Demand Driven Deployment Capabilities in Cyclus, a Fuel Cycle Simulator

Gwendolyn J. Chee[a], Roberto E. Fairhurt Agosta[a], Jin Whan Bae[b], Robert R. Flanagan[c], Anthony Scopatz[c], Kathryn D. Huff[a,*]

[a] *Dept. of Nuclear, Plasma, and Radiological Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801*
[b] *Oak Ridge National Laboratory, Oak Ridge, TN, United States*
[c] *Nuclear Engineering Program, University of South Carolina*

Abstract

Keywords: nuclear fuel cycle, python, time series forecasting

1. Introduction

For many fuel cycle simulators, reactor facilities are automatically deployed to meet a user-defined power demand. However, it is up to the user to define a deployment scheme of supporting facilities to ensure that there is no gap in the supply chain that results in idle reactor capacity. Some users choose to set supporting facilities to have an infinite capacity to avoid this issue, but this is an inaccurate representation of reality. It is straightforward to manually determine a deployment scheme for a once-through fuel cycle, however, it is difficult to effectively implement for complex closed fuel cycle scenarios. To ease setting up of realistic Nuclear Fuel Cycle (NFC) simulations, a Nuclear Fuel Cycle Simulator (NFCSim) should bring demand responsive deployment decisions into the dynamics of the simulation logic [1]. Thus, a next generation NFCSim should predictively and automatically deploy fuel cycle facilities to meet a user defined power demand.

Cyclus is an agent-based nuclear fuel cycle simulation framework [2]. In Cyclus, each entity (i.e. `Region`, `Institution`, or `Facility`) in the fuel

*Corresponding Author
Email address: kdhuff@illinois.edu (Kathryn D. Huff)

Fuel Cycle	Open or Closed	Fuel Type	Reactor Type
EG01 (current)	Open	Enriched-U	Thermal critical reactors
EG23	Closed	Recycle of U/Pu with natural-U fuel	Fast critical reactors
EG24	Closed	Recycle of U/TRU with natural-U fuel	Fast critical reactors
EG29	Closed	Recycle of U/Pu with natural-U fuel	Fast critical reactors and thermal critical reactors
EG30	Closed	Recycle of U/TRU with natural-U fuel	Fast critical reactors and thermal critical reactors

Table 1: Descriptions of the current and other high performing nuclear fuel cycle evaluation groups described in the evaluation and screening study [4].

cycle is an agent. `Region` agents represent geographical or political areas that institution and facility agents can be grouped into. `Institution` agents control the deployment and decommission of facility agents and represent legal operating organizations such as a utility, government, etc. [2]. `Facility` agents represent nuclear fuel cycle facilities. Cycamore [3] provides facility agents to represent process physics of various components in the nuclear fuel cycle (e.g. mine, fuel enrichment facility, reactor).

1.1. Context of Work

An evaluation and screening study of a comprehensive set of nuclear Fuel Cycle Options (FCO) [4] was conducted to assess for performance improvements compared to the existing once-through fuel cycle (EG01) in the United States (US) across a wide range of criteria. Fuel cycles that involved continuous recycling of co-extracted U/Pu or U/TRU in fast spectrum critical reactors consistently scored high on overall performance. Table 1 provides a description of these fuel cycles: EG23, EG24, EG29, and EG30.

The evaluation and screening study assumed that the nuclear energy system was at an equilibrium to understand the end-state benefits of each Evaluation Group (EG). Based on the results from the study, the next step is to understand

and evaluate the transition from the initial EG01 state to these promising future end-states [5]. To successfully conduct analysis of the time-dependent transition analyses, it is necessary to develop NFCSim tools to automate setting up of transition scenarios. Therefore, the Demand-Driven Cycamore Archetypes project (NEUP-FY16-10512) was initiated to develop demand-driven deployment capabilities in Cyclus. This capability, `d3ploy`, is a Cyclus `Institution` agent that deploys facilities to meet the front-end and back-end fuel cycle demands based on a user-defined commodity demand.

1.2. Novelty

We utilized time series forecasting methods to effectively predict future supply and demand of commodities in `d3ploy`. These methods are commonly used in other fields to make future predictions based on past time series data. This is a novel approach that has never been applied to NFCSims.

1.3. Objectives

The main objectives of this paper are: (1) to describe the demand driven deployment capabilities of Cyclus, (2) to describe the prediction methods available in `d3ploy`, (3) to demonstrate the use of `d3ploy` in setting up EG01-23, EG01-24, EG01-29, and EG01-30 transition scenarios with various power demand curves.

2. Methodology

In Cyclus, developers have the option to design agents using C++ or Python. The `d3ploy Institution` agent was implemented in Python to enable the use of well developed time series forecasting Python packages.

In a Cyclus NFC simulation, at every time step `d3ploy` predicts supply and demand of each commodity for the next time step. If undersupply is predicted for any commodity, `d3ploy` deploys facilities to meet its predicted demand. Figure 1 shows the logic flow of `d3ploy` at every time step.

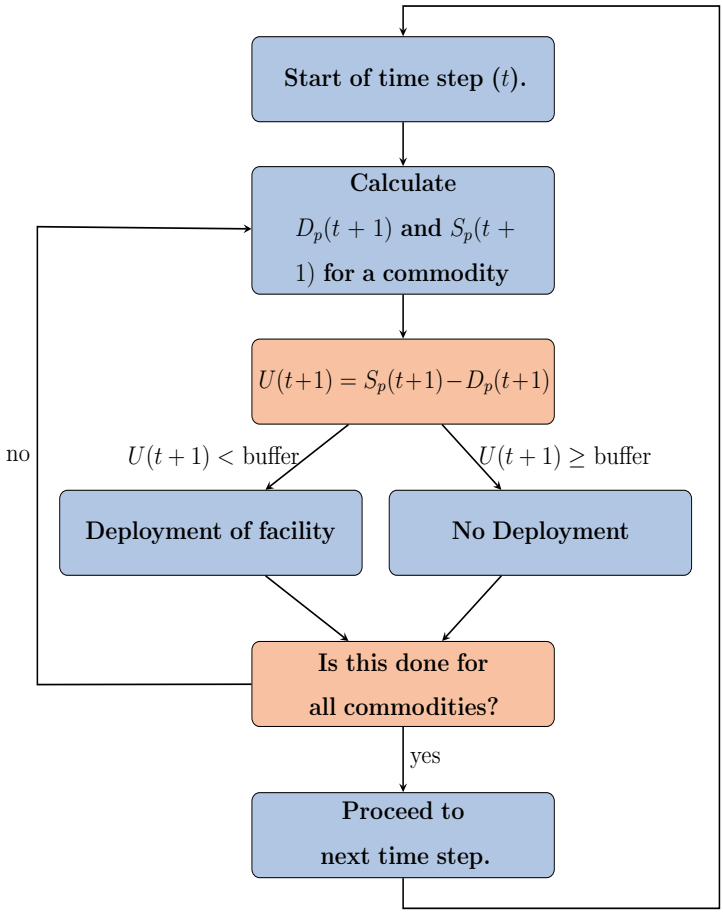


Figure 1: `d3ploy` logic flow at every time step in Cyclus [6].

`d3ploy`'s main objective is to minimize undersupply of power. The sub-objectives are : (1) to minimize the number of time steps of undersupply or under capacity of any commodity, (2) to minimize excessive oversupply of all commodities. This is a reflection of reality in which it is important to never have an undersupply of power on the grid by ensuring power plants are never short of fuel, while not having excessive oversupply resulting in a burden to store unused supplies. NFCSims often face power undersupplies at certain time steps due to lack of viable fuel, despite having sufficient installed reactor capacity. Therefore, using `d3ploy` to automatically deploy supporting facilities will prevent this from occurring.

2.1. Structure

In `d3ploy`, two different institutions control front-end and back-end fuel cycle facilities: `DemandDrivenDeploymentInst` and `SupplyDrivenDeploymentInst`, respectively. This distinction was made because front-end facilities are deployed to meet demand for commodities they produce, whereas, back-end facilities are deployed to meet supply for the commodities they provide capacity for. For example, for front-end facilities, a reactor facility demands fuel and `DemandDrivenDeploymentInst` triggers deployment of fuel fabrication facilities to create supply, and thus, meeting demand for fuel to prevent undersupply. For back-end facilities, the reactor generates spent fuel and `SupplyDrivenDeploymentInst` triggers deployment of waste storage facilities to create capacity meeting the supply of spent fuel to prevent under capacity.

2.2. Input Variables

Table 2 lists and gives examples of the input variables `d3ploy` accepts. Essentially, the user must do the following: define the facilities controlled by `d3ploy` and their respective capacities, the driving commodity and its demand equation, the deployment driving method, and the preferred prediction method. The user also has the option to define supply/capacity buffers for individual commodities, facility preferences, and facility constraints. The subsequent

	Input Parameter	Examples
Required	Demand driving commodity	Power, Fuel, Plutonium, etc.
	Demand equation	P(t) = 10000, sin(t), 10000*t
	Facilities it controls	Fuel Fab, LWR reactor, SFR reactor, Waste repository, etc.
	Capacities of the facilities	3000 kg, 1000 MW, 50000 kg
	Prediction method	Power: fast fourier transform Fuel: moving average Spent fuel: moving average
	Deployment driven by	Installed Capacity/Supply
Optional	Supply/Capacity Buffer type	Absolute
	Supply/Capacity Buffer size	Power: 3000 MW Fuel: 0 kg Spent fuel: 0 kg
	Facility preferences	LWR reactor = 100-t SFR reactor = t-100
	Facility constraint	SFR reactor constraint = 5000kg of Pu

Table 2: `d3ploy`'s required and optional input parameters with examples.

sections provide an in-depth description of the deployment driving methods, buffers, facility preferences, and prediction methods.

2.2.1. Deployment Driving Method

The user has the choice of deploying facilities based on the difference between either predicted demand and supply, or predicted demand and installed capacity. There are two advantages of using installed capacity over predicted supply. First, to prevent over-deployment of facilities with an intermittent supply; one example would be reactor facilities that have periodic downtimes for refueling. If predicted supply was selected instead of installed capacity, `d3ploy` would deploy surplus reactors during refueling downtimes to meet the temporary power undersupply. Second, to prevent infinite deployment of a facility that uses a commodity no longer available in the simulation. For example, in a transition scenario from Light Water Reactors (LWRs) to Sodium-Cooled Fast Reactors

(SFRs), the reprocessing plant that fabricates SFR fuel might demand Pu after the existing inventory is depleted and all Pu-generating LWRs have already been decommissioned. This will result in `d3ploy` deploying infinite reprocessing facilities in a futile attempt to produce SFR fuel, given the lack of Pu. This can also be avoided by using `d3ploy`'s facility constraint capability (section 2.3) to withhold SFR deployment until a sizable inventory of Pu is accumulated in the simulation.

2.2.2. Supply/Capacity Buffer

In `DemandDrivenDeploymentInst`, the user has the option to provide a supply buffer for each commodity; `d3ploy` will account for the buffer when calculating predicted demand and deploy facilities accordingly. In `SupplyDrivenDeployment Inst`, the user has the option to provide a capacity buffer for specific commodities; `d3ploy` will account for the buffer when calculating predicted supply and deploy facilities accordingly. For example, the user could set the power commodity's supply buffer to be 2000 MW. If predicted demand is 10000 MW, `d3ploy` will deploy reactor facilities to meet the predicted demand and supply buffer, resulting in a power supply of 12000 MW. The buffer can be defined as a percentage (equation 1) or absolute value (equation 2).

$$S_{pwb} = S_p * (1 + d) \tag{1}$$

$$S_{pwb} = S_p + a \tag{2}$$

where S_{pwb} is predicted supply/capacity with buffer, S_p is the predicted supply/capacity without buffer, d is the percentage value in decimal form, and a is the absolute value of the buffer.

Using a combination of this buffer capability alongside the installed capacity deployment driving method in a transition scenario simulation effectively minimizes undersupply of a commodity while avoiding excessive oversupply. This is demonstrated in section 3.1.

2.3. Facility Preference and Constraint

The user can elect to specify time-dependent preference equations for each facility; if more than one facility can supply the same commodity, `d3ploy` uses these equations to determine which facility to deploy during a commodity shortage. In table 2, the LWR reactor has a preference of $100 - t$ and the SFR reactor has a preference of $t - 100$. Thus, the simulation will have a preference to deploy LWRs before time step 100 and SFRs afterwards.

The user also has the option to provide each facility with a commodity constraint. In table 2, the SFR has a commodity constraint of 5000kg of Pu. This constrains SFR deployment to the size of the Pu inventory in the simulation. SFRs are deployed only after the 5000kg minimum Pu inventory is satisfied.

In many scenarios, advanced reactors are forced to idle after depleting the initial Pu inventory. In these situations, the facility preferences and constraint capabilities are beneficial to the user. An ideal transition year is selected using the facility preferences, however the transition will only begin when there is sufficient Pu inventory (set by facility constraint) to avoid shortages.

Therefore, when `d3ploy` predicts an undersupply of a commodity, it deploys facilities in order of preference, starting at the highest and moving down if the facility in question does not meet its constraint criteria. If the facilities do not have preferences or constraints, `d3ploy` will deploy the available facilities to minimize the number of deployed facilities while minimizing oversupply of the commodity.

2.4. Prediction Methods

`d3ploy` records supply and demand values at each time step for all commodities. This provides time series data for `d3ploy`'s time series forecasting methods to predict future supply and demand for each commodity. Three main method types were investigated: non-optimizing, deterministic-optimizing, and stochastic-optimizing time series forecasting methods. Non-optimizing methods are techniques that harness simple moving average and autoregression concepts

that use historical data to infer future supply and demand values. Deterministic-optimizing and stochastic-optimizing methods are techniques that use an assortment of more complex time series forecasting concepts to predict future supply and demand values. Deterministic-optimizing methods give deterministic solutions, while stochastic-optimizing methods give stochastic solutions.

Depending on the scenario in question, each forecasting method offers its own distinct benefits and disadvantages. The various methods are compared for each type of simulation to determine the most effective prediction method for a given scenario. The prediction methods will be described in the following sections.

2.4.1. Non-Optimizing Methods

Non-optimizing methods include: Moving Average (`MA`) , Autoregressive Moving Average (`ARMA`), and Autoregressive Heteroskedasticity (`ARCH`). The `MA` method calculates the average of a user-defined number of previous entries in a commodity's time series and returns it as the predicted value (equation 3).

$$Predicted\ Value = \frac{V_1 + V_2 + ... + V_n}{n} \tag{3}$$

The `ARMA` method combines moving average and autoregressive models (equation 4). The first term is a constant, second term is white noise, third term is the autoregressive model, and the fourth term is the moving average model. The `ARMA` method is more accurate than the `MA` method because of the inclusion of the autoregressive term.

$$X_t = c + \epsilon_t + \sum_{i=1}^{p} \varphi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} \tag{4}$$

The `ARCH` method modifies the original moving average term (described in equation 4). This modification makes the `ARCH` method better than the `ARMA` method for volatile time series data [7]. The StatsModels [8] Python package is used to implement `ARMA` and `ARCH` methods in `d3ploy`.

2.4.2. Deterministic-Optimizing Methods

Deterministic methods include: Fast Fourier Transform (`FFT`), Polynomial Fit (`poly`), Exponential Smoothing (`exp-smoothing`) , and Triple Exponential Smoothing (`holt-winters`). The `FFT` method computes the discrete Fourier transform of the time series to predict future demand and supply values (equation 5). This method is implemented in `d3ploy` using the SciPy [9] Python package.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i 2\pi k n / N} \tag{5}$$

The `poly` method models the time series data with a nth degree (user-defined) polynomial to determine future demand and supply values. This method was implemented in `d3ploy` using the NumPy [10] Python package. The `exp-smoothing` and `holt-winters` methods use a weighted average of time series data with weights decaying exponentially for older time series values [11] to create a model to determine future demand and supply values. The `exp-smoothing` method excels in modeling univariate time series data without trend or seasonality, whereas the `holt-winters` method applies exponential smoothing three times resulting in higher accuracy when modeling seasonal time series data. The StatsModels [8] Python package was used to implement both of these methods in `d3ploy`.

2.5. Stochastic-Optimizing Methods

There is one stochastic-optimizing method: step-wise seasonal method. The method was implemented in `d3ploy` by the auto Auto-Regressive Integrated Moving Averages (ARIMA) method in the pmdarima [12] Python package. The ARIMA model is a generalization of the Autoregressive Moving Average (ARMA) model to make the model fit the time series data better. It replaces the time series values with the difference between consecutive values.

	Input Parameters	Simple Transition Scenario: Linearly Increasing Power
Required	Demand driving commodity	Power
	Demand equation [MW]	$t < 40 = 1000, t \geq 40 = 250t$
	Available facilities	`Source`, `Reactor`, `Sink`
	Prediction method	`FFT`
	Deployment driving method	Installed Capacity
Optional	Buffer type	Absolute
	Buffer size	Power: 2000MW, Fuel: 1000kg

Table 3: `d3ploy`'s input parameters for the simple transition scenarios.

3. Results

To demonstrate `d3ploy`'s capability to effectively conduct transition scenario analysis and meet the objectives described in section 1.3, this section will (1) demonstrate `d3ploy`'s capability in simple transition scenarios, (2) compare the prediction methods for different transition scenarios, and (3) demonstrate using `d3ploy` to setup successful EG01-EG23, EG01-EG24, EG01-EG29, and EG01-EG30 transition scenarios. The input files and scripts to produce the results and plots in this paper can be reproduced using [13], and [14].

3.1. Demonstration of `d3ploy`'s capabilities

We conducted a simple linearly increasing power demand simulation to demonstrate `d3ploy`'s capabilities for simulating transition scenarios and to inform decisions about input parameters when setting up larger transition scenarios with many facilities. This simulation is a simple transition scenario that only includes three facility types: `source`, `reactor`, and `sink`. The simulation initially has ten initial `reactor` facilities (`reactor1` to `reactor10`). These reactors have staggered cycle lengths and lifetimes to prevent simultaneous refueling and setup gradual decommissioning. `d3ploy` is configured to deploy `new reactor` facilities to meet the loss of power supply created by the decommissioning of the initial `reactor` facilities. Table 3 shows the `d3ploy` input parameters for this simulation.

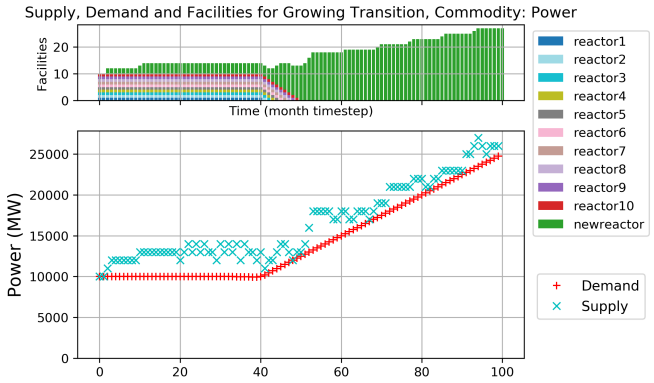
Figures 2a, 2b and 2c demonstrate `d3ploy`'s capability to deploy reactor

and supporting facilities to meet the linearly increasing power demand and subsequently demanded secondary commodities with minimal undersupply. Figure 2a demonstrates that the main objective of `d3ploy` was met as there are no time steps in which the supply of power falls under demand. By using a combination of the `FFT` method for predicting demand and setting the supply buffer to 2000MW (the capacity of 2 reactors), the user minimizes the number of undersupplied time steps for every commodity.

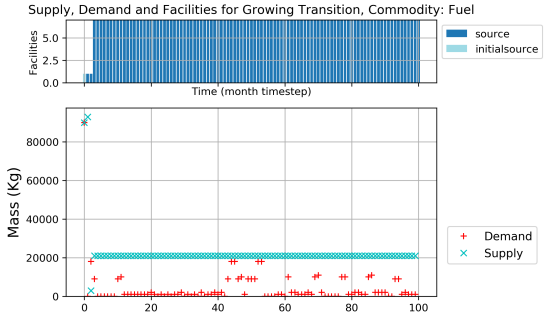
In figure 2b, a source facility with a large fuel throughput is initially deployed to meet the large initial fuel demand for the commissioning of ten reactors. By having an initial facility with a large throughput exist for the first few time steps, `d3ploy` is prevented from deploying supporting facilities that end up being redundant at the later times in the simulation. This is a reflection of reality in which reactor manufacturers will accumulate an appropriate amount of fuel inventory before starting up reactors. There is one time step in which a power undersupply exists after the decommissioning of the large initial facility; this is unavoidable as the prediction methods in `d3ploy` are unable to forsee this sudden drop in demand.

3.2. Comparison of Prediction Methods

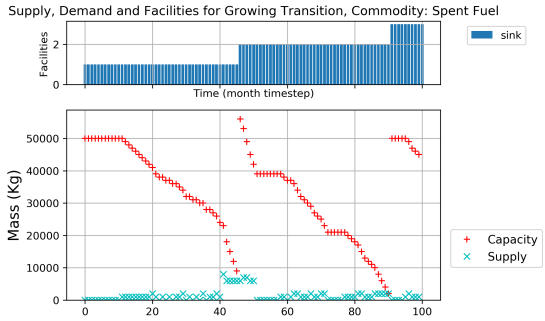
EG01-EG23, EG01-EG24, EG01-EG29, and EG01-EG30 transition scenarios are setup in Cyclus using `d3ploy`. To determine the most effective `d3ploy` prediction methods, a comparison of each prediction method for each transition scenario is conducted for both constant and linearly increasing power demand curves. Similar to the simple transition scenario, these transition scenario simulations begin with an initial fleet of LWRs that are progressively decommissioned starting at the 80 year mark, after which `d3ploy` deploys SFRs and mixed oxide (MOX) LWRs to meet the power demand. Figure 3 shows the setup of facilities and mass flows for EG01-23 and EG01-29 in Cyclus. In EG01-23 and EG01-29, only plutonium is recycled from LWR spent fuel to produce Fast Reactor (FR) fuel. EG01-24 and EG01-30 are similar to EG01-23 and EG01-29 respectively, with the exception that all transuranic elements are recycled.



(a) The power demand is a user-defined equation and power is supplied by the reactors. There are no time steps with undersupply of power.

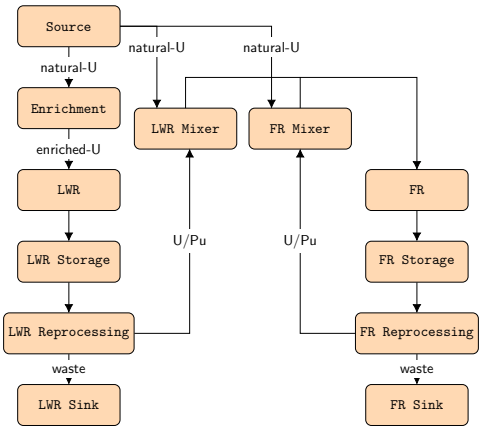


(b) Fuel is demanded by reactors and supplied by source facilities. There is only one time step with undersupply of fuel.

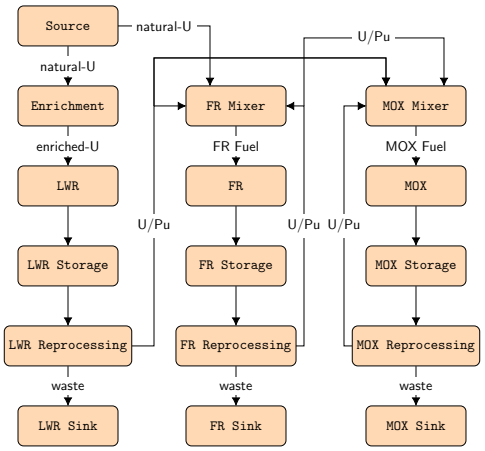


(c) Spent Fuel is supplied by reactors and the capacity is provided by sink facilities. There are no time steps with under capacity of sink space.

Figure 2: Transition Scenario: Linearly increasing power demand.



(a) EG01-EG23.



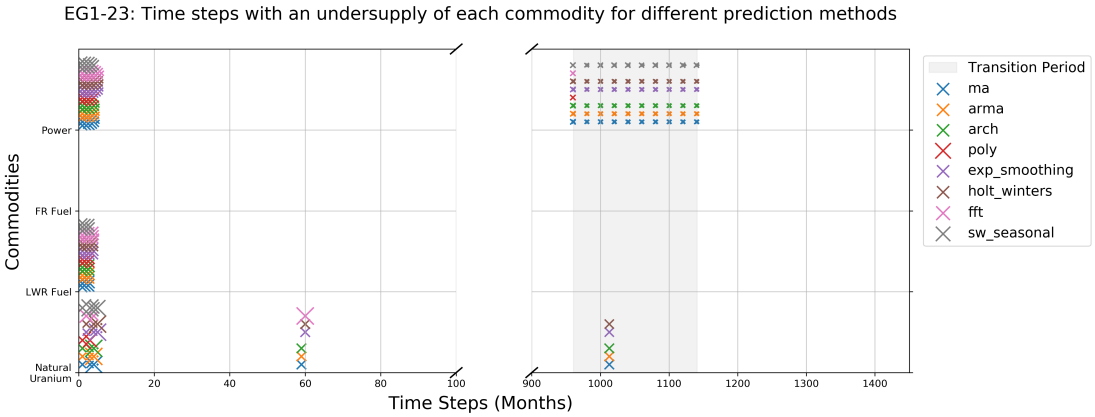
(b) EG01-EG29.

Figure 3: Diagrams with facilities and mass flow of the scenarios EG01-EG23 and EG01-EG29.

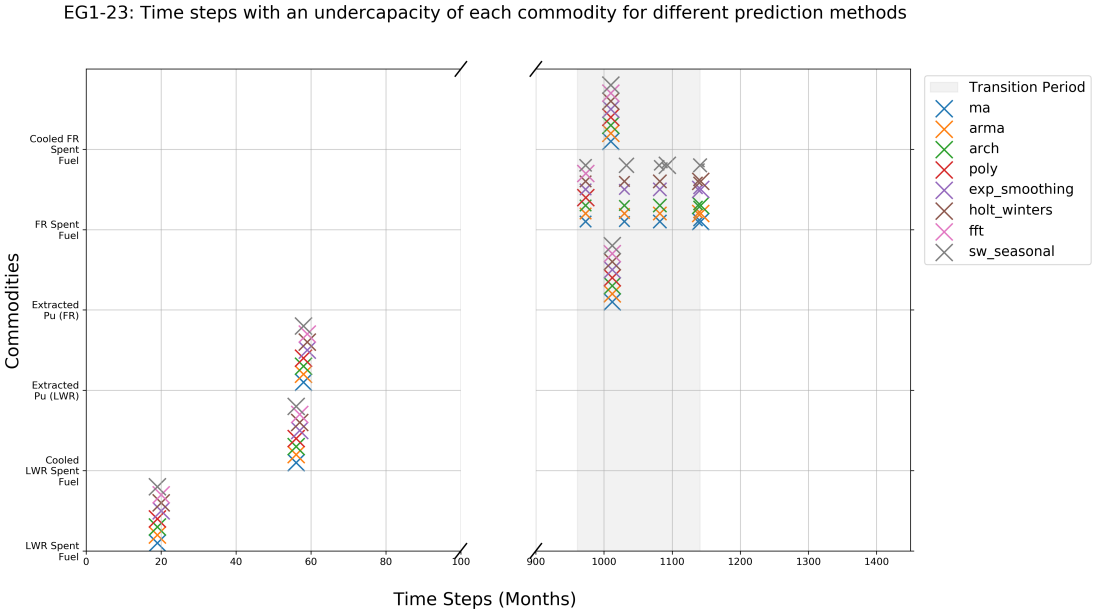
In Figure 4, crosses represent the time steps in which there is undersupply or under capacity of each commodity for a constant power EG01-23 scenario for varying prediction methods. The size of the crosses are proportional to the undersupply value, therefore, larger crosses correspond to larger undersupply. Table 7 shows the number of time steps with power undersupply for constant power EG01-EG23 and EG01-29, linearly increasing power EG01-24 and EG01-30 transition scenarios. Figure 4 demonstrates that the `poly` and `fft` methods perform the best, since they have the least number of points on the plot, indicating that they have the fewest number of time steps with undersupply and under capacity of commodities. Table 7 shows that the `poly` method performs slightly better at minimizing undersupply of power than `fft`. A similar analysis was done for a constant power EG01-29 scenario, and as seen in Table 7 the `poly` prediction method also performed best for minimizing undersupply of power.

In Figure 5, crosses represent the time steps in which there is undersupply or under capacity of each commodity for a linearly increasing power EG01-24 scenario for varying prediction methods. As with Figure 4, the size of the crosses are proportional to the undersupply value. Figure 5 demonstrates that the `fft` method performs the best at minimizing undersupply of all commodities. A similar analysis was performed for a constant power EG01-30 scenario, and as seen in Table 7 the `fft` also performed best for minimizing undersupply of power.

From Figures 4, 5, and Table 7, we can see that the `poly` method performs best for constant power transition scenarios and the `fft` method performs best for linearly increasing power transition scenarios. Undersupply and under capacity of commodities occur during two main time periods: initial demand for the commodity and during the transition period. To further `d3ploy`'s main objective of minimizing the power undersupply, sensitivity analysis of the power supply buffer for each transition scenario is conducted with best performing prediction method to find a buffer size that will minimize power undersupply.

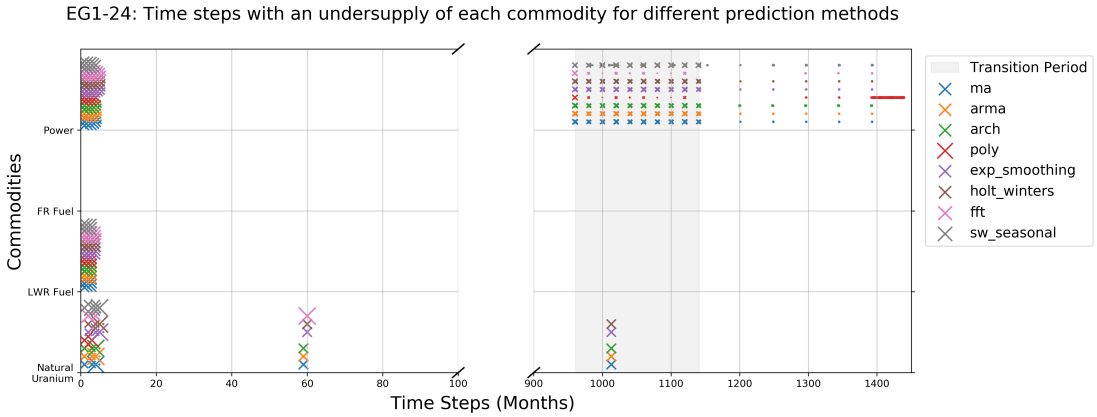


(a) Time dependent undersupply of commodities in simulation

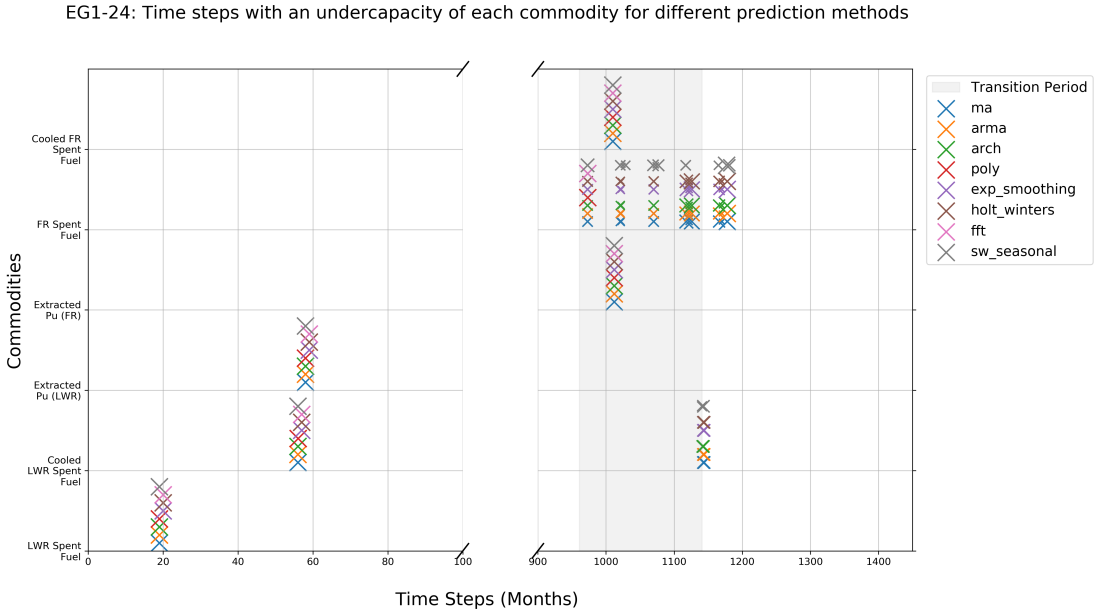


(b) Time dependent under capacity of commodities in simulation

Figure 4: EG01-23 Transition Scenario with Constant Power Demand: Each cross represent a time step in which there is undersupply or under capacity of each commodity for varying prediction methods. The size of each cross is proportional on the size of the undersupply.



(a) Time dependent undersupply of commodities in simulation



(b) Time dependent under capacity of commodities in simulation

Figure 5: EG01-24 Transition Scenario with Linearly Increasing Power Demand: Each cross represent a time step in which there is undersupply or under capacity of each commodity for varying prediction methods. The size of each cross is proportional on the size of the undersupply.

	Power Undersupplied Time Steps			
Algorithm	EG01-EG23 Constant Power	EG01-EG24 Linearly Increasing Power	EG01-EG29 Constant Power	EG01-EG30 Linearly Increasing Power
MA	26	36	15	24
ARMA	26	36	15	24
ARCH	26	36	15	21
POLY	6	65	4	9
EXP_SMOOTHING	27	37	16	25
HOLT-WINTERS	27	37	16	25
FFT	8	20	5	9
SW_SEASONAL	36	107	14	51

Table 4: Undersupply and oversupply of power with the different algorithms used to drive EG01-EG23,24,29,30.

3.3. Sensitivity Analysis

Sensitivity analysis of the power buffer size was conducted for EG01-EG23, EG01-24, EG01-29, and EG01-30 transition scenarios. Varying the power buffer size does not impact the number of undersupply time steps for EG01-EG23 and EG01-EG29 constant power demand transition scenario with the `poly` prediction method. There are 6 and 4 time steps (table 7) in which there is power undersupply for EG01-EG23 and EG01-29 transition scenarios respectively. As seen from figure 4, these undersupply time steps occur at the beginning of the simulation and for one time step when the transition begins. This is expected since without time series data at the beginning of the simulation, `d3ploy` takes a few time steps to collect time series data about power demand to predict and start deploying reactor and supporting fuel cycle facilities. When the transition begins, power is under supplied for one time step, ; following this, `d3ploy` accounts for the undersupply by deploying facilities to meet power demand. Therefore, the power undersupply is minimized for constant power EG01-EG23 and EG01-EG29 transition scenarios with a 0MW power supply buffer.

For EG01-EG24 and EG01-30 linearly increasing power demand transition

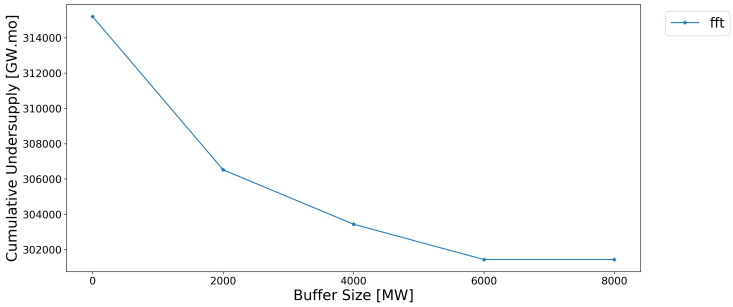
scenarios, power buffer size is varied. Figures 6a, 6c and Table 5 show that with an increasing buffer size, the number of power undersupply time steps decreases. For EG01-24, it plateaus at 6000MW, and for EG01-30, the cumulative undersupply is smallest for a buffer size of 8000MW. As seen from Figures 6b and 6d, these undersupply time steps occur at the beginning of the simulation and for one time step when the transition begins. This is expected since without time series data at the beginning of the simulation, `d3ploy` takes a few time steps to collect time series data about power demand to predict and start deploying reactor and supporting fuel cycle facilities. Therefore, a buffer of 6000MW and 8000MW minimizes the power undersupply for EG01-Eg24 and EG01-EG30 respectively.

Buffer [MW]	Undersupply	EG01-24	EG01-30
0	Time steps [#]	20	9
	Energy [$GW \cdot mo$]	315791	152517
2000	Undersupplied [#]	9	6
	Energy [$GW \cdot mo$]	306520	147166
4000	Time steps [#]	8	6
	Energy [$GW \cdot mo$]	303438	143166
6000	Time steps [#]	7	5
	Cumulative [GW]	303438	139083
8000	Time steps [#]	7	5
	Energy [$GW \cdot mo$]	303438	135083

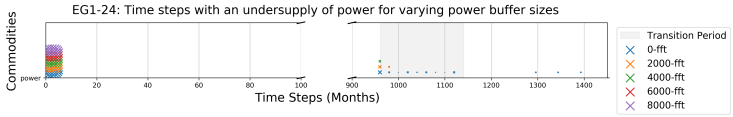
Table 5: Dependency of the undersupply of Power on the buffer size for EG01-EG24 and EG01-EG30 transition scenarios with linearly increasing power demand using the fft prediction method.

3.4. Best Performance Models

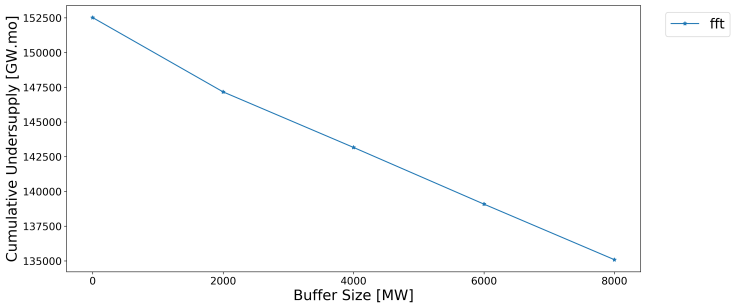
Table 6 shows `d3ploy` input parameters for EG01-EG23, EG01-EG24, EG01-EG29, and EG01-EG30 transition scenarios that minimize undersupply of power and minimize the undersupply and under capacity of the other commodities in the simulation. The need for buffers for commodities is a reflection of reality in which a supply buffer is usually maintained to ensure continuity in the event of an unexpected failure in the supply chain.



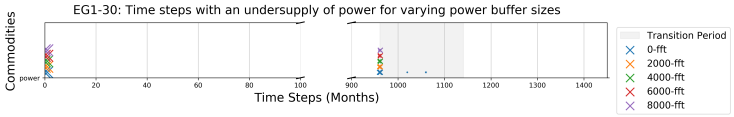
(a) EG01-24: Power buffer size vs. cumulative undersupply



(b) EG01-24: Time-dependent undersupply of power for varying power buffer sizes



(c) EG01-30: Power buffer size vs. cumulative undersupply



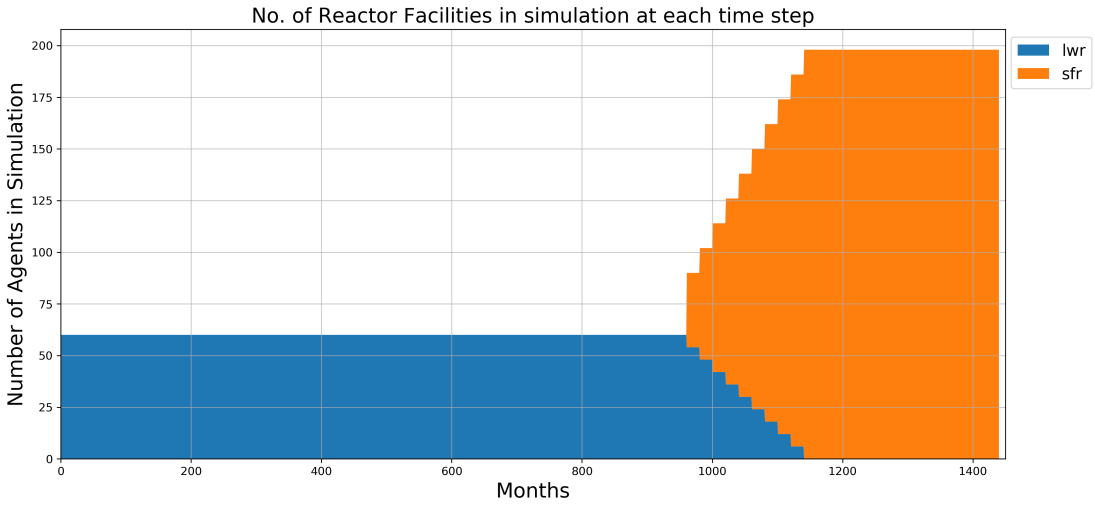
(d) EG01-30: Time-dependent undersupply of power for varying power buffer sizes

Figure 6: Sensitivity Analysis of Power buffer size on cumulative undersupply of Power for EG01-EG24 and EG01-EG30 transition scenarios with linearly increasing power demand using the fft prediction method.

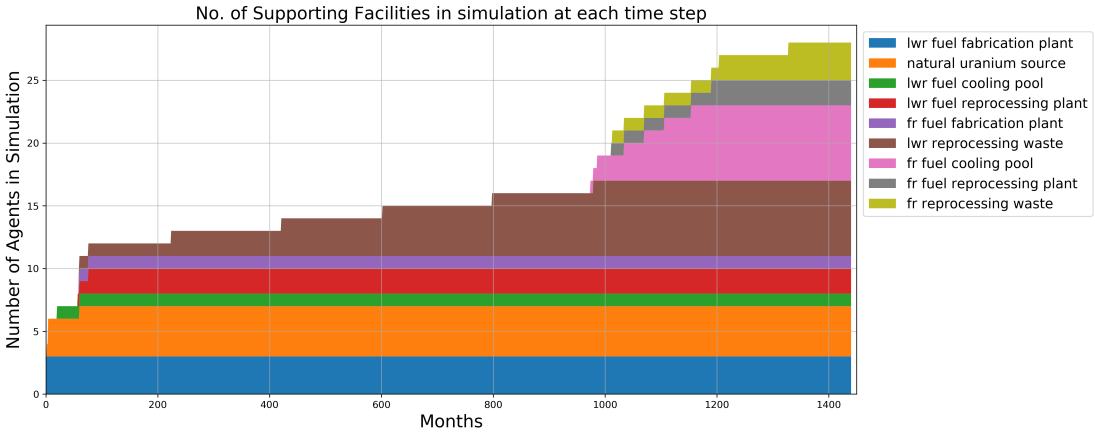
	Input Parameter	Simulation Description			
		EG01-23	**EG01-24**	**EG01-29**	**EG01-30**
Required	Demand driving commodity	Power			
	Demand equation [MW]	60000	$60000 + 250t/12$	60000	$60000 + 250t/12$
	Prediction method	`poly`	`fft`	`poly`	`fft`
	Deployment Driving Method	Installed Capacity			
Optional	Buffer type	Absolute			
	Power Buffer size [MW]	0	6000	0	8000

Table 6: `d3ploy`'s input parameters for EG01-EG23, EG01-EG24, EG01-EG29, and EG01-EG30 transition scenarios that minimizes undersupply of power and minimizes the undersupply and under capacity of the other facilities.

Figure 7 and 8 show time dependent deployment of reactor and supporting facilities for the EG01-23 constant power demand and EG01-30 linearly increasing power demand transition scenarios, respectively. `d3ploy` automatically deploys reactor and supporting facilities to setup a supply chain to meet power demand during a transition from LWRs to SFRs for EG01-23, and from LWRs to MOX LWRs and SFRs for EG01-30. EG01-24 and EG01-29 facility deployment plots are very similar to EG01-23 and EG01-30, respectively, therefore they are not shown.

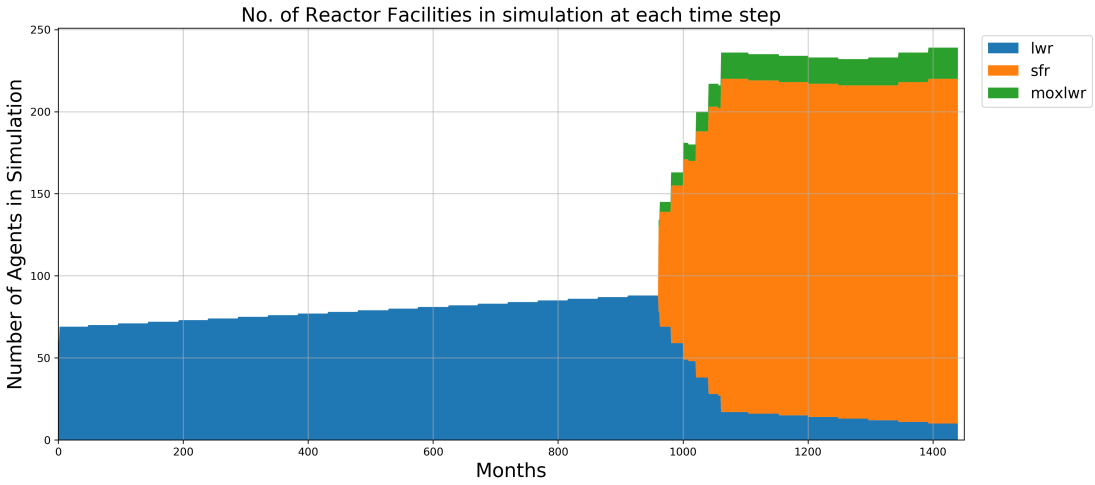


(a) EG01-23: Reactor Deployment

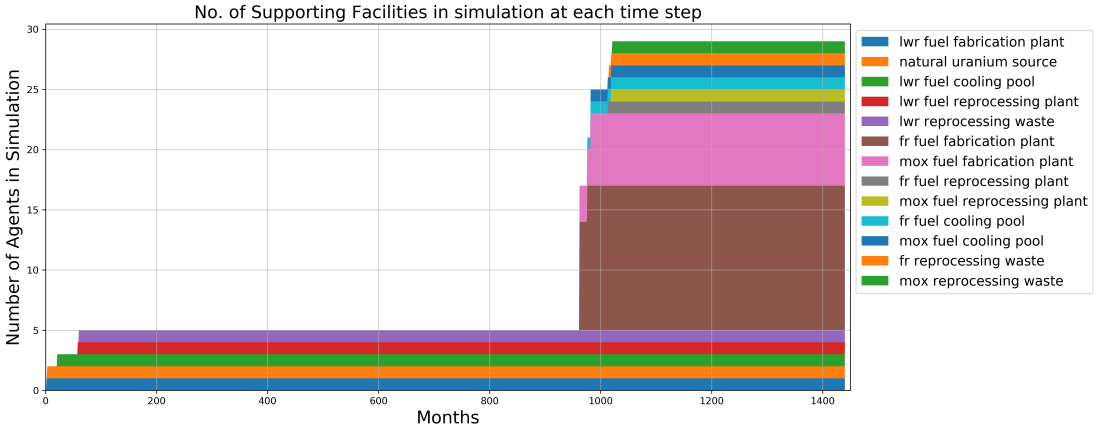


(b) EG01-23: Supporting Facility Deployment

Figure 7: Time dependent deployment of reactor and supporting facilities in the EG01-23 constant power demand transition scenario. `d3ploy` automatically deploys reactor and supporting facilities to setup a supply chain to meet constant power demand of 60000 MW during a transition from LWRs to SFRs.



(a) EG01-30: Reactor Deployment



(b) EG01-30: Supporting Facility Deployment

Figure 8: Time dependent deployment of reactor and supporting facilities in the EG01-30 linearly increasing power demand transition scenario. `d3ploy` automatically deploys reactor and supporting facilities to setup a supply chain to meet constant power demand of $60000 + 250t/12$ MW during a transition from LWRs to MOX LWRs and SFRs.

	Undersupplied Time Steps			
Transition Scenario	EG01-EG23	EG01-EG24	EG01-EG29	EG01-EG30
Power Demand	Constant	Linearly Increasing	Constant	Linearly Increasing
Prediction Method	`poly`	`fft`	`poly`	`fft`
Power Supply Buffer [MW]	0	6000	0	8000
Commodities				
Natural Uranium	2	3	1	1
LWR Fuel	4	6	1	2
SFR Fuel	0	0	2	2
MOX LWR Fuel	-	-	2	2
Power	6	7	4	5
LWR Spent Fuel	1	1	1	1
SFR Spent Fuel	1	1	1	1
MOX LWR Spent Fuel	-	-	1	1

Table 7: Undersupply/capacity of commodities for the best performing EG01-EG23,24,29,30 transition scenarios.

4. Conclusion

In this paper, we demonstrate that by carefully selecting `d3ploy` parameters, we are able to effectively automate the setup of constant and linearly increasing power demand transition scenarios for EG01-23, EG01-24, EG01-29, and EG01-30 with minimal power undersupply. Using `d3ploy` to set up transition scenarios is more efficient than the previous efforts that required a user to manually calculate and use trial and error to set up the deployment scheme for the supporting fuel cycle facilities. Transition scenario simulations set up this way are sensitive to changes in the input parameters resulting in an arduous setup process, since a slight change in one of the input parameters would result in the need to recalculate the deployment scheme to ensure that there is no undersupply of power. Therefore, by automating this process, the user can vary input parameters in the simulation and `d3ploy` will automatically adjust the deployment scheme to meet the new constraints.

5. Future Work

To enable NFCSims to produce insightful and flexible results to inform policy decisions, it is necessary to quantify and include all the subtleties of each segment of the NFC through system analysis and sensitivity studies [15]. We simulate transition scenarios to predict the future, however when implemented in the real world, it will deviate from the optimal scenario. Previously it was difficult to conduct this analysis with Cyclus as the user would have to manually calculate the deployment scheme for every change in input parameter. Therefore, using the `d3ploy` capability, sensitivity analysis studies can be more easily conducted to determine how variation in different input parameters will impact the progression and final state of a transition scenario.

6. Acknowledgments

This research is funded by the Department of Energy (DOE) Office of Nuclear Energy's Nuclear Energy University Program (Project 16-10512, DE-NE0008567) "Demand-Driven Cycamore Archetypes". The authors want to thank members of the Advanced Reactors and Fuel Cycles (ARFC) group at the University of Illinois at Urbana-Champaign. We also thank our colleagues from the Cyclus community, particularly those in the University of Wisconsin Computational Nuclear Engineering Research Group (CNERG) and the University of South Carolina Energy Research Group (ERGS) for collaborative Cyclus development.

References

[1] K. D. Huff, J. W. Bae, K. A. Mummah, R. R. Flanagan, A. M. Scopatz, Current Status of Predictive Transition Capability in Fuel Cycle Simulation, in: Proceedings of Global 2017, American Nuclear Society, Seoul, South Korea, 2017, p. 11.

[2] K. D. Huff, M. J. Gidden, R. W. Carlsen, R. R. Flanagan, M. B. McGarry, A. C. Opotowsky, E. A. Schneider, A. M. Scopatz, P. P. H. Wilson,

Fundamental concepts in the Cyclus nuclear fuel cycle simulation framework, Advances in Engineering Software 94 (2016) 46–59, arXiv: 1509.03604. doi:10.1016/j.advengsoft.2016.01.014.
URL http://www.sciencedirect.com/science/article/pii/S0965997816300229

[3] R. W. Carlsen, M. Gidden, K. Huff, A. C. Opotowsky, O. Rakhimov, A. M. Scopatz, P. Wilson, Cycamore v1.0.0, Figshare-Http://figshare.com/articles/Cycamore_v1_0_0/1041829. doi:http://figshare.com/articles/Cycamore_v1_0_0/1041829.
URL http://figshare.com/articles/Cycamore_v1_0_0/1041829

[4] R. Wigeland, T. Taiwo, H. Ludewig, M. Todosow, W. Halsey, J. Gehin, R. Jubin, J. Buelt, S. Stockinger, K. Jenni, B. Oakley, Nuclear Fuel Cycle Evaluation and Screening - Final Report, US Department of Energy (2014) 51.
URL https://fuelcycleevaluation.inl.gov/Shared%20Documents/ES%20Main%20Report.pdf

[5] B. Feng, B. Dixon, E. Sunny, A. Cuadra, J. Jacobson, N. R. Brown, J. Powers, A. Worrall, S. Passerini, R. Gregg, Standardized verification of fuel cycle modeling, Annals of Nuclear Energy 94 (2016) 300–312. doi:10.1016/j.anucene.2016.03.002.
URL http://www.sciencedirect.com/science/article/pii/S0306454916301098

[6] G. Chee, J. W. Bae, K. D. Huff, R. R. Flanagan, R. Fairhurst, Demonstration of Demand-Driven Deployment Capabilities in Cyclus, in: Proceedings of Global/Top Fuel 2019, American Nuclear Society, Seattle, WA, United States, 2019.

[7] R. R. Flanagan, J. W. Bae, K. D. Huff, G. J. Chee, R. Fairhurst, Methods for Automated Fuel Cycle Facility Deployment, in: Proceedings of Global/Top Fuel 2019, American Nuclear Society, Seattle, WA, United States, 2019.

[8] GitHub Community, StatsModels: Statistics in Python Package (2019).
URL https://www.statsmodels.org/stable/faq.html

[9] E. Jones, T. Oliphant, P. Peterson, SciPy: Open source scientific tools for Python, 2001, 2016.

[10] N. Developers, NumPy, NumPy Numpy. Scipy Developers.

[11] R. J. Hyndman, G. Athanasopoulos, Forecasting: principles and practice, OTexts, 2018.

[12] pmdarima: ARIMA estimators for Python (2019).
URL https://www.alkaline-ml.com/pmdarima/

[13] arfc/d3ploy: A collection of Cyclus manager archetypes for demand driven deployment, 10.5281/zenodo.3464123 (Sep. 2019).
URL https://github.com/arfc/d3ploy

[14] G. Chee, G. T. Park, K. Huff, arfc/transition-scenarios : Validation of Spent Nuclear Fuel Output by Cyclus, a Fuel Cycle Simulator Code (Aug. 2018). doi:10.5281/zenodo.1401495.

[15] S. Passerini, M. S. Kazimi, E. Shwageraus, A Systematic Approach to Nuclear Fuel Cycle Analysis and Optimization, Nuclear Science and Engineering 178 (2) (2014) 186–201. doi:10.13182/NSE13-20.
URL https://doi.org/10.13182/NSE13-20