Demand Driven Deployment Capabilities in Cyclus, a Fuel Cycle Simulator

Gwendolyn J. Chee[a,*], Roberto E. Fairhurst Agosta[a], Jin Whan Bae[b], Robert R. Flanagan[c], Anthony M. Scopatz[d], 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*
[d] *Quansight, LLC*

Abstract

The present United States' nuclear fuel cycle faces challenges that hinder the expansion of nuclear energy technology. The U.S. Department of Energy identified four nuclear fuel cycle options, which make nuclear energy technology more desirable. Successfully analyzing the transitions from the current fuel cycle to these promising fuel cycles requires a nuclear fuel cycle simulator that can predictively and automatically deploy fuel cycle facilities to meet user-defined power demand. This work introduces and demonstrates demand-driven deployment capabilities in Cyclus, a open-source nuclear fuel cycle simulator framework. User-controlled capabilities such as time series forecasting algorithms, supply buffers, and facility preferences were introduced to give users tools to minimize power undersupply in a transition scenario simulation. We demonstrate `d3ploy`'s capability to predict future commodities' supply and demand, and automatically deploy fuel cycle facilities to meet the predicted demand in four transition scenarios. 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.

Keywords: nuclear engineering, nuclear fuel cycle, nuclear fuel cycle simulator,

---

*Corresponding Author
Email address: `gchee2@illinois.edu` (Gwendolyn J. Chee)

time series forecasting, automated deployment

1. Introduction

The nuclear fuel cycle represents the nuclear fuel life cycle from initial extraction through processing, use in reactors, and, eventually, final disposal. This complex system of facilities and mass flows collectively provide nuclear energy in the form of electricity [1]. Nuclear fuel cycle simulator tools were introduced to investigate nuclear fuel cycle dynamics at a local and global level. These simulators track the flow of materials through the nuclear fuel cycle, from enrichment to final disposal of the fuel, while also accounting for decay and transmutation of isotopes. The impacts are evaluated in the form of 'metrics', quantitative measures of performance [2]. These metrics are calculated from mass balances and facility operation histories calculated by a fuel cycle simulator [2]. By evaluating performance metrics of different fuel cycles, we gain an understanding of how each facility's parameters and technology choices impact the system's performance. Therefore, these results can be used to guide research efforts, advise future design choices, and provide decision-makers with a transparent tool for evaluating Fuel Cycle Options to inform policy decisions [1].

Many fuel cycle simulators automatically deploy reactor facilities to meet a user-defined power demand. However, the user must define a deployment scheme of supporting facilities to avoid gaps in the supply chain resulting in idle reactor capacity. Current simulators require the user to set infinite capacity for supporting facilities but this inaccurately represents reality and obfuscates required capacities. Manually determining a deployment scheme for a once-through fuel cycle is straightforward, however, for complex fuel cycle scenarios, it is not. To ease setting up realistic nuclear fuel cycle simulations, a nuclear fuel cycle simulator must bring dynamic demand-responsive deployment decisions into the simulation logic [3]. This means the nuclear fuel cycle simulator decides how many mines, mills, enrichment facilities, reprocessing facilities, etc are deployed to support dynamically changing power demand and reactor types. Thus, a next-generation nuclear fuel cycle simulator must predictively and automatically deploy fuel cycle facilities to meet a user-defined power demand.

In Cyclus, an agent-based nuclear fuel cycle simulation framework [2], each entity (i.e. `Region`, `Institution`, or `Facility`) in the fuel cycle is an agent. `Region` agents represent geographical or political areas in which `Institution` and `Facility` agents reside. `Institution` agents represent legal operating organizations such as utilities, governments, and control the deployment and decommissioning of `Facility` agents [2]. `Facility` agents represent nuclear fuel cycle facilities such as mines, conversion facilities, reactors, reprocessing facilities, etc. Cycamore [4] provides basic `Region`, `Institution`, and `Facility` archetypes compatible with Cyclus.

1.1. Context of Work

The impact of climate change on natural and human systems is increasingly apparent [5]. The production and use of energy contribute to two-thirds of the total Green House Gas (GHG) emissions [5]. Furthermore, as the human population increases and previously under-developed nations rapidly industrialize, global energy demand is forecasted to increase. Energy generation technology selection profoundly impacts climate change via growing energy demand. Large scale deployment of emissions free nuclear power plants could significantly reduce GHG production [5].

However, large scale nuclear power deployment faces challenges of cost, safety, and used nuclear fuel [6]. Nuclear power has high capital costs, an unresolved long-term nuclear waste management strategy and perceived adverse safety, environmental, and health effects [6]. The nuclear power industry must overcome these challenges to ensure continued global use and expansion of nuclear energy technology.

The challenges described above are associated with the present once-through fuel cycle in the United States (US), in which fabricated nuclear fuel is used once and placed into storage to await disposal. An evaluation and screening study of a comprehensive set of nuclear fuel cycle options [7] was conducted to assess for

Fuel Cycle	**Open or Closed**	**Fuel Type**	**Reactor Type**
EG01 (current)	Open	Enriched-U	Thermal
EG23	Closed	Recycled U/Pu + Natural-U	Fast
EG24	Closed	Recycled U/TRU + Natural-U	Fast
EG29	Closed	Recycled U/Pu + Natural-U	Fast & Thermal
EG30	Closed	Recycled U/TRU + Natural-U	Fast & Thermal

Table 1: Descriptions of the current and other high performing nuclear fuel cycle evaluation groups described in the evaluation and screening study [7].

promising Evaluation Groups (EGs) with performance improvements compared with the existing once-through fuel cycle (EG01) in the 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 describes these fuel cycles: EG23, EG24, EG29, and EG30.

The evaluation and screening study assumed the nuclear energy systems were at equilibrium to understand the end-state benefits of each evaluation group [8]. In the current work, our goal is to model the transition from the initial EG01 state to these promising future end-states without assuming equilibrium fuel cycles. To successfully analyze time-dependent transition scenarios, the nuclear fuel cycle simulator tool must automate the transition scenario simulation setup. 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 user-defined power demand.

1.2. Novelty

We utilized time series forecasting methods to effectively predict future commodities' supply and demand in `d3ploy`. Solar and wind power generation commonly use these methods to make future predictions based on past time series data [9, 10, 11, 12]. Industrial supply chain management also uses sophisticated time series forecasting techniques to predict demand for quantities of goods in the supply chain [13]. This is a novel approach that has never been applied to nuclear fuel cycle simulators.

1.3. Objectives

The main objectives of this paper are: (1) to describe the demand-driven deployment capabilities in CYCLUS, (2) to describe the prediction methods available in `d3ploy`, and (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 simulation, at every time step, `d3ploy` predicts the supply and demand of each commodity for the next time step. Commodities refer to materials in the nuclear fuel cycle such as reactor fuel. Upon undersupply for any commodity, `d3ploy` deploys facilities to meet its predicted demand. Therefore, if the simulation begins with user-defined power demand, `d3ploy` deploys reactors to meet power demand, followed by enrichment facilities to meet fuel demand, and so on, to create the supply chain. Based on the demand and supply trends of each commodity, `d3ploy` predicts their future demand and supply, and deploys facilities accordingly to meet the future demand to prevent demand from surpassing supply. Figure 1 shows the logical flow of `d3ploy` at every time step. In subsequent subsections, we describe how to set up a transition scenario using `d3ploy` and the input parameters `d3ploy` accepts.

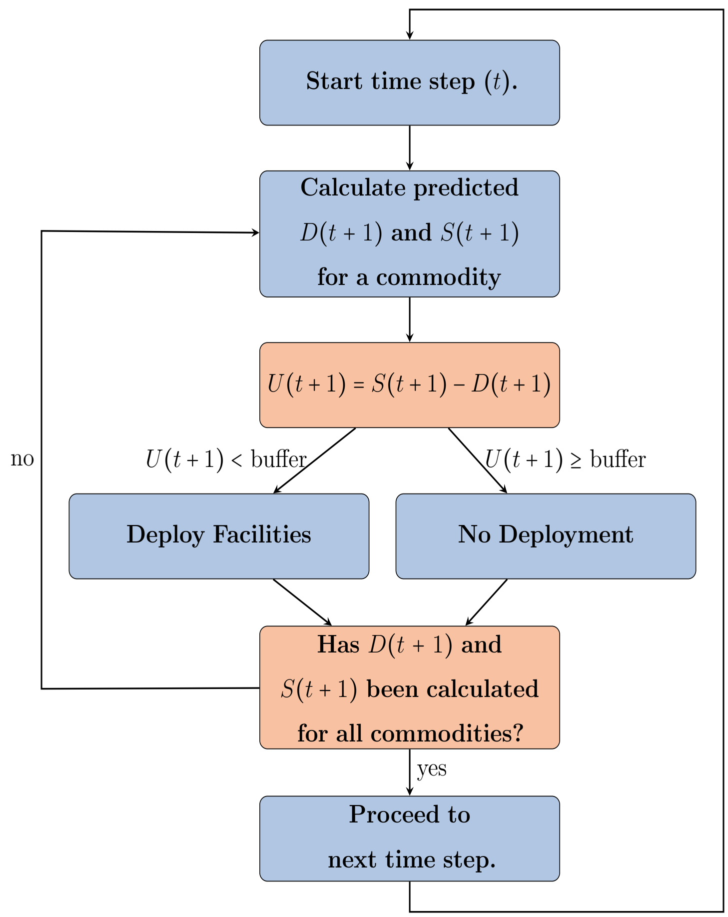


Figure 1: `d3ploy` logic flow at every time step in CYCLUS [14].

`d3ploy` aims to minimize the undersupply of power:

$$\text{obj} = \min \sum_{t=1}^{t_f} |D_{t,p} - S_{t,p}|. \tag{1}$$

where:

$$\begin{aligned} t_f &= \text{Number of time steps [months]} \\ t &= \text{time [month]} \\ D &= \text{Demand} \\ S &= \text{Supply} \\ p &= \text{power [MW]} \end{aligned}$$

The sub-objectives are to minimize the number of time steps of undersupply or under-capacity of any commodity:

$$obj = min \sum_{c=1}^{M} \sum_{t=1}^{t_f} |D_{t,c} - S_{t,c}|, \tag{2}$$

and to minimize excessive oversupply of all commodities:

$$obj = min \sum_{c=1}^{M} \sum_{t=1}^{t_f} |S_{t,c} - D_{t,c}|. \tag{3}$$

where:

$$\begin{aligned} c &= \text{commodity type} \\ M &= \text{Number of commmodities} \end{aligned}$$

Minimizing excessive oversupply reflects reality, in which utilities ensure grid availability by ensuring power plants are never short of fuel while avoiding expensive storage of excess fuel. Nuclear fuel cycle simulations often face power shortages due to lack of viable fuel, despite having sufficient installed reactor capacity. Using `d3ploy` to automate the deployment of supporting facilities prevents this.

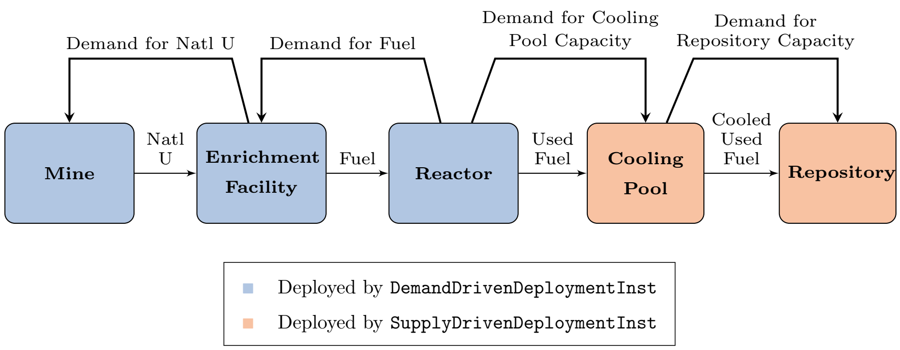


Figure 2: Simple once-through fuel cycle depicting which facilities are deployed by `DemandDrivenDeploymentInst` and `SupplyDrivenDeploymentInst`.

2.1. Structure

Front-end facilities meet the demand for commodities they produce, whereas back-end facilities meet supply for the commodities they demand. Therefore, in `d3ploy` two distinct institutions control front-end and back-end fuel cycle facilities: `DemandDrivenDeploymentInst` and `SupplyDrivenDeploymentInst`, respectively. For example, when a reactor facility demands fuel, `DemandDriven-DeploymentInst` deploys fuel fabrication facilities to create fuel supply. For back-end facilities, the reactor generates spent fuel, and `SupplyDrivenDeploymentInst` deploys used fuel storage facilities to create capacity to store the spent fuel. Figure 2 depicts a simple once-through fuel cycle and the `Institution` type governing each facility's deployment.

2.1.1. Deployment-Driving Method

To prevent over-deployment of facilities with an intermittent supply such as reactors that require refueling, and to prevent infinite deployment of a facility that demands a commodity no longer available in the simulation, we introduced the capability to deploy facilities based on the difference between predicted demand and installed capacity. The user may deploy facilities based on the difference between predicted demand and predicted supply, *or* predicted demand and installed capacity. For example, a reprocessing plant that fabricates Sodium-

Cooled Fast Reactor (SFR) fuel demands for Pu after depletion of the existing Pu inventory and decommissioning of the LWRs that produce it. If we used the deployment-driving method driven by the difference in predicted demand and predicted supply, this results in infinite deployment of reprocessing facilities in a futile attempt to produce SFR fuel, crashing the simulation. Instead, if we use the deployment-driving method driven by the difference in predicted demand and installed capacity, only one reprocessing facility will be deployed, the simulation will finish, and the user will see that a large Pu inventory must be accumulated. Therefore, using the deployment-driving method that deploys facilities based on the difference between predicted demand and installed capacity is ideal for most transition scenarios.

2.2. Input Variables

Table 2 lists and gives examples of the input variables `d3ploy` accepts. The user must define the following input variables: (1) **The available facilities for `d3ploy` to deploy in the simulation and their respective capacities.** Users must define the facilities they want `d3ploy` to deploy. It is the user's responsibility to ensure the defined facilities create a supply chain to produce the demand driving commodity. (2) **The demand driving commodity and its demand equation.** For most simulations, the demand driving commodity is power. The demand equation is defined by a mathematical equation with units of MW. For example, a constant power demand equation is 10000, while a linearly increasing power demand equation is $100t$. (3) **The deployment driving method**. This input variable is described in Section 2.1.1. (4) **The prediction method**. This input variable is described in Section 2.4. There are also optional input variables: (5) **Supply/capacity buffers for individual commodities.** This input variable is described in section 2.2.1. (6) **Facility preferences.** This input variable is described in section 2.3. (7) **Facility fleet shares.** This input variable is described in section 2.3.

	Input Parameter	Examples
Required	Demand driving commodity	Power
	Demand equation [MW]	P(t) = $10000, sin(t), 10000t$
	Available Facilities	Mine, LWR, 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
Optional	Supply/Capacity Buffer type	Absolute
	Supply/Capacity Buffer size	Power: 3000 MW Fuel: 0 kg Spent fuel: 0 kg
	Facility preferences [month]	LWR = 100-t SFR = t-99
	Fleet share percentage [%]	MOX LWR = 85% SFR = 15%

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

2.2.1. Supply/Capacity Buffer

The user has the option to specify a supply buffer for each commodity; `d3ploy` accounts for the buffer when calculating predicted demand and deploys facilities accordingly. The buffer is defined as a percentage:

$$S_{pwb} = S_p(1 + d) \tag{4}$$

or an absolute value:

$$S_{pwb} = S_p + b \tag{5}$$

where:

$$\begin{aligned} S_{pwb} &= \text{predicted supply/capacity with buffer} \\ S_p &= \text{predicted supply/capacity} \\ d &= \text{buffer's percentage value in decimal form} \\ b &= \text{buffer's absolute value} \end{aligned}$$

Using the buffer capability and installed capacity to drive facility deployment in a transition scenario simulation will effectively minimize undersupply of a commodity while avoiding excessive oversupply. This is demonstrated in Section 3.1.

2.3. Facility Preference and Fleet Share

The user can define time-dependent preference equations to facilities' that supply the same commodity. If there are two reactor types, Light Water Reactors (LWRs) and Sodium-Cooled Fast Reactors (SFRs), in a simulation, the user can make use of time-dependent preferences to make the simulation deploy LWRs at earlier times in the simulation, and deploy SFRs at later times in the simulation when there is a power demand. In Table 2, the user defined that the LWR has a preference of $100 - t$, while the SFR has a preference of $t - 99$. Figure 3 depicts how the preference for each reactor changes with time. When there is a power undersupply, `d3ploy` will deploy the reactor that has a larger preference at that time step. At time step 100, LWR preference is 0, while SFR preference is 1; therefore an SFR is deployed if there is a power shortage. Thus, the transition occurs at the 100^{th} time step.

The user also has the option to specify percentage-share for facilities that provide the same commodity. For example, if there are two reactor types, mixed oxide (MOX) LWRs and SFRs, in a simulation, the user can make use of percentage-share specifications to determine the percentage of power supplied by each reactor. When MOX LWR has a share of $s\%$ and SFR has a share of $(100 - s)\%$, MOX LWR deployment constrains to $s\%$ of total power demand and SFR deployment constrains to $(100 - s)\%$ of total power demand.

The transition year is selected by customizing facility preferences to prefer advanced reactors at that year. The fleet-share percentage determines the share of each type of reactor to transition to. Figure 4 shows the logical flow of which facility `d3ploy` deploys when there are multiple facilities offering the same commodity.

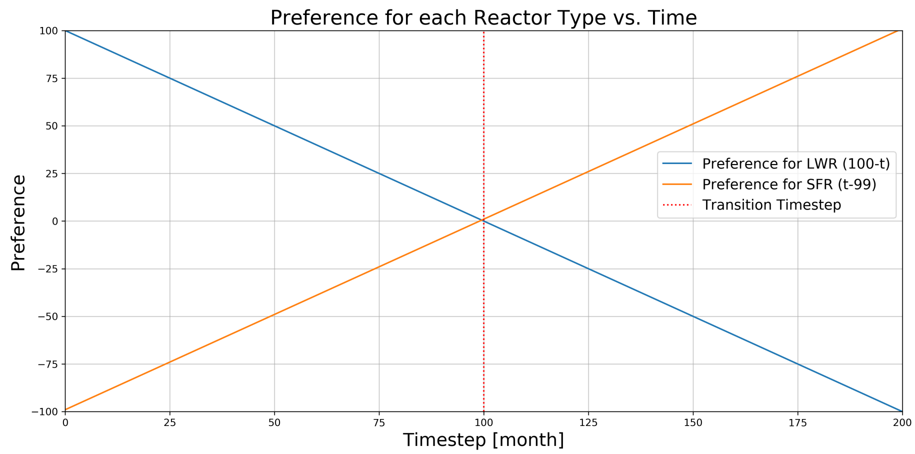


Figure 3: `d3ploy` has a $100 - t$ preference for LWRs and a $t - 99$ preference for SFRs. When there is a power undersupply, `d3ploy` will deploy the reactor that has a larger preference at that time step.

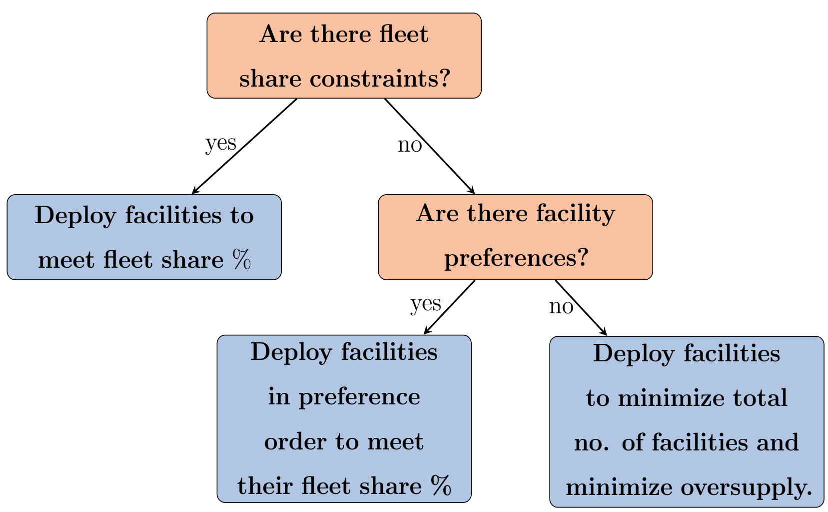


Figure 4: Logical flow of how `d3ploy` selects which facility to deploy when there are multiple facilities offering the same commodity.

2.4. Prediction Methods

`d3ploy` records supply and demand at each time step for all commodities. Time-series data informs `d3ploy`'s time series forecasting methods which predict future supply and demand for each commodity. The time series forecasting methods investigated include non-optimizing, deterministic-optimizing, and stochastic-optimizing methods. Non-optimizing methods are techniques that harness simple moving average and autoregression concepts which use historical data to infer future supply and demand values. Deterministic-optimizing and stochastic-optimizing methods are techniques that use an assortment of more sophisticated 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 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 following sections describe the prediction methods.

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 6).

$$Predicted\ Value = \frac{\sum_{n=1}^{N} V_n}{n} \tag{6}$$

where:

$$\begin{aligned} V_n &= \text{time series value} \\ N &= \text{length of time series} \end{aligned} \tag{7}$$

The ARMA method combines moving average and autoregressive models (equation 8). The first term is a constant, the second term is white noise, the 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:

$$X_t = c + \epsilon_t + \sum_{i=1}^{p} \varphi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}. \tag{8}$$

where:

$$\begin{aligned} c &= \text{a constant} \\ \epsilon_t &= \text{error terms (white noise)} \\ \varphi &= \text{the autoregressive models parameters} \\ \theta &= \text{the moving average models parameters} \\ p &= \text{order of the autoregressive polynomial} \\ q &= \text{order of the moving average polynomial} \end{aligned}$$

The ARCH method models time series data by describing the variance of the current error term as a function of the sizes of the previous time periods' error terms [15]. This allows the method to support changes in the time dependent volatility, such as increasing and decreasing volatility in the same series [15]. The ARCH method is better than the ARMA method for volatile time-series data [16]. The StatsModels [17] 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 uses the fast Fourier transform algorithm to map a time series into the frequency domain. The algorithm returns complex numbers from which frequency, amplitude, and phase is extracted. Future demand and supply values are predicted by summing the significant components, then using the inverse Fourier transform method to return it into a usable form. The discrete Fourier transform (DFT) transforms a sequence of N complex numbers (X_k) into another sequence of complex numbers (x_n) [18]:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N}. \tag{9}$$

where:

$$\begin{aligned} X &= \text{sequence of complex numbers} \\ k &= 0, ..., N-1 \\ N &= \text{No. of complex numbers} \\ x &= \text{sequence of complex numbers} \\ n &= 0, ..., N-1 \end{aligned}$$

This method is implemented in `d3ploy` using the SciPy [19] Python package.

The `POLY` method fits the time series data with a user-defined n^{th} degree polynomial and uses the fitted trend-line to determine future demand and supply values:

$$Y_t = \beta_0 + \sum_{n=1}^{N} \beta_n t^n + \varepsilon \tag{10}$$

where:

$$\begin{aligned} t &= \text{time index} \\ n &= \text{polynomial order} \\ \beta &= \text{fitted parameters} \\ \varepsilon &= \text{unobserved random error} \end{aligned}$$

This method was implemented in `d3ploy` using the NumPy [20] Python package. The `EXP-SMOOTHING` and `HOLT-WINTERS` methods use a weighted average of time-series data with exponentially decaying weights for older time series values [21] 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 [21]:

$$y_{t+1} = \alpha y_i + (1 - \alpha) y_t. \tag{11}$$

where:

$$\begin{aligned} y &= \text{timeseries value} \\ \alpha &= \text{smoothing factor } (0 < \alpha < 1) \end{aligned} \tag{12}$$

The `HOLT-WINTERS` method applies triple exponential smoothing, resulting in higher accuracy when modeling seasonal time series data [22]:

$$\begin{aligned} F_{t+m} &= (S_t + m b_t) I_{t-L+m} \\ S_t &= \alpha \frac{y_t}{I_{t-L}} + (1 - \alpha)(S_{t-1} + b_{t-1}) \\ b_t &= \gamma (S_t - S_{t-1}) +)(1 - \gamma) b_{t-1} \\ I_t &= \beta \frac{y_t}{S_t} + (1 - \beta) I_{t-L} \end{aligned} \tag{13}$$

where:

$$\begin{aligned} F &= \text{forecast at m periods ahead} \\ t &= \text{time period index} \\ S &= \text{smoothed observation} \\ y &= \text{the observation} \\ b &= \text{trend factor} \\ I &= \text{seasonal index} \\ \alpha, \beta, \gamma &= \text{constants} \end{aligned}$$

The StatsModels [17] Python package was used to implement the `EXP-SMOOTHING` and `HOLT-WINTERS` methods in `d3ploy`.

2.5. *Stochastic-Optimizing Methods*

We implemented one stochastic-optimizing method: step-wise seasonal method (`SW-SEASONAL`). The method was implemented in `d3ploy` by the Auto-Regressive Integrated Moving Averages (ARIMA) method in the pmdarima [23] Python package. The ARIMA model is a dependent time series that is modeled as a linear combination of its own past values and past values of an error series [24]:

$$(1 - B)^d Y_t = \mu + \frac{\theta(B)}{\phi(B)} a_t \tag{14}$$

where:

$$\begin{aligned} t &= \text{time index} \\ \mu &= \text{mean term} \\ B &= \text{backshift operator, such that } BX_t = X_{t-1} \\ \phi(B) &= \text{autoregressive operator} \\ \theta(B) &= \text{moving average operator} \\ a_t &= \text{random error} \end{aligned}$$

3. Results

To demonstrate `d3ploy`'s capability conduct transition scenario analysis effectively and meet the objectives described in section 1.3, this section (1) demonstrates `d3ploy`'s capability in a simple transition scenario, (2) compares the use of different `d3ploy` prediction methods in EG01-EG23, EG01-EG24, EG01-EG29, and EG01-EG30 transition scenarios, and (3) demonstrates successful `d3ploy` setup of EG01-EG23, EG01-EG24, EG01-EG29, and EG01-EG30 transition scenarios. The input files and scripts to reproduce the results and plots in this paper are found in [25] and [26].

3.1. *Demonstration of `d3ploy`'s Capabilities*

We conducted a simple transition scenario simulation with linearly increasing power demand to demonstrate `d3ploy`'s capabilities and inform input parameter

	Input Parameters	Simple Transition Scenario
Required	Demand driving commodity	Power
	Demand equation [MW]	$t < 40 = 1000, t \geq 40 = 1000 + 250t$
	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 scenario with linearly increasing power demand.

choices when setting up complex many-facility transition scenarios. This simulation is defined as *simple* since it only includes three facility types: `source`, `reactor`, and `sink`. The simulation begins with ten `reactor` facilities (`reactor1` to `reactor10`). These reactors have staggered cycle lengths and lifetimes to prevent simultaneous refueling and set up 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 5, 6a, and 6b demonstrate `d3ploy`'s capability to deploy reactors and supporting facilities to minimize undersupply when meeting linearly increasing power demand and subsequent secondary commodities demand. In Figure 5 there exists no time steps in which the supply of power falls under demand, meeting the main objective of `d3ploy`. By using a combination of the `FFT` method for predicting demand and a power supply buffer of 2000MW (the capacity of 2 reactors), we minimized the number of undersupplied time steps for every commodity.

In figure 6a, a large-throughput source facility is initially deployed to meet the large initial fuel demand for the commissioning of ten reactors. Deployment of a large-throughput source facility for the first few time steps ensures `d3ploy`

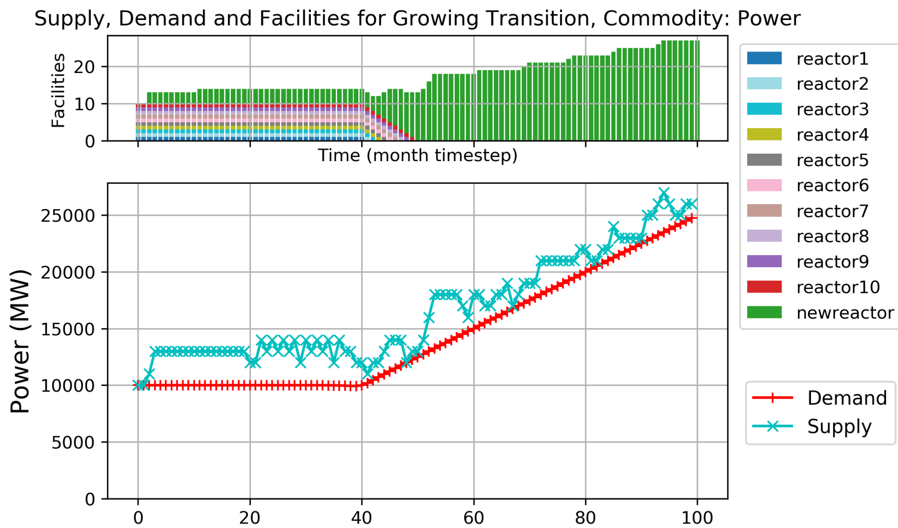
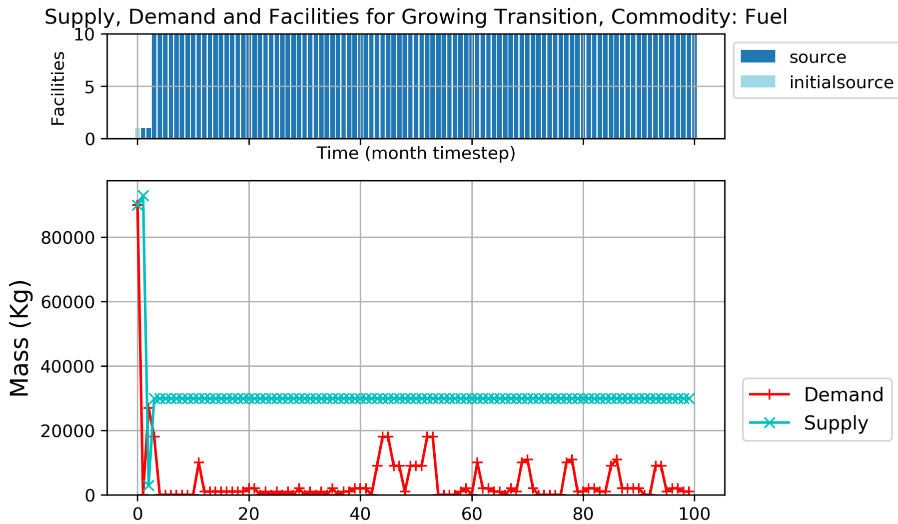


Figure 5: Power demand and supply, and reactor facility deployment plot for a simple linearly increasing power demand transition scenario with three facility types: `source`, `reactor`, and `sink`. The simulation begins with `reactor1` to `reactor10` and `d3ploy` deploys `newreactors` to meet increasing power demand. Power undersupply was avoided entirely.

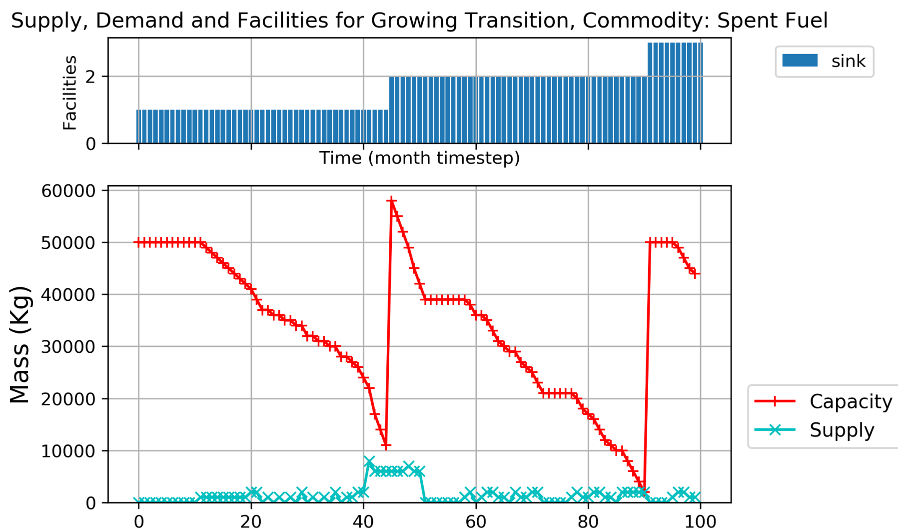
does not deploy supporting facilities that become redundant at later times in the simulation. This reflects reality in which reactor manufacturers accumulate an appropriate amount of fuel inventory before starting up reactors.

3.2. Comparison of Prediction Methods

EG01-EG23, EG01-EG24, EG01-EG29, and EG01-EG30 transition scenarios are set up in Cyclus using `d3ploy`. We ran each transition scenario with different prediction methods to determine the prediction method that best minimizes power undersupply for that scenario. We defined the EG01-EG23 and EG01-EG29 transition scenario simulations to have a constant power demand, while EG01-EG24 and EG01-EG30 have a linearly increasing power demand. Similar to the simple transition scenario, these transition scenario simulations begin with an initial fleet of LWRs that start progressively decommissioning



(a) Fuel demand and supply, and source facility deployment plot. Fuel is demanded by reactors and supplied by source facilities. There is only one time step with undersupply of fuel.



(b) Spent fuel capacity and supply, and sink facility deployment plot. Spent fuel is supplied by reactors and the capacity to store them is provided by sink facilities. There are no time steps with under-capacity of sink space.

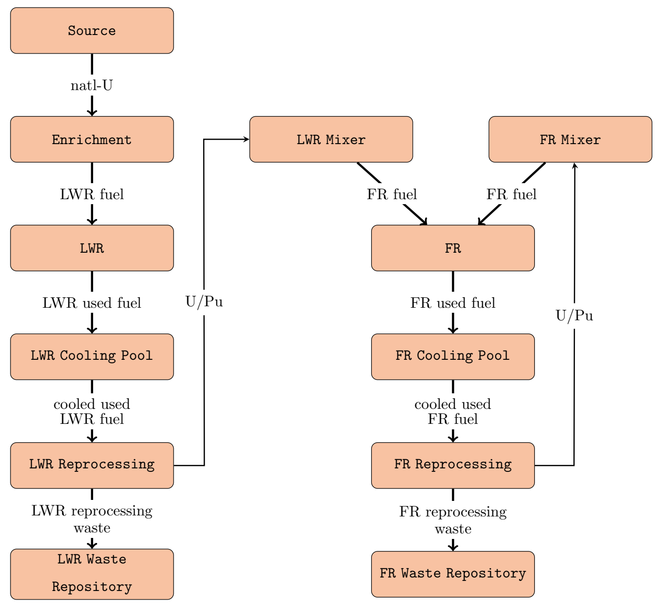
Figure 6: Simple linearly increasing power demand transition scenario with three facility types: `source`, `reactor`, and `sink`.

at the 80-year mark, after which `d3ploy` deploys SFRs and MOX LWRs to meet the power demand. Figure 7 shows the setup of facilities and mass flows for EG01-EG23 and EG01-EG29 in Cyclus. In EG01-EG23 and EG01-EG29, recycled plutonium from LWR spent fuel produces SFR fuel. EG01-EG24 and EG01-EG30 are similar to EG01-EG23 and EG01-EG29, respectively, with the exception that all transuranic elements are recycled.

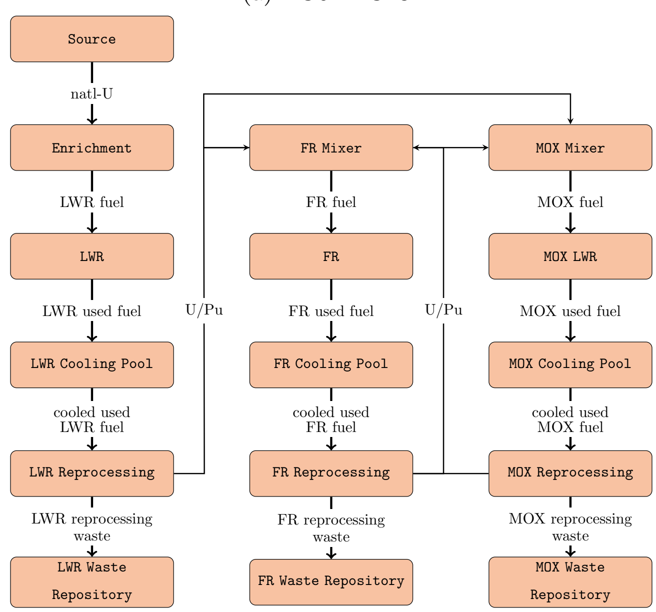
In Figure 8, each histogram represents the number of time steps with undersupply or under capacity for all commodities for each prediction method. Table 4 shows the total number of time steps with power undersupply for constant power EG01-EG23 and EG01-EG29 transition scenarios and linearly increasing power EG01-EG24 and EG01-EG30 transition scenarios for each prediction method. Figure 8 demonstrates that the `POLY` method performed the best for the EG01-EG23 transition scenario, with the smallest bars on the plot, indicating that they have the fewest number of time steps with undersupply and under capacity of commodities. We conducted a similar analysis for the constant power EG01-EG29 scenario, and as seen in Table 4, the `POLY` prediction method also performed best for minimizing undersupply of power.

In Figure 9, each histogram represents the number of time steps with undersupply or under capacity for all commodities for each prediction method. Figure 9 demonstrates that the `FFT` method performed the best for the EG01-EG24 transition scenario. We conducted a similar analysis for the linearly increasing power EG01-EG30 scenario, and as seen in Table 4, the `FFT` prediction method also performed best for minimizing undersupply of power.

Figures 8, 9, and Table 4 show 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 primary objective of minimizing the power undersupply, sensitivity analysis of the power supply buffer is conducted with the best-performing prediction method for each transition scenario.



(a) EG01-EG23.



(b) EG01-EG29.

Figure 7: Facility and mass flow of the transition scenarios EG01-EG23 and EG01-EG29 in Cyclus. EG23 and EG29 are closed fuel cycles with continuous recycling of U/Pu. EG23 consists of fast reactors, while EG29 consists of both fast and thermal reactors.

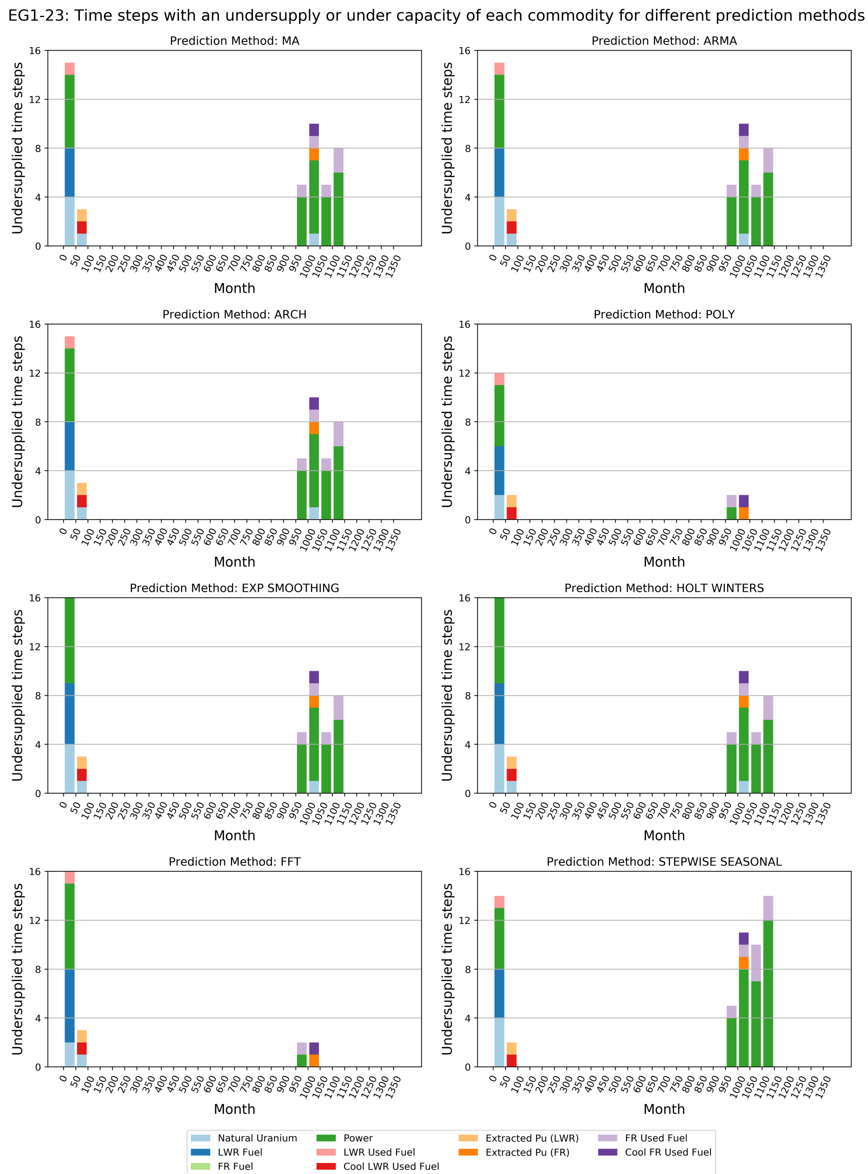


Figure 8: EG01-EG23 transition scenario with constant power demand. Each subplot shows the total number of time steps in which there exists undersupply and under capacity of commodities for each prediction method. The different colors represent different commodities and each vertical bar refers to 50 time steps in the simulation. The `POLY` method performs the best, with the least number of time steps with undersupply and under capacity.

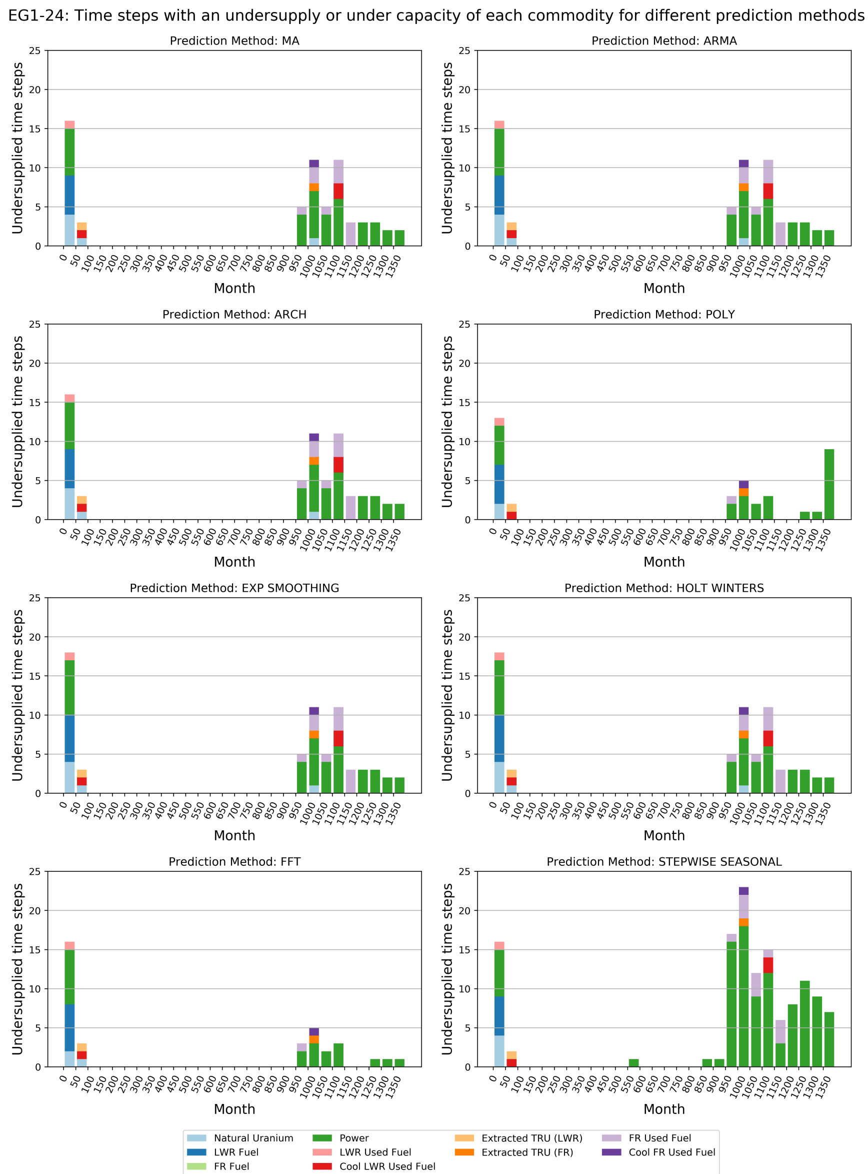


Figure 9: EG01-EG24 transition scenario with linearly increasing power demand. Each subplot shows the total number of time steps in which there exists undersupply and under capacity of commodities for each prediction method. The different colors represent different commodities and each vertical bar refers to 50 time steps in the simulation. The `FFT` method performs the best, with the least number of time steps with undersupply and under capacity.

No. of Time Steps with Power Undersupply for Each Transition Scenario				
Algorithm	**EG01-EG23**	**EG01-EG24**	**EG01-EG29**	**EG01-EG30**
`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: Total number of time steps with undersupply of power for the EG01-EG23, EG01-EG24, EG01-EG29, and EG01-EG30 transition scenarios for different prediction methods.

3.3. Comparison of Power Buffer Sizes

For the EG01-EG23, EG01-EG24, EG01-EG29, and EG01-EG30 transition scenarios, the power buffer size is varied for the best-performing prediction method, which is the `POLY` method for EG01-EG23 and EG01-EG29, and the `FFT` method for EG01-EG24 and EG01-EG30. Varying the power buffer size does not impact the number of undersupplied time steps for the EG01-EG23 and EG01-EG29 constant power demand transition scenarios. In Table 7, there are 6 and 4 time steps in which there is power undersupply for the EG01-EG23 and EG01-EG29 transition scenarios, respectively. As seen in figure 8, these undersupply time steps occur at the beginning of the simulation and for one time step when the transition begins. We expected this 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 reactors and supporting fuel cycle facilities. When the transition begins, power is undersupplied for one time step, following this, `d3ploy` accounts for the undersupply by deploying facilities to meet power demand. Therefore, we minimized the power undersupply for constant power EG01-EG23 and EG01-EG29 transition scenarios with a 0MW power supply buffer.

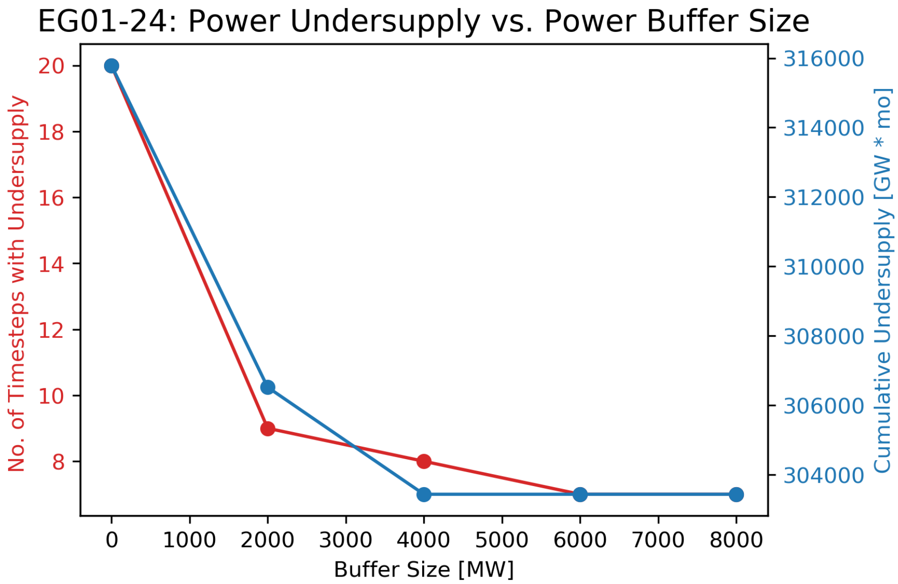
Buffer [MW]	Undersupply	EG01-EG24	EG01-EG30
0	Time steps [#]	20	9
	Energy $[GW \cdot mo]$	315791	152517
2000	Undersupplied [#]	9	6
	Energy $[GW \cdot mo]$	306520	147166
4000	Time steps [#]	8	6
	Energy $[GW \cdot mo]$	303438	143166
6000	Time steps [#]	7	5
	Cumulative $[GW]$	303438	139083
8000	Time steps [#]	7	5
	Energy $[GW \cdot mo]$	303438	135083

Table 5: The effect of 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.

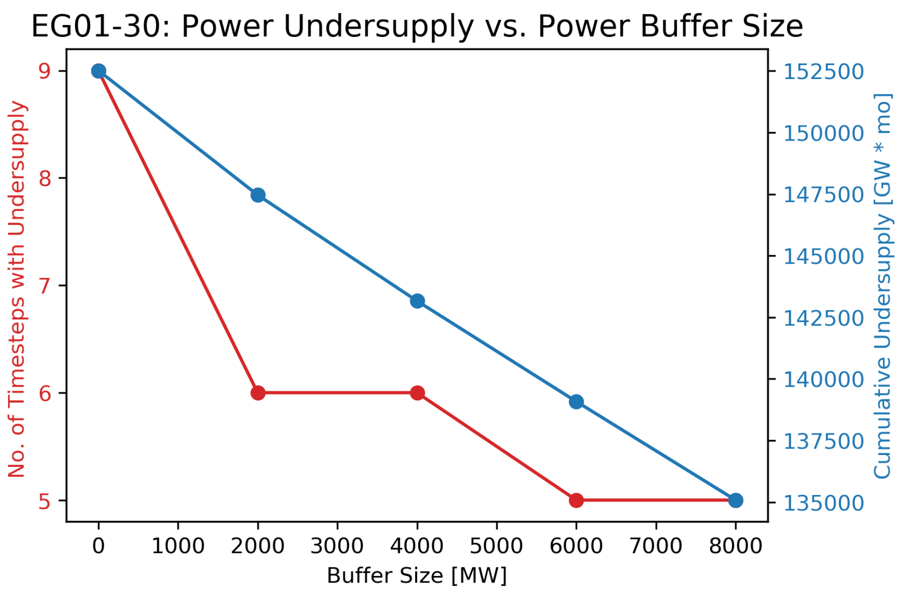
We varied the power buffer size for the EG01-EG24 and EG01-EG30 linearly increasing power demand transition scenarios. Figures 10a, 10b, and Table 5 show that increasing the buffer size increases the robustness of the supply chain by minimizing power undersupply. The cumulative undersupply is minimized with a 6000MW and 8000MW buffer for EG01-EG24 and EG01-EG30 respectively. However, this means 6 or 8 extra reactors are required, which is unrealistic. In Figure 10a, a 4000MW buffer size has 8 time steps with undersupply, while a 6000MW buffer size has 7 time steps with undersupply. In Figure 10b, a 2000MW buffer size has 6 time steps with undersupply, while a 8000MW buffer size has 5 time steps with undersupply. The extra commissioning of multiple reactors does not justify the 1 time step. Therefore, a buffer of 4000MW and 2000MW minimizes the power undersupply for EG01-EG24 and EG01-EG30 transition scenarios, respectively.

3.4. Best Performance Models

Table 6 shows the `d3ploy` input parameters for EG01-EG23, EG01-EG24, EG01-EG29, and EG01-EG30 transition scenarios that minimize the undersupply



(a) EG01-EG24: Power buffer size vs. cumulative undersupply



(b) EG01-EG30: Power buffer size vs. cumulative undersupply

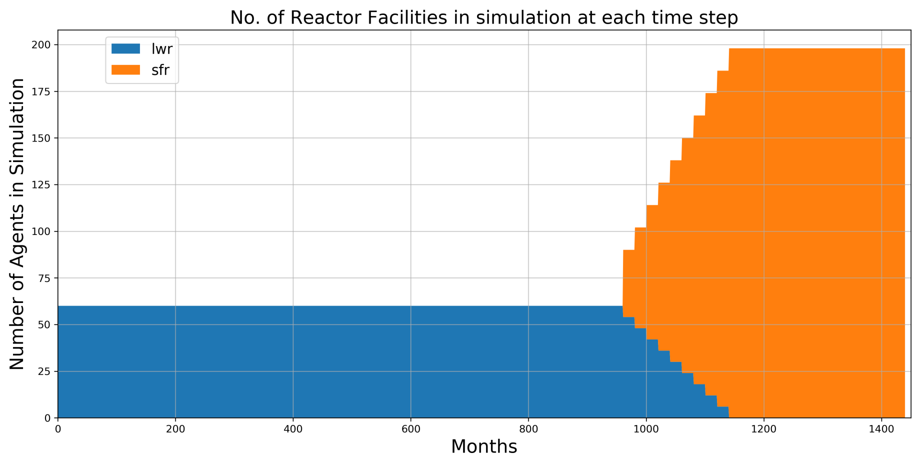
Figure 10: The effect of sensitivity analysis of power buffer size on cumulative undersupply of power for EG01-EG24 and EG01-EG30 transition scenarios with linearly increasing power demand using the `FFT` prediction method.

Input Parameter	Simulation Description			
	EG01-EG23	**EG01-EG24**	**EG01-EG29**	**EG01-EG30**
Demand Driving Commodity	Power	Power	Power	Power
Demand Equation [MW]	60000	60000 +250t/12	60000	60000 +250t/12
Prediction Method	`POLY`	`FFT`	`POLY`	`FFT`
Deployment Driving Method	Installed Capacity	Installed Capacity	Installed Capacity	Installed Capacity
Fleet Share Percentage	MOX: 85% SFR: 15%	MOX: 85% SFR: 15%	MOX: 85% SFR: 15%	MOX :85% SFR: 15%
Buffer type	Absolute			
Power Buffer Size [MW]	0	4000	0	2000

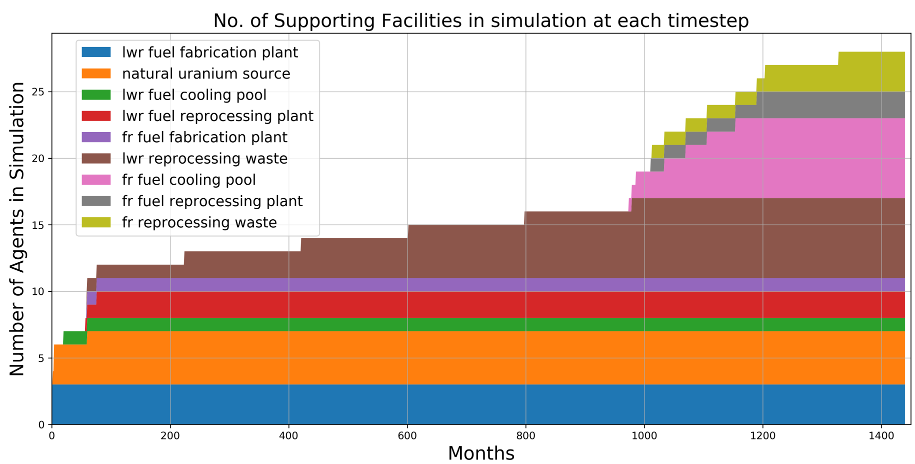
Table 6: `d3ploy`'s input parameters for EG01-EG23, EG01-EG24, EG01-EG29, and EG01-EG30 transition scenarios that minimizes undersupply of power and minimizes the undersupply and under-capacity of the other facilities.

of power as well as the undersupply and under-capacity of the other commodities in the simulation. The need for commodity supply buffers 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.

Figures 11 and 12 show time-dependent deployment of reactor and supporting facilities for the EG01-EG23 constant power demand and EG01-EG30 linearly increasing power demand transition scenarios, respectively. `d3ploy` automatically deploys reactor and supporting facilities to set up a supply chain to meet power demand during a transition from LWRs to SFRs for EG01-EG23, and from LWRs to MOX LWRs and SFRs for EG01-EG30. EG01-EG24 and EG01-EG29 facility deployment plots are similar to EG01-EG23 and EG01-EG30, respectively, therefore they are not shown.

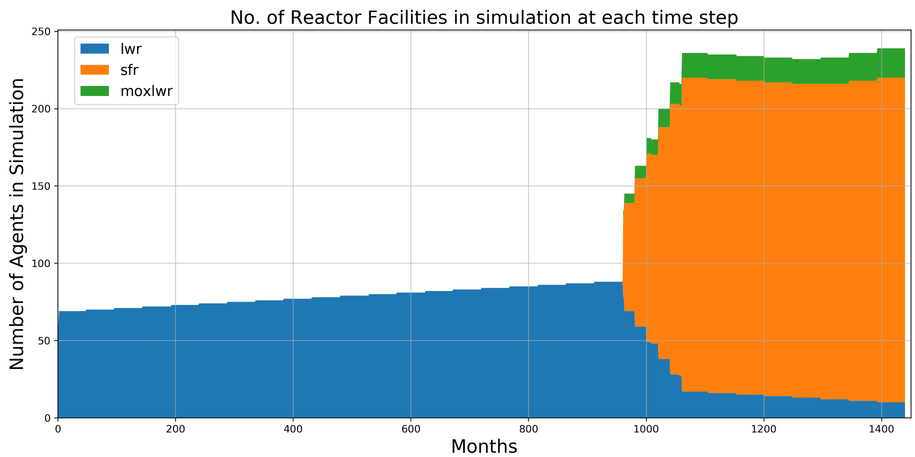


(a) EG01-EG23: Reactor Deployment

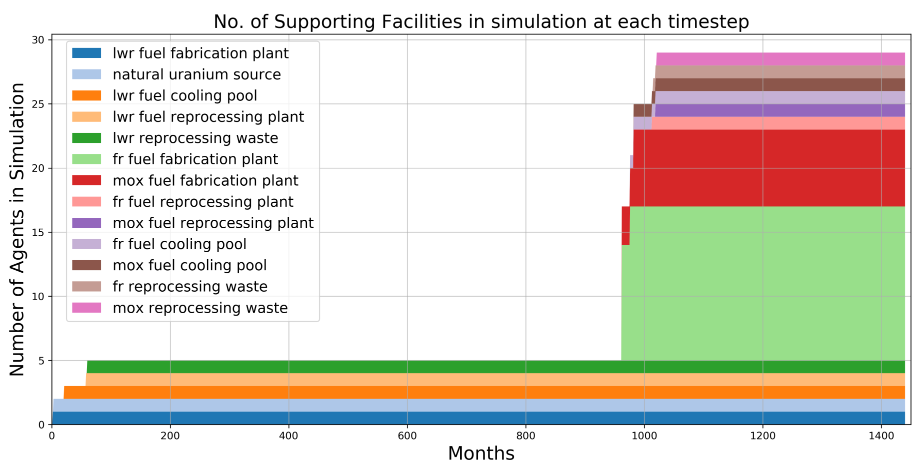


(b) EG01-EG23: Supporting Facility Deployment

Figure 11: Time dependent deployment of reactor and supporting facilities in the EG01-EG23 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-EG30: Reactor Deployment



(b) EG01-EG30: Supporting Facility Deployment

Figure 12: Time dependent deployment of reactor and supporting facilities in the EG01-EG30 linearly increasing power demand transition scenario. `d3ploy` automatically deploys reactor and supporting facilities to setup a supply chain to meet linearly increasing power demand of $60000 + 250t/12$ MW during a transition from LWRs to MOX LWRs and SFRs.

	No. of Time Steps with Undersupply			
Transition Scenario	EG01-EG23	EG01-EG24	EG01-EG29	EG01-EG30
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 key commodities for the best performing EG01-EG23, EG24, EG29, EG30 transition scenarios.

4. Conclusion

In this paper, we demonstrate that with careful selection of `d3ploy` parameters, we can 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 input parameter would result in the need to recalculate the deployment scheme to ensure no undersupply of power. Therefore, by automating this process, when the user varies input parameters in the simulation, `d3ploy` automatically adjusts the deployment scheme to meet the new constraints.

5. Future Work

We simulate transition scenarios to predict the future; however, when implemented in the real world, the transition scenario tends to deviate from the optimal scenario. Therefore, nuclear fuel cycle simulators must be used to conduct sensitivity analysis studies to understand the subtleties of a transition scenario better to reliably inform policy decisions. Previously it was difficult to conduct sensitivity analysis with Cyclus as users have to manually calculate the deployment scheme for a single change in an input parameter. By using the `d3ploy` capability, sensitivity analysis studies are more efficiently conducted as facility deployment in transition scenarios are automatically set up.

6. Acknowledgments

Department of Energy (DOE) Office of Nuclear Energy funds this research through the 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. Special thanks to Kip Kleimenhagen and Matthew Kozak for their excellent proofreading help. We also thank our colleagues from the Cyclus community for collaborative Cyclus development.

The authors contributed to this work as described below. Gwendolyn J. Chee conceived and designed the simulations, wrote the paper, prepared figures and/or tables, performed the computation work, contributed to and validated the software product, and reviewed drafts of the paper. Roberto E. Fairhurst Agosta performed the computation work and contributed to the software product. Jin Whan Bae conceived and designed the simulations and contributed to and validated the software product. Robert R. Flanagan conceived and designed the simulations and contributed to the software product. Anthony M. Scopatz conceived and designed the simulations and acquired funding support for the project. Kathryn D. Huff directed and supervised the work, acquired funding support for the project, conceived and designed the simulations, contributed to the software product, and reviewed drafts of the paper.

References

[1] A. M. Yacout, J. J. Jacobson, G. E. Matthern, S. J. Piet, A. Moisseytsev, Modeling the Nuclear Fuel Cycle, in: The 23rd International Conference of the System Dynamics Society," Boston, Citeseer, 2005. URL http://www.inl.gov/technicalpublications/Documents/3169906.pdf

[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] 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.

[4] 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

[5] Climate Change and Nuclear Power 2018, Non-serial Publications, INTERNATIONAL ATOMIC ENERGY AGENCY, Vienna, 2018. URL https://www.iaea.org/publications/13395/climate-change-and-nuclear-power-2018

[6] D. Petti, P. J. Buongiorno, M. Corradini, J. Parsons, The future of nuclear energy in a carbon-constrained world, Massachusetts Institute of Technology Energy Initiative (MITEI).

[7] 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

[8] 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

[9] G. Reikard, Predicting solar radiation at high resolutions: A comparison of time series forecasts, Solar Energy 83 (3) (2009) 342–349.

[10] M. Diagne, M. David, P. Lauret, J. Boland, N. Schmutz, Review of solar irradiance forecasting methods and a proposition for small-scale insular grids, Renewable and Sustainable Energy Reviews 27 (2013) 65–76.

[11] S. S. Soman, H. Zareipour, O. Malik, P. Mandal, A review of wind power and wind speed forecasting methods with different time horizons, in: North American Power Symposium 2010, IEEE, 2010, pp. 1–8.

[12] J. W. Taylor, P. E. McSharry, R. Buizza, Wind power density forecasting using ensemble predictions and time series models, IEEE Transactions on Energy Conversion 24 (3) (2009) 775–782.

[13] G. C. Souza, Supply chain analytics, Business Horizons 57 (5) (2014) 595–605.

[14] 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.

[15] R. F. Engle, Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation, Econometrica: Journal of the Econometric Society (1982) 987–1007.

[16] 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.

[17] S. Seabold, J. Perktold, Statsmodels: Econometric and statistical modeling with python, in: Proceedings of the 9th Python in Science Conference, Vol. 57, Scipy, 2010, p. 61.

[18] K. R. Rao, D. N. Kim, J. J. Hwang, Fast Fourier transform-algorithms and applications, Springer Science & Business Media, 2011.

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

[20] T. E. Oliphant, A guide to NumPy, Vol. 1, Trelgol Publishing USA, 2006.

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

[22] N. Sematech, Engineering statistics handbook, NIST SEMATECH.

[23] T. G. Smith, pmdarima: Arima estimators for python, 2017.

[24] S. A. S. Institute, SAS user's guide: statistics, Vol. 2, Sas Inst, 1985.

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

[26] G. Chee, K. Huff, arfc/transition-scenarios : MS Thesis Release (Dec. 2019). `doi:10.5281/zenodo.3569721`.