Research Activities in the ARFC Group

A Brief Summary

Advanced Reactors and Fuel Cycles Group

University of Illinois at Urbana-Champaign

September 5, 2019

Outline

1. Pyre
2. DDCA
3. Depletion and UDB
4. SaltProc
5. Acknowledgments

Motivation/Goals

Motivation

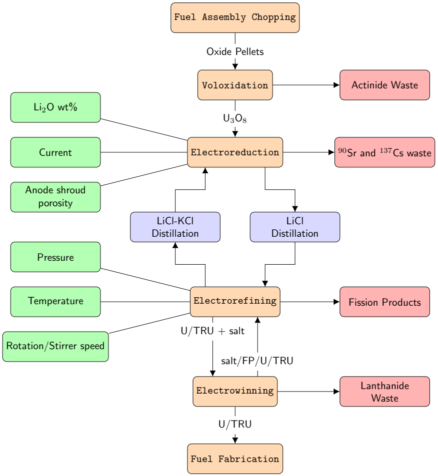
- Safeguard by design
- Model diversion inside facilities
- transition from LWR to SFR

Goals

- Detect diversion using signatures and observables.
- Optimum detector and inspection locations in pyroprocessing
- Characterize detection sensitivities and false positive rates

PyRe Archetype

- Facility containing multiple sub-processes:
  - Separately handled.
  - Independent transactions, possibility of diversion.
- Operation setting impact efficiency.
- Generic facility:
  - Multiple types of pyro plants.
  - LWR vs SFR.



Pyre - Diversion Options

Material diversion occurs in two different modes: **nefarious** or **operator**.

- **Nefarious Diversion** imagines diversion by a single bad actor with facility access.
- **Operator Diversion** imagines undeclared production.

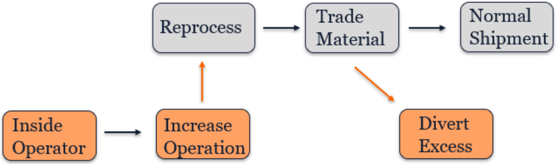


Figure: Operator vs nefarious diversion.

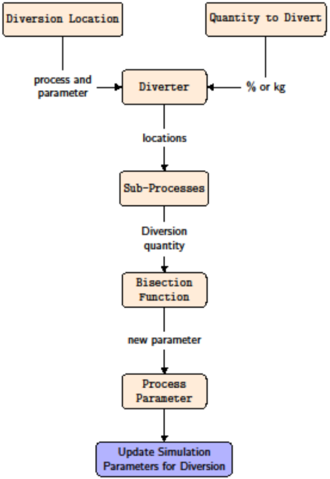
Diverter Class

Inputs:

- Location
  - Sub-process
  - Operation Setting
- Quantity
- Frequency
- Number of Diversions

Purpose

The goal of a separate diverter class is to allow this method to be used by facilities other than pyre through a toolkit.



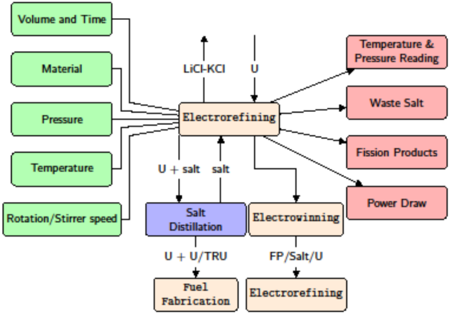
Diversion Detection

Diversion Detection

Material transactions are no longer a reliable method. Instead we use signatures and observables:

- Temperature, power draw, etc.

A Cumulative Sum change algorithm is used to detect any significant changes.

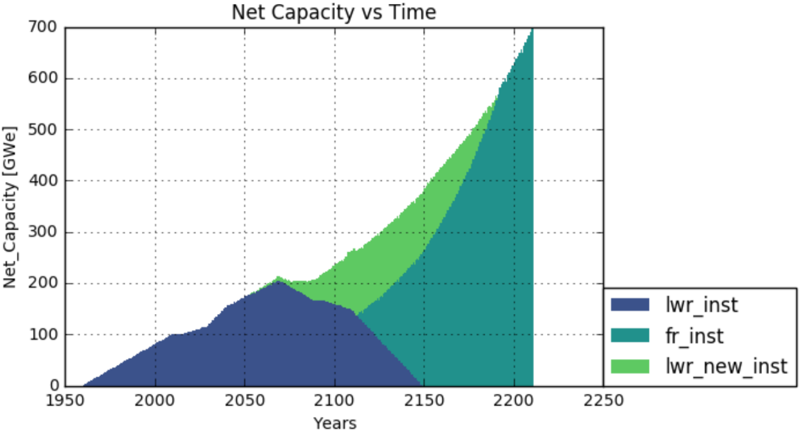


Transition Scenario

A main attraction of pyroprocessing is the ability to handle LWR and SFR waste.

- To verify this capability in PyRe, we ran an EG01 – EG24 transition scenario.
- We want to observe the following:
  - Appropriate deploying of PyRe
  - Ability to meet demand of new SFRs
  - Diversion capabilities
  - Accurate transition from UOX to SFR fuels

Transition Scenario - Results



Diversion Settings

Two Pyre prototypes:

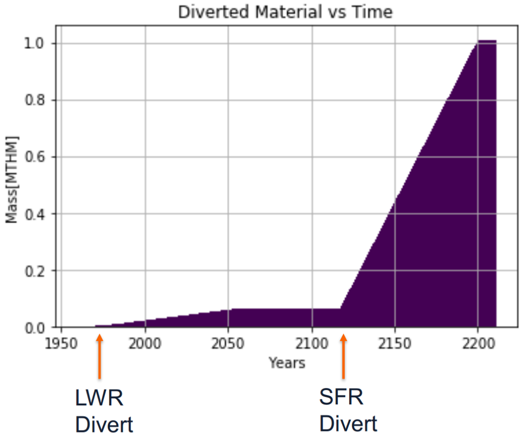
- LWR vs SFR

LWR Pyre:

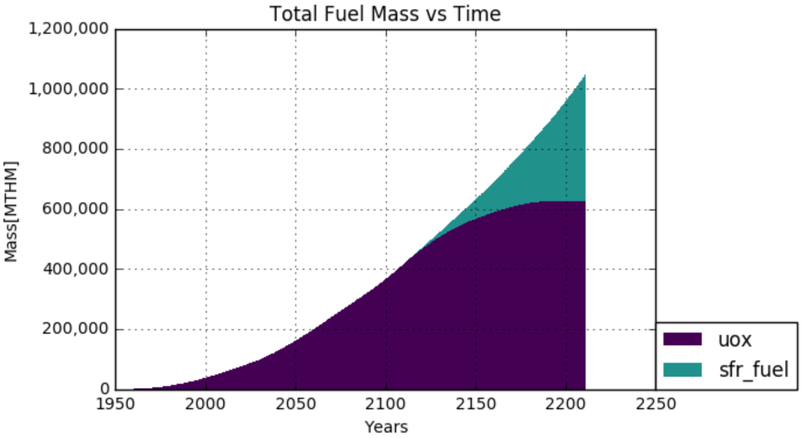
- Fewer diversions
- More material per instance
- Less frequent

SFR Pyre:

- Frequent diversion
- Small quantities



Transition Scenario - Utilization



Conclusions

We have developed a customizable method of diverting material from inside Cyclus facilities.

- Preliminary work has been done on the detection of two different types of diversion: Nefarious and Operator

PyRe was demonstrated to function as both LWR and SFR reprocessing method

- Generic facility capable of modeling multiple facility layouts

Outline

1. Pyre
2. DDCA
3. Depletion and UDB
4. SaltProc
5. Acknowledgments

DDCA Overview

Title: Demand-Driven Cycamore Archetypes

PI: Anthony Scopatz, University of South Carolina[1]

Co-PI: Kathryn Huff, University of Illinois at Urbana-Champaign

Start: October, 2016

End: October, 2017

Objectives: Develop an in situ demand driven development schedule calculation through non-optimizing, deterministic-optimizing, and stochastic-optimizing algorithms as Cyclus archetypes. Demonstrate these new archetypes in program-supporting fuel cycle scenarios.

[1] Anthony departed academia in year 2 of the project. The PIship was transferred to Travis Knight at USC

Quick Statistics

Publications Affiliated with this Work

Journal Articles	3 (2 upcoming)
Full Conference Papers	3 (2 upcoming)
Conference Summaries	7
Technical Reports	2 (1 upcoming)
Theses	1MS (2 upcoming)

Students Supported

The funding supported graduate students and occasional undergraduates at UIUC. **Jin Whan Bae** recieved his MS and is now at ORNL purusing Cyclus usability. **Gwendolyn Chee** is writing an MS thesis related to this work and related work conducted at ANL with Bo Feng. Undergraduate **Louis Kissinger** is a baccalaureate researcher this year in MCS at ANL. Others include **Roberto Fairhurst**, **Gyu Tae Park**, **Snehal Chandan**, and **Aditya Bhosale**.

Motivation

Main Objective

To improve usability of Cyclus for transition scenarios.

Main Challenge

Deploying reactors to meet power demand is trivial, and existed in the earliest versions of Cyclus. **Automated, predictive deployment and decommissioning of other facilities is more complex.** These include mining, milling, enrichment, fuel fabrication, reprocessing, and others.
For example, a balanced closed fuel cycle may require ensuring that there is enough fast reactor fuel for their operation and may drive deployment of a fleet of light water reactors.

Goals

Goals of this work

- Develop demand driven deployment capabilities in CYCLUS (`d3ploy`)
- Demonstrate the use of `d3ploy` to set up EG01-23, EG01-24, EG01-29 EG01-30 transition scenarios with constant and linearly increasing power demand curves.

Method

Because Cyclus is Agent-Based

- Its regions and institutions have the agency to dynamically make and alter deployment decisions.
- Each agent can make their own predictions of the future based on current and past performance of the simulation.

We embedded advanced time series prediction algorithms to automatically deploy fuel cycle facilities for the user. This was implemented in `d3ploy`, an Institution agent.

Motivation

Gap in capability: User must define when support facilities are deployed

Figure: User defined Deployment Scheme

Bridging the gap: Developed demand-driven deployment capability in Cyclus. This capability is named `d3ploy`.

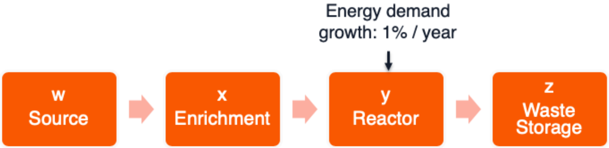


Figure: Demand Driven Deployment Scheme

`d3ploy` Objectives

`d3ploy`'s Main Objective

Minimize the number of time steps of undersupply or under capacity of power.

`d3ploy`'s Sub-Objectives

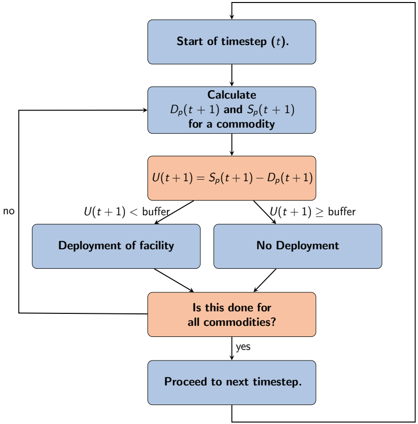
- Minimize the number of time steps of undersupply or under capacity of any commodity.
- Minimize excessive oversupply of all commodities

d3ploy Input Parameters

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

	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

d3ploy logic flow



D_p : $PredictedDemand$
S_p : $PredictedSupply$
$U = S_p - D_p$

Figure: d3ploy logic flow at every timestep in CYCLUS [?].

`d3ploy` Prediction Methods

Non-Optimizing Methods

- Moving Average (`ma`)
- Autoregressive Moving Average (`arma`)
- Autoregressive Heteroskedasticity (`arch`)

Deterministic-Optimizing Methods

- Fast Fourier Transform (`fft`)
- Polynomial Fit (`poly`)
- Exponential Smoothing
- Triple Exponential Smoothing (`holt-winters`)

Stochastic-Optimizing Methods

- Auto-Regressive Integrated Moving Averages (`ARIMA`)

Breakdown of Results

4 transition scenarios sought to minimize undersupply and under capacity of all commodities.

1. EG01-23 ($P(t) = P_0$)
2. EG01-24 ($P(t) = P_0 + rt$)
3. EG01-29 ($P(t) = P_0$)
4. EG01-30 ($P(t) = P_0 + rt$)

This is achieved by:

1. Comparison of prediction methods for each of 4 scenarios is conducted to determine the best method.
2. Sensitivity analysis of power supply buffer is conducted to determine best buffer size.
3. Using best prediction method, look ahead rate, buffer size, demonstrate `d3ploy` deploying reactor and supporting facilities to meet power demand for 4 scenarios.

Comparison of Prediction Methods

EG01-23 Constant Power Demand Transition Scenario

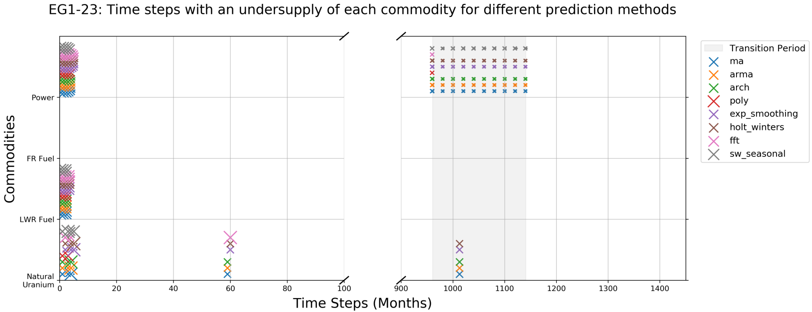


Figure: Time dependent undersupply of commodities for different prediction methods for the EG01-23 Transition Scenario with Constant Power Demand. The size of each cross is based on the size of the undersupply. Fewer crosses on plot indicates the method is more successful at preventing undersupply of each commodity

Comparison of Prediction Methods

EG01-23 Constant Power Demand Transition Scenario

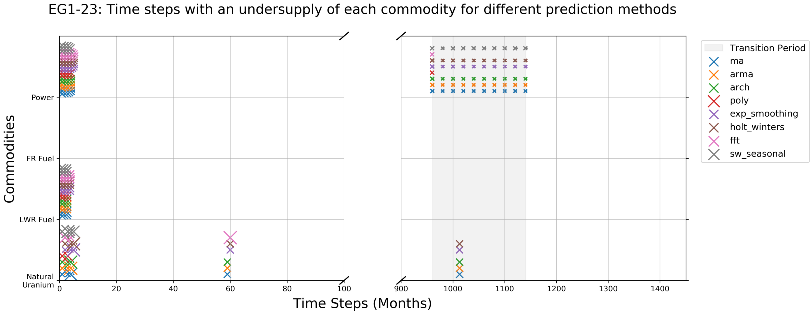


Figure: Time dependent undersupply of commodities for different prediction methods for the EG01-23 Transition Scenario with Constant Power Demand. The size of each cross is based on the size of the undersupply. Fewer crosses on plot indicates the method is more successful at preventing under capacity of each commodity

Comparison of Prediction Methods

Main Takeaway

The best performing prediction method for each transition scenario is:

1. EG01-23 Constant Power Demand: `poly`
2. EG01-24 Linearly Increasing Power Demand: `fft`
3. EG01-29 Constant Power Demand: `poly`
4. EG01-30 Linearly Increasing Power Demand: `fft`

Best Performing Transition Scenarios

Undersupply and under capacity of commodities for the best performing transition scenarios

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

	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

Conclusion

These results demonstrate that by carefully selecting `d3ploy` parameters, we are able to **effectively automate deployment** of reactor and supporting facilities to set up constant and linearly increasing power demand transition scenarios for EG01-23, EG01-24, EG01-29, and EG01-30 with minimal power undersupply.

Not completely eliminating undersupply and under capacity of commodities in the simulation 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.

Outline

1. Pyre
2. DDCA
3. Depletion and UDB
4. SaltProc
5. Acknowledgments

Outline

1. Pyre
2. DDCA
3. Depletion and UDB
4. SaltProc
5. Acknowledgments

SaltProc flowchart

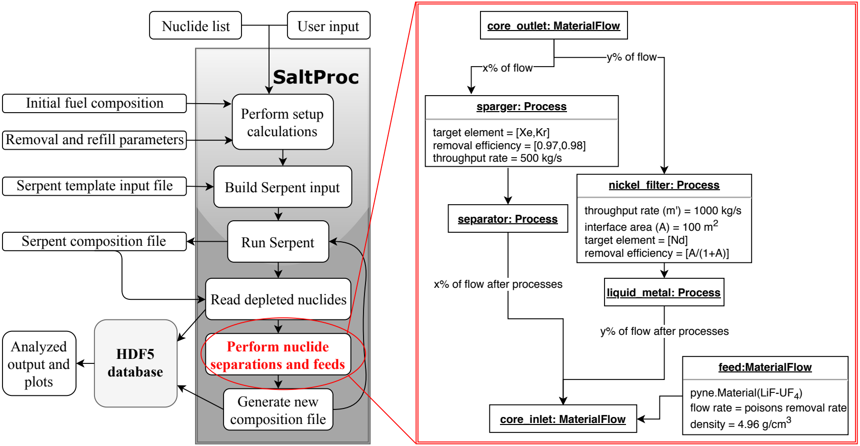


Figure: Tentative generic flowchart for SaltProc v1.0 python package.

Outline

1 Pyre

2 DDCA

3 Depletion and UDB

4 SaltProc

5 Acknowledgments

Acknowledgement

This work is supported by U.S. Department of Energy, Nuclear Energy University Program, under contract #NEUP-FY16-10512.

References I