

	No. of Source	No. of Source
Time Step	Facilities Deployed	Facilities Decomissioned
1	1	0
2	0	1
3 to 15	0	0

plot?

6.9 Test A-decl-1

In test A-decl-1, only a `source` facility is present in the test scenario. Table 26 shows the input parameters of the source facility in the test scenario. Table 27 shows the expected analytical solution based on the test scenario.

Table 26: Test A-decl-1 Scenario Input Parameters

Source Parameter	Value	Units
Initial demand	1	kg
Initial facilities	1	#
Growth Rate	-1	

Table 27: Test A-decl-1 Analytical Solution

	No. of Source	No. of Source
Time Step	Facilities Deployed	Facilities Decomissioned
1 to 12	1	0
13	0	1
14 to 15	0	0

plot?

7 Numerical Test Results

It was found that none of the exact tests and base case tests passed. The failures are attributed to three reasons:

1. There is a test failure when there is no initial facility present. An initial condition must be given for the algorithm to understand the capacity of the facility it deploys.

2. There is a test failure when there is growth in the demand. The algorithm failed to deploy facilities to meet the increase in demand.

3. There is a test failure when facilities are expected to be decommissioned. There is yet to be an implementation on decommissioning behavior.

Despite failures in these situations, the non-optimizing method proved to have commissioning capabilities to meet demand. Because of the limitation in the algorithm, it is hard to predict demand with precision in such a short time. However, the same numerical experiments can be applied to the deterministic optimizing algorithm, which holds more promise. The goal is that the deterministic optimization method will be able to overcome the issues faced by the non-optimizing method with reference to the requirement of an initial condition, growth in demand. Also, the capability to decommission facilities upon oversupply will be added.

Appendix C reflects the numerical experiment solution output by the non-optimizing prediction algorithm for each test scenario defined in section 6.

8 References

References

Appendix A - parameter configuration

Appendix A shows the json file that contains the simulation parameters that are common between all the test scenarios discussed in Section 6.

```
template = {
"simulation": {
"archetypes": {
"spec": [
{"lib": "agents", "name": "NullRegion"},
{"lib": "cycamore", "name": "Source"},
{"lib": "cycamore", "name": "Reactor"},
{"lib": "cycamore", "name": "Sink"},
{"lib": "d3ploy.no_inst", "name": "NOInst"}
]
},
"control": {"duration": "15", "startmonth": "1", "startyear": "2000"},
"recipe": [
{
"basis": "mass",
"name": "fresh_uox",
"nuclide": [{"comp": "0.711", "id": "U235"}, {"comp": "99.289", "id": "U238"}]
},
{
"basis": "mass",
"name": "spent_uox",
"nuclide": [{"comp": "50", "id": "Kr85"}, {"comp": "50", "id": "Cs137"}]
}
],
"facility": [{
"config": {"Source": {"outcommod": "fuel",
"outrecipe": "fresh_uox",
"throughput": "1",
"source_record_supply": "fuel"}},
"name": "source"
},
{
"config": {"Sink": {"in_commods": {"val":"spent_uox"},
"max_inv_size": 1,
"sink_record_demand": "fuel_cap"}},
"name": "sink"
},
{
"config": {
"Reactor":{
"assem_size":"1",
"cycle_time": "1",
"fuel_incommods": {"val": "fuel"},
"fuel_inrecipes": {"val": "fresh_uox"},
"fuel_outcommods": {"val": "spent_uox"},
"fuel_outrecipes": {"val": "spent_uox"},
"n_assem_batch": "1",
"n_assem_core": "1",
"power_cap": "1",
"refuel_time": "0",
"reactor_fuel_demand": "fuel_reactor"
}
},
"name": "reactor"
}]}}
```

Appendix B - Sample Test Code

Sample test code for test A-const-1

Appendix B shows the python file that contains the a segment of the simulation parameters that are unique to test A-const-1 and the code for test A-const-1.

```
# Test A_const_1
INIT_DEMAND = copy.deepcopy(TEMPLATE)
INIT_DEMAND["simulation"].update({"region": {
"config": {"NullRegion": "\n      "},
"institution": {
"config": {
"NOInst": {
"calc_method": "arma",
"demand_commod": "POWER",
"demand_std_dev": "0.0",
"growth_rate": "0.0",
"initial_demand": "1",
"prototypes": {"val": "source"},
"steps": "1",
"supply_commod": "fuel"
}
},
"name": "source_inst"
},
"name": "SingleRegion"
}})

@pytest.mark.base
def test_a1_init_demand():
# tests if NOInst deploys a source
# given initial demand and no initial facilities
output_file = 'init_file.sqlite'
input_file = output_file.replace('.sqlite', '.json')
with open(input_file, 'w') as f:
json.dump(INIT_DEMAND, f)
s = subprocess.check_output(['cyclus', '-o', output_file, input_file],
universal_newlines=True, env=ENV)
# check if ran successfully
```

```
assert("Cyclus run successful!" in s)

# getting the sqlite file
cur = get_cursor(output_file)

# check base solution
source_base = cur.execute(query).fetchone()
assert(1 <= source_base[0] <= (1 + tol))

@pytest.mark.exact
def test_a1_init_demand_exact():
output_file = 'init_file.sqlite'
cur = get_cursor(output_file)
# check exact solution
source_exact = cur.execute(query + " AND EnterTime = 1").fetchone()
assert(source_exact[0] == 1)
```

Appendix C - Numerical Experiment Solution for test scenarios

Test A-const-1

Table 28: Test A-const-1 Numerical Experiment Solution

Time Step	No. of Source Facilities Deployed
1 to 15	0

Test A-const-2

Table 29: Test A-const-2 Numerical Experiment Solution

Time Step	No. of Source Facilities Deployed
1	0
2	1
3	1
4	1
5	0
6	1
7 to 15	0

Test A-grow-1

Table 30: Test A-grow-1 Numerical Experiment Solution

Time Step	No. of Source Facilities Deployed
1 to 15	0

Test A-grow-2

Table 31: Test A-grow-2 Numerical Experiment Solution

Time Step	No. of Source Facilities Deployed
1 to 15	0

Test B-const-1

Table 32: Test B-const-1 Numerical Experiment Solution

Time Step	**No. of Source Facilities Deployed**	**No. of Reactor Facilities Deployed**
1 to 15	0	0

Test B-const-2

Table 33: Test B-const-2 Numerical Experiment Solution

Time Step	**No. of Source Facilities Deployed**	**No. of Reactor Facilities Deployed**
1	0	0
2	1	0
3	1	0
4	1	0
5	0	0
6	1	0
7 to 15	0	0

Test B-grow-1

Table 34: Test B-grow-1 Numerical Experiment Solution

Time Step	**No. of Source Facilities Deployed**	**No. of Reactor Facilities Deployed**
1 to 15	0	0

Test A-const-3

Table 35: Test A-const-3 Numerical Experiment Solution

Time Step	**No. of Source Facilities Deployed**	**No. of Source Facilities Decomissioned**
1 to 15	0	0

Test A-decl-1

Table 36: Test A-decl-1 Numerical Experiment Solution

Time Step	**No. of Source Facilities Deployed**	**No. of Source Facilities Decomissioned**
1 to 15	0	0