Advanced Nuclear Fuel Cycle Transitions:

Optimization, Modeling Choices, and Disruptions

By

Robert W. Carlsen

A dissertation submitted in partial fulfillment of

the requirements for the degree of

Doctor of Philosophy

(Nuclear Engineering)

at the

UNIVERSITY OF WISCONSIN–MADISON

2016

Date of final oral examination: 03/29/2016

The dissertation is approved by the following members of the Final Oral Committee:

Paul P.H. Wilson, Professor, Nuclear Engineering
Erich Schneider, Professor, Nuclear Engineering
Jeff Linderoth, Professor, Industrial and Systems Engineering
Vickie Bier, Professor, Industrial and Systems Engineering
Michael L. Corradini, Professor, Nuclear Engineering

This work is dedicated to my wonderful wife Esther.

ACKNOWLEDGMENTS

This research was performed in part using funding received from the Department of Energy's Nuclear Energy University Programs (NEUP). The author would like to thank the NEUP for its generous support.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Nuclear fuel cycle analysis is a field focused on understanding and modeling the nuclear industry and ecosystem at a macroscopic level. To date, fuel cycle analysis has mostly involved hand-crafting details of fuel cycle scenarios for investigation. Many different tools have evolved over time to help address the need to investigate both the equilibrium properties of nuclear fuel cycles and the dynamics of transitions between them. There is great potential for computational resources to improve both the quality of answers and the size of questions that can be asked. Cyclus is one of the first nuclear fuel cycle simulators to strongly accommodate larger-scale analysis with its free availability, liberal open-source licensing, and first-class Linux support. Cyclus also provides features that uniquely enable investigating the effects of modeling choices and modeling fidelity within fuel cycle scenarios. This is made possible by the complementary nature of Cyclus' dynamic resource exchange and plugin based architecture. This work is divided into three major pieces focusing on optimization, investigating effects of modeling choices, and dealing with uncertainty.

Effective optimization techniques are developed for automatically determining desirable facility deployment schedules for fuel cycle scenarios with Cyclus. A novel method for mapping optimization variables to deployment schedules is developed. This method allows relationships between reactor types and power capacity constraints to be represented implicitly in the definition of the optimization variables. This not only enables optimizers without constraint support to be used, but it also prevents wasting computational resources searching through many infeasible deployment schedules. With the simplified constraint handling, optimization can be used to analyze larger problems in addition to providing better solutions generally. The developed methodology also enables the deployed power generation capacity over time and the deployment of non-reactor support facilities to be included as optimization variables.

There exist many fuel cycle simulators built with many different combinations of mod-

eling choices and assumptions. This makes comparing results from them difficult. The flexibility of Cyclus makes it a rich playground for comparing the effects of such modeling choices in a consistent way. Effects such as reactor refueling cycle synchronization, inter-facility competition, on-hand inventory requirements, and others are compared in four fuel cycle scenarios each using combinations of fleet or individually modeled reactors with 1-month or 3-month long time steps. There are noticeable differences in results from the different cases. The largest differences are seen during periods of constrained fuel availability for reactors. Research into the effects of modeling choices such as these can help improve the quality and consistency of fuel cycle analysis codes in addition to increasing confidence in the utility of fuel cycle analysis generally.

With respect to deploying new nuclear technologies, there is significant uncertainty associated with many things such as time-frames for technology availability, the cost of deploying advanced reactor technologies, and ever-changing policy and regulation. Historically, fuel cycle analysis has focused on answering questions of fuel cycle feasibility and, to a lesser extent, questions of optimality. To date, however, there has not been much work done to address this uncertainty in fuel cycle analysis. Such work could help extend the reach of fuel cycle analysis toward answering questions of fuel cycle robustness. To help address this need, a methodology is developed and demonstrated for evaluating fuel cycle deployment strategies that accounts for uncertainty. Rather than assuming an exact, particular future, techniques are developed for measuring the hedging properties of deployment strategies under a diverse set of possible futures. Additionally, methods for using optimization to automatically find good hedging strategies are demonstrated.

1 INTRODUCTION

Nuclear fuel cycle analysis is a domain of nuclear engineering dedicated to investigating and modeling the nuclear industry at a macroscopic level. The properties of nuclear materials and processes manipulating them over their life cycle from mining to recycling to disposal are very complex. Nuclear materials are mixtures of many isotopes/nuclides that each have unique and important properties affecting their utility and treatment. Different reactor and spent nuclear fuel recycling technologies have many different constraints both on what they material consume and produce. Understanding how these constraints propagate through the fuel cycle as well as how materials' isotopes/nuclides accumulate at different locations within the fuel cycle for different technology configurations quickly can become a difficult problem. For example, determining how a particular advanced reactor technology may affect spent fuel disposal properties and volumes can become very difficult — especially in cases involving things such as recycling and transitions between technologies. Tackling the complexity arising from this challenge has resulted in the development of special nuclear fuel cycle analysis tools.

The primary type of tool for such investigations is referred to as a *fuel cycle simulator*. These simulators allow the construction of scenarios with particular facility deployments and configuration and track nuclear material needs, material flows, and the accumulation of various nuclides at different places within the fuel cycle (e.g. reactors, interim spent fuel storage, long-term waste disposal repositories, etc.). This information enables drawing conclusions about the costs and benefits associated with particular fuel cycle configurations and allows the exploration of limitations and constraints associated with various fuel cycle related decisions.

To date, nuclear fuel cycle analysis has mostly involved hand-crafting many details of fuel cycle scenarios for analysis. There is great potential for utilizing computational resources to improve both the quality of answers and the size of questions that can be

asked. Cyclus is one of the first nuclear fuel cycle simulators to strongly accommodate larger-scale analysis with its free availability, liberal open-source licensing, and first-class Linux support. This work develops methods for automatically determining desirable facility deployment strategies, examines effects of a few, common fuel cycle modeling assumptions, and demonstrates techniques for evaluating deployment strategies under uncertain future conditions.

After discussing background and surveying related literature, the research is broken into three separate experiments:

- **Experiment 1**: Optimization Structure and Comparison.
- **Experiment 2**: Facility Discretization and Time Step Effects.
- **Experiment 3**: Hedging Strategies for Disruption.

In Experiment 1, an effective technique for automatically determining desirable deployment schedules for fuel cycle scenarios is developed using the Cyclus fuel cycle simulator. Important pieces of this exercise include: designing a representative fuel cycle scenario (e.g. simulation duration, reactor types/configuration, etc.), defining and structuring optimization variables, and comparing/evaluating several optimization algorithms.

Carefully developing an effective way to map optimization variables to a deployment schedule has many potential benefits such as: reducing the complexity and size of the solution space, eliminating the need for sophisticated constraints, and side-stepping optimizer restrictions. The several optimization algorithms selected for comparison will be evaluated based on their performance in a fuel cycle scenario by comparing convergence results to each other and also by comparing results with prior, related work. One custom optimizer is also crafted using knowledge of the typical problem structure of interest. Although the investigated fuel cycle scenarios will not necessarily be realistic in many details, the results will demonstrate the ability of this type of analysis to discover interesting strategies and

provide insight. Optimization for fuel cycle analysis can become an invaluable tool for generally identifying potential pitfalls and opportunities.

The heavy Cyclus usage, developed optimization methods, and scenario details from Experiment 1 establish a good foundation for investigating impacts of some common modeling assumptions made in fuel cycle analysis. Cyclus' unique ability to easily swap which facility models are used in the scenarios makes adjusting design assumptions much more feasible. Experiment 2 investigates effects of two simple things that may often be taken for granted in fuel cycle analysis: time step duration and discretization of facilities (e.g. individual vs. fleet reactor modeling).

Modeling reactors individually can provide valuable insight not possible with the fleet-based reactor modeling. Individual reactor modeling, for example, enables the investigation of refueling staggering among many facilities. Although individual reactor outage modeling can be very useful for certain types of analysis, it can negatively impact result quality if things such as refueling cycle staggering are not handled appropriately. If individual reactor modeling fidelity is not required, then performance benefits may result in a fleet-based modeling approach being more appropriate. Cyclus' flexibility for accommodating different modeling choices uniquely enables many interesting comparisons. The value of realism in fuel cycle modeling can be greatly be enhanced by an associated understanding of how outcomes are affected by modeling fidelity — an understanding that can be enhanced through exercises like Experiment 2.

Experiment 3 further builds on the theme of Experiment 1 enabling bigger questions to be asked in fuel cycle analysis. This is made possible by Cyclus' ability to be deployed in cluster type environments in conjunction with unique methods developed for evaluating deployment strategies under uncertainty. Rather than assuming an exact, particular future (which is very unrealistic), strategies can be evaluated under a diverse set of possible futures. Questions like "How would a potential permanent moratorium on deploying technology X affect what we build now?" and "How can we hedge against likely changes in the balance

between economics and environmental concerns?" can begin to be approachable.

Experiment 3 develops methods for evaluating hedging effectiveness of particular deployment strategies under uncertainty. This requires more computational resources than optimization work for Experiments 1 and 2. For a single hedging strategy, optimal response deployments must be identified across the spectrum of uncertainty resolutions, requiring several optimization runs. Outcome distributions can then be generated that provide a good high-level mechanism for evaluating the risk-reward trade-off of different strategies. Discovering good hedging strategies is even more difficult. Various objective approximation techniques are developed in order to reduce complexity of searching for hedging strategies to manageable levels with varying reductions in accuracy. These techniques are demonstrated and evaluated.

2.1 Fuel Cycle Simulators

Many fuel cycle analysis tools have been created over the years built in a variety of ways. Early on, much fuel cycle analysis and simulation was done using spreadsheets. These tools often used simple macro and scripting capabilities to provide more complex behavior. Eventually analysis needs began to outgrow the spreadsheet form and there was a shift towards using system dynamics tools. Some of the more commonly used models are built using proprietary system dynamics software such as Powersim. Of these, Wilson says, "In many cases, a complete model of an advanced nuclear fuel cycle is one of the biggest models ever developed under these software systems" [41]:. The range of capability is large and greatly affects how the simulators can be used both logistically (e.g. deployed in large cluster environments) and scientifically (e.g. what metrics can be calculated, fidelity of reactor physics, etc.).

System dynamics based simulators consist of several "stocks" that have levels that change over time due to "flows" between them. The flows are determined by an equation that is a function of the state of the system (i.e. the system's stocks and flows). A modeled system is usually then solved in discrete time within the limitations of the software and constraints imposed by the model. In system dynamics simulators, the number of stocks and potential flows between them is generally a static property of the modeled system. System dynamics based fuel cycle simulators use a collection of stocks to represent the state of groups of like facilities commonly referred to as "fleets". As the simulation steps through time, the levels of stocks (e.g. reactor fresh fuel inventory, repository waste inventories, etc.) are adjusted according to the calculated flow values.

CAFCA [4] and VISION [19] are two fuel cycle simulators that have been used for optimization research recently. VISION is a system-dynamics based fuel cycle simulator built on the non-free, proprietary Powersim platform. VISION is only supported on

Windows and uses Microsoft Excel as a primary mechanism for data storage and analysis. VISION has fleet-based facility deployment. The user does not generally have direct control over deployment of individual facilities but can affect it in aggregate via inputs such as a power demand curve. VISION has been used to conduct various studies on fuel cycle transitions [30]. CAFCA is a system-dynamics based fuel cycle simulator, originally written in Matlab, now implemented on the non-free, proprietary Vensim platform. CAFCA is only available for the Windows and Mac platforms. Similar to VISION, the user does not generally have direct control over individual facility deployment but can affect it indirectly in aggregate. A more thorough discussion of the state of the art of fuel cycle simulation is provided by Gidden [16].

2.1.1 Cyclus

Cyclus is a new fuel cycle simulator developed primarily by the CNERG research group at the University of Wisconsin - Madison. It has many unique features that set it apart from other fuel cycle simulators. Rather than using a system dynamics approach, Cyclus is an agent-based simulator. A Cyclus simulation consists of many agents that interact with each other in various ways supported by the Cyclus kernel. An agent can be a tangible facility (e.g. a power reactor) or can represent more abstract entities such as facility-managing institutions and geo-political regions of the world. The simulation proceeds in discrete time steps giving each agent an opportunity to do internal processing and interact with other agents. There are two primary ways for Cyclus agents to interact: transacting "resources" (e.g. spent fuel, carbon credits, etc.) with each other and deploying/decommissioning other agents. Cyclus comprehensively tracks transacted resources and lifetime events for all agents. The following sections provide more detail about notable features that differentiate Cyclus from other simulators.

2.1.1.1 Deployment

In Cyclus, every agent (facility, institution, etc.) has the ability to schedule deployment and decommissioning of other individual agents; agents can even schedule their own decommissioning. This removes the common fleet-based deployment related restrictions common in other fuel cycle simulators. Events associated with the lifetime of each facility are tracked individually. If desired, the user can exercise full control over deployment of every single facility in the simulation including but not limited to: reactors, mining facilities, fuel fabrication facilities, repositories, etc. While this level of control is possible, this behavior is not required by Cyclus. It is just as easy to run simulations with the varying levels of deployment decision automation that are common in other simulators. For such automation, a user will use (or create their own) managing agent(s) that choose facility deployments using heuristics while satisfying constraints such as a power demand curve.

2.1.1.2 System Dynamics vs Agent-based

Most fuel cycle simulators have traditionally been built as system dynamics models. System dynamics refers to an approach for modeling the behavior of complex systems that was formalized by Jay Forrester [15]. This approach involves approximating the time evolution of a system of differential equations. Due to the nature of system dynamics tools, fuel cycle models usually involve continuous mass flows between fleets of facilities that operate identically. Speaking of the limitations of system-dynamics differential equation based [15] fuel cycle simulators in his report to the Blue Ribbon Commission Wilson, states [41]:

> While this does still allow the study of system dynamics, it requires all facilities to operate with an average behavior and can cause particular difficulty when matching supplies of separated material with desired fuel compositions for advanced reactor systems.

In general system dynamics based tools have difficulty modeling systems with infinite

heterogeneity [3]. After describing a simple system for the adoption of a product from the combined effects of marketing and word-of-mouth, Borschev creates equivalent system dynamics and agent-based models. Then he describes a simple variation:

> Let us now refine our model a bit. Let the word of mouth influence of a particular person (i.e. the fraction of people who will adopt as a result of a contact with that person) depend on how long ago he has purchased the product. We will assume that soon after the purchase the adopter "promotes" the product quite actively, and then this influence goes down and stabilizes at some moderate level...

He then explains the difficulty of adjusting the system dynamics model to accommodate the modification. The difficulty is caused by the infinite possible variation present in individuals' behavior. He finally describes a very inefficient workaround:

> ... a stock containing objects with sufficiently different properties is decomposed into an array of buckets, and objects move between the buckets as their properties change. Consider however objects having not one, but several such properties. The array of buckets grows as a product of dimensions and, after a few steps the number of cells in the array may easily exceed the number of individual objects in the real world we are modeling [27]. This obviously makes such system dynamics model senseless (and terribly slow), while the agent based model will always contain as many objects as needed.

System dynamics based fuel cycle simulators have various practical restrictions on inter-facility interaction, such as the one just described. A concrete example of this would be modeling many individual reactors where each exhibited custom supplier preferences based on things such as quality of past supplied fuel and the recency of past deals with suppliers. Such reactors might also adjust their on-hand fresh fuel inventory based on the

stability/predictability of prior supply. Even simple kinds of cumulative history dependent behavior like this quickly give rise to the challenge described above by Borschev.

Bringing some rigor to the comparison of modeling methods, North proves that for common classes of problems, the asymptotic time and space performance of agent-based models is computationally optimal [25]. He goes on to define 'optimal' to mean that it is not possible for other techniques to use less space or time asymptotically with the number of agents.

Although there are many practical differences between agent-based and system dynamics modeling techniques, North gives a brief overview of an informal proof that all computational agent-based models can be expressed using partial recursive functions (e.g. with system dynamics techniques) [25]. The reverse is also true. However, expressing the complex algorithmic behavior of agents in such functions is often very unnatural and difficult [12].

2.1.1.3 Material Flow

While many fuel cycle simulators focus on nuclear materials (with an isotopic composition) Cyclus agents transact "resources". Resources are a generalized concept of which a nuclear material is one type. In Cyclus, trading carbon credits and property rights would be handled by the simulator transparently requiring no special accommodation. This generalization enables analyses to explore interactions of nuclear fuel cycles with a more real connection to the context in which they live (e.g. competing energy alternatives, political priorities, etc.).

Most fuel cycle simulators generally operate in a continuous material flow mode. Discrete material flow, while empowering to analysis, generates larger quantities of information. This extra information can be used to calculate interesting metrics that are difficult to replicate in continuous flow models. Cyclus' discrete treatment of materials and material exchange allows facility behavior to depend on integral supply/demand of other facilities.

Individual facilities can also choose to favor suppliers that can more singly supply a larger fraction of their fuel needs. Discrete material treatment also enables interesting metrics such as proliferation resistance of facilities' internal material handling procedures and origin-tracking of materials.

The agent-based, dynamic material exchange model of Cyclus also enables material flow connections between facilities' to respond dynamically to the natural evolution of the simulation [17]. For example, a reactor might choose to resell its fresh fuel inventory back into the market if it suffers a long-term disruption. And such a reactor can be used with other facility types that have no knowledge or special support for this behavior. This kind of flexibility enables analysis of policy impacts and other higher order stimuli without requiring support for custom scenarios to be built into the simulator core.

2.1.1.4 Development Philosophy

Cyclus is developed completely in the open with the permissive BSD license. Anyone is free to download it's source code, modify, distribute, and contribute to the project freely. Although this has several advantages, one particularly relevant one for optimization research is the ability to deploy the simulator in large computing environments without license fees or complicated licensing server requirements. Cyclus also has 1st class support for the Linux platform which is very common in the high performance and high throughput computing communities.

A Cyclus simulation produces one output file/artifact: a single database. In addition to material flows and facility histories, the database contains the simulation input file, the Cyclus version and dependency versions, early-termination information, and other information in support of reproducible science. The database is generated in one of a number of robust, open source formats that can be queried, viewed, and analyzed with many well-developed free and open source tools.

2.2 Fuel Cycle Metrics

There are many potentially useful metrics for evaluating the goodness of different fuel cycle options — things that help quantify both the positive and negative effects of nuclear energy. Knowing which metrics to compare depends both on the purpose of an analysis as well as on the capabilities of fuel cycle simulation tools being used. For this work, metrics are the key component of objective functions used for optimization analysis — providing a way to measure the goodness of particular facility/technology deployment strategies. This remainder of this section describes several metrics commonly of interest for fuel cycle analysis generally.

Treating energy (usually electricity) needs as the driving force for nuclear technology use and development (as opposed to something like weapons production), energy production can be considered the sole benefit with several associated (usually undesirable) side-effects or costs. These side-effects include issues related to economics and waste generation among others. Common metrics are used to quantify and compare such side-effects. Because energy production is a primary driver, fuel cycle metrics are often normalized to the amount of energy produced [41].

Basic information provided by fuel cycle simulators commonly includes inter-facility material flows, facility inventories, and times of facility related events (e.g. deployment, disruption, decommissioning, etc.). Most research and literature has built directly on this kind of data with many low level metrics. The bias toward these types of metrics is due in part to the convenience of their calculation. Flicker gives a good overview of this preference [13]. In her survey, safety and other difficult-to-calculate sustainability metrics are underrepresented. Although proliferation resistance metrics are often treated in the reports she analyzed, there is no standard way to measure "weapons usable" that accounts for diverse inter-related factors including: material concentrations, difficulty of use (e.g. radioactivity levels), ease of diversion, etc. A brief survey of some more common metrics used to compare fuel cycle options follows.

2.2.1 Economics

Costs associated with nuclear energy are often more complex to estimate due to the long timescales over which spent fuel must be managed safely. The many costs are often split into a few categories:

- Capital costs of building facilities (including financing)
- Operation and Maintenance costs which include:
  - Fixed costs relating to administration, security, and other expenses that occur regardless of the facility's operational state.
  - Variable costs such as fresh fuel a reactor and other consumables. Variable costs are often split into fuel and non-fuel categories.

When dealing with energy options, a common metric for comparison is the levelized cost of electricity (LCOE). It can be calculated using Equation 2.1 [26].

$$\text{LCOE} = \frac{\sum_{t=1}^{n} \frac{I_t + M_t + F_t}{(1+r)^t}}{\sum_{t=1}^{n} \frac{E_t}{(1+r)^t}} \tag{2.1}$$

Where I_t is the investment expenses in year t, M_t is the operations and maintenance expenses in year t, F_t is the fuel expenses in year t, E_t is the electricity generation in year t, r is the annual discount rate, and n is the lifetime of the system in years.

The Advanced Fuel Cycle cost basis report [37] is a standard reference in the nuclear community that provides cost data and estimates for both current and potential future technologies. The data cover both front-end aspects of the fuel cycle (e.g. mining and milling, conversion, enrichment, etc.) as well as back-end aspects (e.g. interim spent fuel storage, reprocessing, waste conditioning, etc.). Regarding cost uncertainties, Wilson states, "Because most advanced fuel cycle facilities have rarely, if ever, been constructed and deployed in a truly open market setting, there is high uncertainty in most cost projections

for future nuclear technologies" [41]. Attempting to partially address this challenge, the cost basis report's data include cost probability distributions that can be used to estimate total system costs in various forms; Hays does such an analysis in his fuel cycle optimization work [18]. After this fashion, fuel cycle economic analyses are often performed as form of post-processing using facility lifetime and material flow data.

2.2.2 Environmental Impact

Spent fuel from reactors is significantly radioactive and generates non-trivial amounts of decay heat. It is common to look at decay power curves over time for fuel cycles as done by Park in his comparison of four specific fuel cycles [28]. Equation 2.2 shows how to calculate the decay power of a particular material composition:

$$H(t) = \sum_{n \in nuclides} \sum_{d \in daughters} BR_{n,d} \lambda_n N_n(t) Q_{n,d} \tag{2.2}$$

Where H is the material's total decay power, $BR_{n,d}$ is the branching ratio indicating the fraction of decays of nuclide n that decay to daughter d, λ_n is the decay constant for nuclide n, and $Q_{n,d}$ is the energy released from a single atom of nuclide n decaying to daughter d. For decaying materials nuclide composition can be determined as a function of time using the Bateman equations 2.3:

$$N_D = \frac{N_1(0)}{\lambda_D} \sum_{i=1}^{D} \lambda_i c_i e^{-\lambda t} c_i = \prod_{j=1, i \neq j}^{D} \frac{\lambda_j}{\lambda_j - \lambda_i} \tag{2.3}$$

N_D is the number of atoms of isotope D in a decay chain, D is the number of decays in the decay chain, N_1 is the initial population of the chain's first isotope, λ_i and λ_j are the decay constants for isotopes i and j in the decay chain, and t is the time since time zero. Decay heat curves are sometimes collapsed into a single, integrated decay heat over various time horizons (typically 100s to 1000s of years) as done by Hays in his deployment optimization analysis [18].

Nuclides in spent fuel are often characterized by the magnitude of their biological impact (e.g. on humans). Each nuclide is given a weight or coefficient - often from a standard such as ICRP 72 [33]. Coefficients exist for different circumstances including inhalation and ingestion. Toxicity profiles are typically calculated like Equation 2.4:

$$J(t) = \frac{1}{E_{tot}} \sum_{n \in nuclides} N_n(t) w_n \tag{2.4}$$

Where $J(t)$ is the toxicity (ingestive, inhalation, etc.), E_{tot} is the total energy produced from fission, $N_n(t)$ is the number of atoms of nuclide n at time t, and w_n is a toxicity coefficient (ingestive, inhalation, etc.). Eastham et al. [9] do such a calculation using both time-integrated and peak toxicity as metrics in their analysis.

Various resource utilization metrics have been used quantify the nuclear industry's burden on the environment. A common metric in this category used by both Hays [18] and Park [28] in their analyses is a simple ratio of the mass of natural uranium used to the amount of energy produced (e.g. electricity). Another variation used by Eastham [9] measures uranium utilization as the fraction of available fission energy (including fertile isotopes) utilized before disposal. Other, less common metrics include various land and water usage metrics - also usually normalized to energy production.

2.2.3 Non-proliferation

Most proliferation-related metrics attempt to estimate how difficult it would be to convert the material made available at various points in a fuel cycle into a weapon of some kind. Because this is a somewhat ill-defined problem, available metrics all make many assumptions and generally do not attempt to treat the human factor (e.g. opportunity, motives, etc.). Bare sphere critical mass is perhaps one of the simplest metrics defined in Equation 2.5 [9]:

$$M = \frac{4\pi^4 \rho}{3}\left(\frac{D}{\nu\Sigma_f - \Sigma_a}\right)^{\frac{3}{2}} \tag{2.5}$$

Where ρ is the fuel density, Σ_f and Σ_a are the fission and absorption cross sections respectively, and ν is the average number of neutrons released per fission. Eastham also uses a more sophisticated figure of merit (FOM) that attempts to characterize the difficulty of handling the spent fuel due to its radioactivity (Equation 2.6):

$$FOM = 1 - \log_{10}\left(\frac{M}{800} + \frac{Mh}{4500} + \frac{M}{50}\left[\frac{G}{500}\right]^{\frac{1}{\log_{10} 2}}\right) \tag{2.6}$$

In this FOM, M is the bare sphere critical mass, h is the decay heat (W/kg), and G is the gamma ray dose rate (rad/hr) from a sample of size $0.2M$ evaluated at 1 m from the surface. Metrics like this can be used in different ways to characterize fuel cycles such as:

- Using the peak FOM over various time horizons.
- Measuring how all or parts of the fuel cycle change the FOM of different material streams.

2.3 Optimization Techniques

Using a fuel cycle simulator as part of an optimization objective is a challenging problem for several reasons:

- The objective function is generally not a linear function of input parameters.
- No derivative information for the objective is available.
- The objective function may be discontinuous.
- Input variables to the objective function may be discrete.

- The input parameter space could be large necessitating constraints for reasonable performance.
- The objective function may be stochastic - i.e. different runs with the same inputs may produce different outputs.
- The objective function is expensive to evaluate - seconds to minutes for a single iteration.

This kind of problem is often referred to as black-box optimization. Another challenge is desirability of the optimization process to be effective over many different simulation scenarios. There is no single, well-defined objective to be optimized.

2.3.1 Algorithms

There are several classes of algorithms for attacking black-box optimization problems. An overview of some of the more common algorithms follows.

2.3.1.1 Pattern Search

Pattern search covers a large class of algorithms that are well studied and used in optimization (i.e. prevalent in Matlab). While there are many variations, most pattern search algorithms roughly follow this basic process:

1. Define a discrete mesh over the entire input parameter space.
2. Select an initial point on the mesh and evaluate the objective function there.
3. Perform a finite number of evaluations of the objective function at points on the mesh. This is a generic step and the points can be selected using any method. This is referred to as the "search" step. If a better point is found:
    - Optionally coarsen the mesh.

- Go back go the beginning of the search step and repeat.

Otherwise, continue to the next step.

4. Evaluate the objective function at neighboring points in the mesh to the current best point. This is referred to as "polling". Optionally expand the mesh size if the poll was successful (a better point was found). Contract the mesh if the poll was unsuccessful.

5. Repeat from the search step until desired convergence criteria are met (often when the step size between mesh points drops below some threshold).

For many classes of problems (usually continuous and differentiable) pattern search algorithms have been shown to converge well to stationary points. The polling behavior together with mesh contraction provide this convergence property. This property is somewhat unique among global, derivative-free optimization algorithms.

There are many variations of pattern search that have been studied. Because of its flexibility evolutionary and other optimization algorithms are often used as part of the search step. This hybridization with other global optimization algorithms is an approach that has been studied and implemented in various optimizers such as PSwarm [39]. Mesh adaptive search algorithms [2], for example, improve on basic generalized pattern search by using more sophisticated polling direction calculation, providing better handling of nonlinear constraints among other things. Techniques for effectively handling linear constraints have been investigated by Lewis [23] and others.

2.3.1.2 Swarm Algorithms

Among the various swarm algorithms that have been developed, the particle swarm and ant colony algorithms are likely the two most common. Particle swarm algorithms roughly follow this basic process:

1. Select an initial population of particles (i.e. points in the input parameter space). Each particle is assigned a velocity in addition to its position. Initial velocities are often generated randomly with reasonable bounds. The objective function is evaluated at each particle's position. Particles often keep a history of the best positions they have occupied.

2. Select a leader strategy. A particle knows about the best positions discovered by its "leader" particle(s). A particle can have many leaders. While there are many possible leader topologies, the most common is the star topology, where all particles are leaders for all other particles (a fully connected graph). Each particle knows about the entire swarm's best positions [20].

3. Update each particle's velocity using a combination of:

    - the particle's current velocity
    - the best position(s) found by the particle
    - the best position(s) found by the particle's leader(s)
    - constraints

4. Update each particle's position using its velocity and evaluate the objective function at all particles' new positions.

5. Repeat from step 3 until desired convergence criteria are met.

Particle swarm algorithms have been studied extensively for nearly two decades since their formal introduction by Eberhart and Kennedy [10] in 1995. Given their effectiveness, particle swarm algorithms are surprisingly simple to implement. One such implementation is PSwarm. Various mechanisms for dealing effectively with linear constraints include both hard and soft constraint enforcement. Examples include:

- Damping particle velocity near constraint boundaries [38]. This is hard constraint enforcement maintaining particles inside the feasible region. One possible variation might include per-variable damping based on distance to the most violated constraint plane.

- Using a modified fitness function to elect leader particles and best positions. [31]. This is an example of soft constraint enforcement allowing particles to move outside the feasible region.

Convergence properties of particle swarm optimization have also been studied by Clerc and Kennedy [7].

2.3.1.3 Evolutionary Algorithms

The basic principles of genetic algorithms are fairly simple:

1. Choose an initial population of solutions and evaluate the objective function at each point.

2. Select solutions from the population for crossover. Selection techniques usually take into account the solutions' objective values.

3. Combine selected solutions to generate new child solutions - referred to as crossover or recombination.

4. Select and mutate portions of certain solutions.

5. Select the next generation of solutions from among all parent and child solutions and repeat from step 2.

Although the principles are simple, genetic algorithms often have many parameters, and a proper balance of selection, crossover, and mutation is crucial to achieving satisfactory optimization performance.avoiding premature convergence. Discussing the significance of

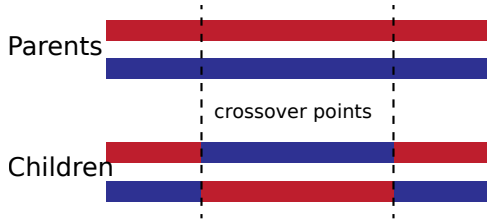

Figure 2.1: Two point crossover [32].

mutation, Whitley states "Mutation, therefore acts as a background operator, occasionally changing bit values and allowing [forgotten alleles] to be retested" [40]. Noraini compares the performance of 3 more common selection methods on a traveling salesman problem [24]:

- Tournament selection
- Proportional roulette wheel selection
- Rank-based roulette wheel selection

He finds that rank-based roulette selection is consistently the most effective followed by tournament selection and then proportional roulette selection. Crossover methods are also numerous and include varying numbers of parents per offspring, different granularity of mixing parents' genes, resequencing genes based on their interdependence, and other things. The most basic crossover method is N-point crossover consisting of aligning two parents' genes, randomly choosing N points, and swapping every other segment to generate two new children. A simple example of 2-point crossover is shown in figure 2.1.

Due to the crossover operation present in many evolutionary algorithms, it is difficult to guarantee properties of generated child solutions with respect to constraints. Because of this, most methods use a constraint penalty rather than using any kind of strict feasibility enforcement. There are two basic types of penalties:

- interior: the penalty function generally small in the feasible region interior but grows very large near the boundary of the feasible region.

- exterior: the penalty function is very small (often zero) in the feasible region and proportional to the level of constraint violation outside the feasible region.

Exterior penalties are more common because they don't require initial feasibility of solutions. Although penalty techniques are the most common, several others that have been investigated in research. Coello surveys many techniques [8] including:

- Problem specific, custom crossover and mutation operators design preserve solution feasibility.
- Problem specific, custom repair algorithms that recover feasibility in generated solution.
- Self-adaptive penalty functions in which the weight or significance of a constraint changes based on how often/much it is violated.

2.3.1.4 Surrogate Based Techniques

Surrogate based optimization is a technique where an approximate model (or surrogate) of an objective function is used to improve overall optimization performance. This surrogate can be built prior to optimization, or it can be constructed incrementally as optimization proceeds. The surrogate model can then be used to inform the optimizer's exploration of the variable space with promising points. Types of surrogate models include polynomial response surfaces, Kriging, support vector machines, and artificial neural networks. Many techniques assume the objective function is continuous and smooth. Kriging is one of the more accurate surrogate techniques, although it tends to suffer from computational complexity issues for higher dimensional problems [14]. Surrogate-based optimization often roughly follows the following process:

1. Sample the actual objective function over an initial set of points.
2. Construct/update a surrogate model using sampled objective function points.

3. Search the surrogate model for good points using desired techniques (e.g. derivative-based techniques, evolutionary algorithms, pattern search, etc.)

4. Sample the actual objective function using points found from searching the surrogate.

5. Repeat steps 2 through 4 until done.

2.3.2 Multi-objective Optimization

It is often desirable to investigate problems treating multiple objectives individually - to be optimized simultaneously. This kind of optimization is referred to as multi-objective optimization. While there exist various techniques there are a few basic concepts used by most methods in the case of derivative free or black-box optimization. Solutions are generally compared using the principle of Pareto optimality. For every evaluation point (i.e. solution), each objective function is evaluated individually. A solution is considered Pareto optimal (also "non-inferior" or "non-dominated") if no single objective value can be improved without a corresponding decrease in one or more of the other objectives. A solution "dominates" another solution if it has better values for all objectives. Optimization problems that have competing objectives do not have a single optimum solution. Rather, they have a set of Pareto optimal solutions that comprise a trade-off surface or Pareto front between the objectives.

Optimization methods with an interacting population of solutions are particularly well suited for approximating Pareto fronts. As such, much effort in (black-box) multi-objective optimization has been directed toward evolutionary algorithms [11]. There also exist established multi-objective techniques for particle swarm [34] and simulated annealing optimization.

2.3.3 Implementations

Recently, a relatively thorough comparison of optimizers' performance was done by Rios and Sahinidis [35]. In their paper, several optimizers are analyzed including TOMLAB solvers, DAKOTA framework solvers, MCS, SNOBFIT, PSwarm, Matlab's fminsearch, and others. The analysis includes over 500 test objective functions that span convex to non-convex, 10 to 300 variables, and smooth to non-smooth. All test problems were bound-constrained only - no linear or nonlinear constraints. All solvers were limited to a maximum of 2500 function evaluations. The results focus on the number of problems solved by each optimizer to within either 1% or 0.01 (whichever is larger) of the known true optimum. So a solver that achieves 95% of the optimum receives no credit. The fixed 2500 function evaluation limit also makes it difficult to draw conclusions for higher dimensional performance of algorithms; higher dimensional problems should certainly require more evaluations for suitable convergence from any solver.

The TOMLAB solvers were consistently the most effective. However, they are non-free proprietary optimizers and also require a separate Matlab license to run making them less desirable and less suitable for large-scale computing. Other strong performers include MCS, and SID-PSM, and CMA-ES. For the non-convex non-smooth functions, all of the solvers struggled, although PSwarm jumps up much higher in the rankings than for the other function categories (convex smooth, convex non-smooth, etc.). Although not part of the general results discussion, the paper contains several charts showing the mean, median, best, and worst points over 10 runs found by each of the optimizers as a fraction of the known optimum for the following categories of objective functions:

- convex and smooth
- convex and non-smooth
- non-convex and smooth

- non-convex and non-smooth
- 1 to 2 variables
- 3 to 9 variables
- 10 to 30 variables
- 31 to 300 variables

These charts give a slightly different perspective of the optimizers' relative performance. Viewed this way, PSwarm and DAKOTA DIRECT are strong contenders along with the TOMLAB solvers and MCS, achieving values within a few percent of the true optima for nearly all runs.

DAKOTA [1] is a large open source optimization framework written in C++. It contains many different solvers ranging from gradient based methods to pattern search to evolutionary algorithms. DAKOTA provides a unified interface onto several different solver types. Some of the solvers are actually wrappers around other open source solvers such as NOMAD and HOPSPACK - both of which were a part of Rios and Sahinidis' derivative free optimizer comparison analysis [35]. In addition to being actively maintained, DAKOTA also provides tools for uncertainty quantification and sensitivity analysis. One of the optimizers, JEGA, has native support for multi-objective optimization for approximating Pareto fronts [11] (i.e. trade-off surfaces between competing objectives). DAKOTA also contains support for Kriging and other surrogate based optimization. Support in solvers for parallel execution, discrete variables, and constraints both linear and nonlinear are available in various combinations.

PSwarm [39] is a hybrid optimizer that uses pattern search and particle swarm algorithms. A simple particle swarm algorithm is used as the search portion of pattern search. Particle evaluations are projected onto the pattern search mesh. Notably, PSwarm has good support for linear constraints [38]. The maximum volume inscribed ellipsoid technique is

used to generate a divers initial feasible population. Feasibility is maintained in the search step by damping particle position changes according to the most limiting constraints when near a boundary. Feasibility is maintained during polling by generating poll directions using a tangent cone for constraints that are "active" (i.e. close to being violated). PSwarm is open source and is implemented in both Matlab and C and also has bindings for Python and R.

2.4 Fuel Cycle Optimization

Historically, much of fuel cycle analysis and optimization has been performed on equilibrium or steady-state fuel cycles - meaning that the kinds of technologies deployed and the numbers and ratios of facilities is not changing (and hasn't for a long time). There has also been little formal optimization. Rather, research has often focused on examining hand-picked fuel cycles or performing parameter sweeps and comparing realizations. The following few paragraphs summarize representative research along these lines.

In their 1987 paper, Kunsch and Teghem [21] performed a multi-objective linear programming optimization over equilibrium fuel cycles. Their objectives to be minimized were: fuel cycle cost, natural uranium consumption, commercial balance, and employment. Their analysis includes parametric comparisons of the trade-offs between different objective values.

Park et al. [28] compare four hand-selected steady-state fuel cycles using simple metrics covering uranium utilization, proliferation potential, and waste characteristics. The four scenarios use combinations of pressurized water reactors (PWR), CANDU reactors, and sodium fast reactors (SFR). Three types of recycling are also investigated: direct use of spent fuel in CANDU reactors, mixed oxide recycling for PWRs, and pyro-processing [22]. No particular simulation tool was used. Rather, reactor and reprocessing characteristics (e.g. material in/out flows) were obtained using deployed instances of the facilities and/or

from design studies.

Eastham et al. [9] perform a parametric analysis over equilibrium fuel cycles focusing on material transformations occurring in reactors. A reactor model is developed that operates on actinide populations of fuel material. Front and back end aspects of the fuel cycle are not modeled. Thorium cycles were simulated using 6-20% enriched ^{233}U as input fuel for the reactor model, and more standard uranium cycles were simulated using 2-10% enriched ^{235}U. For the parametric sweep over input fuels, several metrics were tallied such as fission energy utilization fraction, minimum waste critical mass, and fuel-waste toxicity ratio.

2.4.1 Dynamic Deployment Optimization

Much less research and optimization work exists for dynamic or transition fuel cycle scenarios. Perhaps one of the most relevant studies was performed recently by Hays using the VISION fuel cycle simulator [18]. Hays performs a multi-objective deployment optimization analysis using the using a custom coded simulated annealing algorithm. Non-inferior solutions (approximately Pareto optimal) are recorded and used to approximate Pareto fronts or trade-off surfaces between combinations of 3 objectives:

- Cost of electricity per MWh.
- Long-term decay heat integrated from 100 to 1500 years after fuel is discharged from reactors and normalized to the total amount of electrical energy produced.
- Mass of uranium mined normalized to the total amount of electrical energy produced.

The input variables for the objective function are the fraction of new reactors deployed annually that are fast reactors for each year in a 100 year VISION simulation. The amount of new deployed electrical capacity each year is determined by a fixed power demand curve. This analysis is a 100 dimensional, continuous variable, bound-constrained, derivative-free optimization problem. Additionally, some forecasting heuristics are used to reject and

avoid evaluating solutions that are likely to be undesirable or infeasible. Termination of the simulation before all facilities have reached end-of-life can result in biased objectives - particularly those related to economics. In order to combat this bias, Hays chose to run simulations an additional 100 years with no new facility deployments until all fuel had reached its final disposition state.

Each optimization run involved 7000 to 16000 simulations in an ad-hoc 4-node windows cluster. In his results, Hays notes an undesirable turnover in solutions along the Pareto front that occurs due to internal deployment heuristics in the VISION model. Separations capacity is built to meet maximum annual demand rate. Fast reactors have a larger fuel demand when they are first built than for the remainder of their life. This artifact is a manifestation of internal assumptions encoded in the model that become active only for certain inputs to the objective function.

Forecasting techniques to optimize deployment have also been studied. This generally takes the form of simulation-internal optimization. Schweitzer performs deployment forecasting analysis [36] using the VISION fuel cycle simulator. Rather than use optimization tools/algorithms, parameter sweeps were done to identify optimal lead times to separations facilities in a recycle scenario with both LWRs and fast reactors. He reports optimal separations facility deployment lead times as a function of 3 separations facility sizes (capacities) and 3 fast reactor deployment patterns. A LWR spent fuel inventory forecasting algorithm determined facility deployment times. Optima were approximately identified by comparing the closeness of predicted inventory curves (from the forecasting algorithm) to actual inventory curves for LWR spent fuel.

In the same paper, Schweitzer also performs a few measurements quantifying lost GWe-years of energy from fast reactors being unable to operate due to fuel supply disruptions. He concludes by noting that increasing separations facility lead time appears to be a more effective mitigation strategy for dealing with potential separations facility disruptions.

Other transitional deployment optimization work has been performed by Passerini

[29] using separately both a hill climbing algorithm and an evolutionary algorithm solver with the CAFCA fuel cycle simulator. With the evolutionary solver, he performed both single and multi objective analyses. Passerini's analysis focuses on finding optimum times for the introduction of recycling technology based using one or more objective functions constructed from several criteria including:

- Levelized cost of electricity.
- Construction of various reactor types (minimize or maximize).
- Total system cumulative high level waste inventory.
- Total natural uranium usage.

He used two input variables for the objective function:

1. Thermal reprocessing introduction date.
2. Fast reactor technology introduction date.

If discretized to an annual basis, these two independent variables can each take on about 100 values resulting in a parameter space with only 10,000 total combinations. Each optimization run with the hill climbing and evolutionary algorithms performed on the order of 500 hundred function evaluations. For the single objective evolutionary analysis, each sub objective was converted to a dimensionless quantity by normalization to the objective's value in a reference scenario. Then the final objecive was computed as a weighted linear combination of each sub objective; weights used were informed by an expert elicitation. Results from this analysis yielded at least one scenario with multiple local optima where the hill climbing algorithm failed to converged to the global optimum. The multi-objective analysis was used to estimate Pareto fronts between the individual sub objectives.

2.5 Summary

Many different tools have evolved over time to help address the need to investigate both the equilibrium properties of nuclear fuel cycles and the dynamics of transitions between them. These tools, while very useful, have various limitations that motivated the development of the Cyclus fuel cycle simulator. Cyclus' properties make it suitable for extending the types of fuel cycle analysis that can be performed. Two such avenues are optimization work made possible by support for deployment to large-scale high-performance computing environments and the investigation of effects of modeling choices/fidelity within fuel cycle scenarios made possible by Cyclus' dynamic resource exchange and plugin architecture.

While optimization has been used to augment fuel cycle analysis in the past in a couple of cases, prior work has been limited in scope — partially due to available simulation tools. Fuel cycle optimization has mostly been limited to investigating equilibrium properties of fuel cycles (e.g. not accounting for the dynamics of transitioning from the current U.S. LWR reactor fleet to a fleet some kind of advanced reactors). For cases where transitions have been investigated, things such as deployed power generation capacity over time and deployment of non-reactor support facilities have been a fixed component of the scenario rather than allowing optimization to include these decisions the automated search for good solutions. Also, tooling and infrastructure for utilizing custom objectives and deploying the optimization to diverse, large-scale computing environments has rarely, if ever, been developed and shared openly. These and other limitations of previous work are great opportunities for increasing the capability and utility of fuel cycle analysis.

There exist many fuel cycle simulators built with many different combinations of modeling choices and assumptions. This makes comparing results from them difficult. The flexibility and features provided by the Cyclus simulator provide a rich playground for comparing the effects of such modeling choices in a consistent way. Such investigation can help improve the quality and consistency of fuel cycle analysis codes in addition to increasing confidence in the value provided by fuel cycle analysis generally.

Developing and deploying new nuclear technologies is a long and complicated process in the real world. There is significant uncertainty associated with many things such as time-frames for technology availability, cost of deploying advanced reactor technologies, and ever-changing policy and regulation. Historically, fuel cycle analysis has focused on answering questions of fuel cycle feasibility and, to a lesser extent, questions of optimality. To date, however, there has not been much work done to address this uncertainty in fuel cycle analysis. Such work could help extend the reach of fuel cycle analysis toward answering questions of fuel cycle robustness — questions that become very important when planning for the future which is arguably one of the primary purposes of fuel cycle simulation and analysis.

3 EXPERIMENT 1: OPTIMIZATION STRUCTURE AND COMPARISON

The purpose of this experiment is to design an effective technique for determining desirable deployment schedules for fuel cycle scenarios automatically using optimization. Although generating a detailed fuel cycle scenario and investigating properties of discovered deployments are important parts of this exercise, the two most important foci are:

1. Formulating objective function input variables in an efficient, advantageous way (Section 3.1).

2. Comparing optimization algorithms (Section 3.2).

The following few sections describe the details of the experiment (e.g. optimization structure, optimizers for comparison, scenario details, objective function, etc.). After the experimental methods have been covered, results for the optimizer comparison are presented at the end in Section 3.5 with limited discussion of features of the best discovered deployment schedules.

3.1 Optimization Structure

At the most fundamental level, the optimization structure consists of an optimizer choosing several variables that together identify a set of facility deployments through time. These deployments are used to run a simulation with Cyclus. The resulting simulation database is interrogated and its data are used in metric calculations to compute a single, scalar objective value that represents the desirability of the deployment schedule. This objective value is communicated back to the optimizer which then uses the feedback to choose more deployment schedules — eventually iterating toward an approximately optimum set of facility deployments.

In overall structure this optimization analysis builds directly on previous work by Carlsen [5] and is similar to some work done by Ross Hays [18]. Primary differences are in the input variable definitions and in the objective function details. Hays' analysis was primarily multi-objective, where this work instead focuses on a single objective function. Here, the optimizer chooses the numbers and times of all reactor facility deployments in each Cyclus simulation.

Many black-box optimizers do not perform well when provided explicit constraints and often resort to using constraint violation penalties. More sophisticated constraint handling is difficult and many black-box optimizers do not support any type of constraints beyond upper and lower variable bounds. In previous optimization work done with Cyclus, each objective function variable represented an integer number of facilities of a specific type to deploy in a specific time step. Extensions of the analysis done by Carlsen [5] suffered from the constraint-related challenges just mentioned. To combat these challenges, a new way to define optimization variables is developed and presented here.

The simulation time horizon is divided into annual deployment periods on which new reactors are constructed. Each scenario has upper and lower bounds for power demand, and the optimizer must maintain deployed power capacity within these bounds. Objective function input variables are divided into two semantically different types for each deploy period:

1. One *new power capacity* variable ($V_{power}(t)$). This identifies where capacity will be built up to in the range between upper and lower power bounds.

2. One or more reactor type capacity fraction variables ($V_{facac}(t, r)$) representing the fraction of new capacity to be satisfied by deployments of each reactor type. A variable for the last reactor type is omitted and the value is set to the remaining, unsatisfied power capacity fraction.

The variables are designed to be box-bounded between zero and one. The algorithm

for computing power variables (type 1 above) is described in Equation 3.1. Computing the new power capacity of each variable, the fractions of each reactor type, and other facility deployments for time t requires usage of the full computed build schedule prior to time t. Although the transformation is described in separate pieces (Equations 3.1, 3.2, and 3.3), the actual full transformation requires all components to be computed in parallel, iterating through each deploy period chronologically.

$$P_{new}(t) = max\{0, V_{power}(t) \cdot max\,[0, P_{max}(t) - L(t)] + L(t) - P(t^{-})\} \tag{3.1}$$

Equation 3.1 computes $P_{new}(t)$ - the new power capacity that will be deployed on time t. $L(t) = max(P(t^{-}), P_{min}(t))$, $P_{min}(t)$ and $P_{max}(t)$ are the minimum and maximum power capacity that must be operating on time t respectively, $V_{power}(t)$ is the power input variable value for time t, t^{-} represents time t before any new deployments occur on the time step. Values of the included variables and functions (e.g. $P(t)$) are based on the build schedule as computed up through time t.

$$N(t,r) = \begin{cases} floor\left(\dfrac{V_{fac}(t,r) \cdot \left[P_{new}(t) - \sum_{r'=1}^{r-1} N(t,r') \cdot C(r')\right]}{C(r)} + 0.5\right) & : r > 0 \\ floor\left(\dfrac{P_{new}(t) - \sum_{r'=1}^{r_{last}} N(t,r') \cdot C(r')}{C(r)} + 0.5\right) & : r = 0 \end{cases} \tag{3.2}$$

$N(t,r)$ is the number of reactors of type r to deploy at time t, $floor(x)$ is the closest integer to x that is less than or equal to x, $V_{fac}(t,f)$ is the input variable value for facility f at time t, $P_{new}(t)$ is defined in Equation 3.1, and $C(r)$ is the power capacity for a single reactor of type r (e.g. 1000 MWe). The $N(t, r = 0)$ deployments must be computed after all $N(t, r > 0)$ deployments. Reactor type $r = 0$ is used to deploy any remaining new power capacity that is not satisfied by the other reactor types. Computed $N(t,r)$ values become

part of the build schedule underlying computations in in Equations 3.1 and 3.2.

Deployments for facilities that do not contribute to power (or other) capacity can also be represented. For these facilities, the optimization variables represent a fraction of the number of dependent capacity generating facilities (e.g. reactors) for a particular time step. The dependence relationship can be customized. For example, we might want to deploy enrichment facilities as a fraction of the number of deployed pressurized water reactors and boiling water reactors. Equation 3.3 is used to compute deployments of facilities that do not contribute to power capacity (e.g. everything except reactors):

$$N(t,f) = max\left[0, floor\left(V_{fac}(t,f) \cdot \sum_{r \in r_f} N(t,r) + 0.5\right) - N(t^-,f)\right] \tag{3.3}$$

where r_f represents the set of capacity-generating (e.g. reactors) facility types that facility type f's deployments depend on and $N(t^-,f)$ represents the number of facilities of type f that are currently deployed and operating before new deployments of type f are computed/performed for time t. Note that $N(t,f)$ depends on the reactor deployments for time t and must be computed after them.

This structuring of the optimization variables allows relationships between reactor types and power capacity constraints to be represented implicitly in the definition of the variables. All combinations of variable values represent meaningful deployment schedules that lie within the region of interest. This not only enables optimizers without constraint support to be used, but it also prevents wasting computational resources searching through many infeasible deployment schedules.

Capability for transforming deployment schedules back into variable values has also been developed. In addition to aiding debugging and validation, this allows a build schedules from one scenario to be used to seed alternate scenario optimizations (e.g. scenarios with different power bound constraints). The reverse transformation is designed to allow build schedules to be successfully round-tripped (i.e. from build-schedule to

variable values to build schedule) without information loss/mutation. Converting from variable values to a build schedule and back to variable values again, however, does not generally result in the same variable values. Because the underlying input space is actually discrete, there the reverse transformation (from build schedule to variable values) generates only one of many possible variable sets that are all correspond to the same build schedule.

3.2 Optimizer Selection

A few optimizers will be chosen and compared for their performance solving problems similar in structure to those that will be used later in Experiments 2 and 3. In the interest of keeping all tools used for this research free open, only free and open source optimizers are considered; this excludes Matlab based optimizers since Matlab itself is neither free nor open. Optimization runs are conducted using the HT Condor infrastructure at University of Wisconsin - Madison. Optimizer selection for this experiment was guided by the following rough requirements:

1. Support for hundreds of variables.
2. Highly parallelizable.
3. Robust in non-deterministic and/or flaky environments (e.g. network issues causing failed objective evaluations).
4. Configuration and setup complexity.

Sandia's open source DAKOTA framework provides a large collection of optimizers available through a common interface and is the source of all optimizers for this analysis except one. The one exception is an optimizer I created specifically for this work based on the PSwarm optimizer described previously in Sections 2.3.1 and 2.3.3. Optimizers considered and rough characteristics are shown in Table 3.1.

	1. Max # vars	2. Parallelism	3. Robustness	4. Configuration
Custom PSwarm	Unlimited	Good	Good	Easy
JEGA	Unlimited	Good	Good	Tricky
SCOLIB EA	Unlimited	Good	Good	Tricky
NCSU DIRECT	64	Poor	Okay	Easy
SCOLIB DIRECT	1000s	Great	Okay	Easy
NOMAD	<=1000	Poor	Good	Easy
HOPSPACK	Unlimited	Great	Good	Easy
Surrogate methods	10s to 100s	Good	Poor	Tricky

Table 3.1: Candidate optimizer characteristics.

Work on these deployment schedules will require optimizers to handle problems with a few hundred variables. Although the variables are continuous, they are actually mapped to an underlying discrete domain resulting in non-smooth jumps in objective values. The objective function is also likely non-convex. Rios et al. [35] compare several well known optimizers including a few from the DAKOTA framework. Notably from their work, the differentiation in performance between the optimizers is much less pronounced for problems of this form and size. PSwarm, evolutionary, and DIRECT solvers from DAKOTA are included in their analysis. The solvers selected for this comparison exercise are:

1. DAKOTA JEGA single objective solver (JEGA).
2. DAKOTA SCOLIB evolutionary solver (EA).
3. DAKOTA SCOLIB DIRECT solver (DIRECT).
4. DAKOTA HOPSPACK solver (pattern).
5. Custom PSwarm implementation (PSwarm).

3.3 Scenario Description

The scenario starts with 100 LWRs and follows an exponential curve with a 1% annual growth rate for 200 years. The decommissioning of the initial 100 reactors is staggered

over years 15 to 55. There are two facility types for which the optimizer chooses deployments: enriched-uranium-fueled light water reactors (LWR) and recycled-plutonium-fueled sodium fast reactors (SFR). Deployments are annual unless otherwise noted resulting in a 400-variable optimization problem: one variable for total power level each year and one variable for SFR deployment fraction each year.

Enrichment feeds LWRs with fresh fuel with infinite capacity. Fast reactors become available to build in year 35. LWR spent fuel separations begins in year 15 with 2000 MTHM/yr capacity and increases to 3000 MTHM/yr in year 25. Fast reactor spent fuel separations and fuel fabrication have infinite capacity. All spent fuel is stored/cooled for 7 years before it is available for separations. Figure 3.1 shows the material flow relationships between the facilities. Compositions used for the simulations were simple, containing primarily ^{238}U, ^{235}U, ^{239}Pu, ^{241}Am, and small quantities of a few lanthanides.

Cyclus version 1.3 was used for this analysis. Facility models used were the standard ones provided with Cyclus in the Cycamore library (also version 1.3). A custom storage facility was used because Cycamore does not yet provide one. A custom fleet-based reactor was also used for both reactor types rather than modeling facilities individually with the Cycamore reactor primarily for performance reasons. The fleet reactor has no sophisticated physics functionality and instead just transmutes all fresh fuel into a fixed spent fuel composition just like the Cycamore reactor. Additional parameters for the reactor types are shown in Table 3.2.

	LWR	FR
Lifetime (yr)	80	80
Cycle length (months)	18	14
Batch size (kg)	29565	7490
Batches per core	3	5

Table 3.2: Reactor facility parameters.

The optimizers were given the ability to deploy power capacity anywhere within $\pm$ 10% of the 1% growth curve. Three separate optimization runs were performed for each opti-

mizer using different random number generator seeds for each run. The maximum number of function evaluations for each optimization run was set to 60,000.

3.4 Objective Function

The objective function was designed to drive toward a fast-reactor-only fuel cycle as quickly as possible while simultaneously discouraging/penalizing unfueled reactors. Because fast reactors are fueled only with recycled fissile material, it is possible for some deployment schedules to cause fast reactors to idle without fuel. The objective function used is shown in Equation 3.4.

$$O_{sim} = \frac{\sum_{t \in sim} E_{t,\,LWR}}{\sum_{t \in sim} E_{t,\,tot}} \tag{3.4}$$

O_{sim} is the objective function value for an entire simulation, $E_{t,\,LWR}$ is the energy produced by all light water reactors (LWR) in time step t, and $E_{t,\,tot}$ is the energy produced by all reactors in time step t. Because the optimizer is trying to minimize the objective value, more LWRs results in a larger numerator (a worse objective value), and unfueled fast reactors shrinks the denominator (also a worse value). It is notable that this objective function does not penalize unfueled reactors very heavily relative to the penalty for LWR energy. This is intentional and allows the optimization to explore potentially interesting trade-offs related to expediting the transition.

3.5 Results

The overhead and time involved in dispatching and running Cyclus simulations is much larger than the time and computational resources required to run the each optimizer itself. Because of this, the optimizers are compared by primarily looking at the best objective function value achieved as a function of the total number of objective function evaluations.

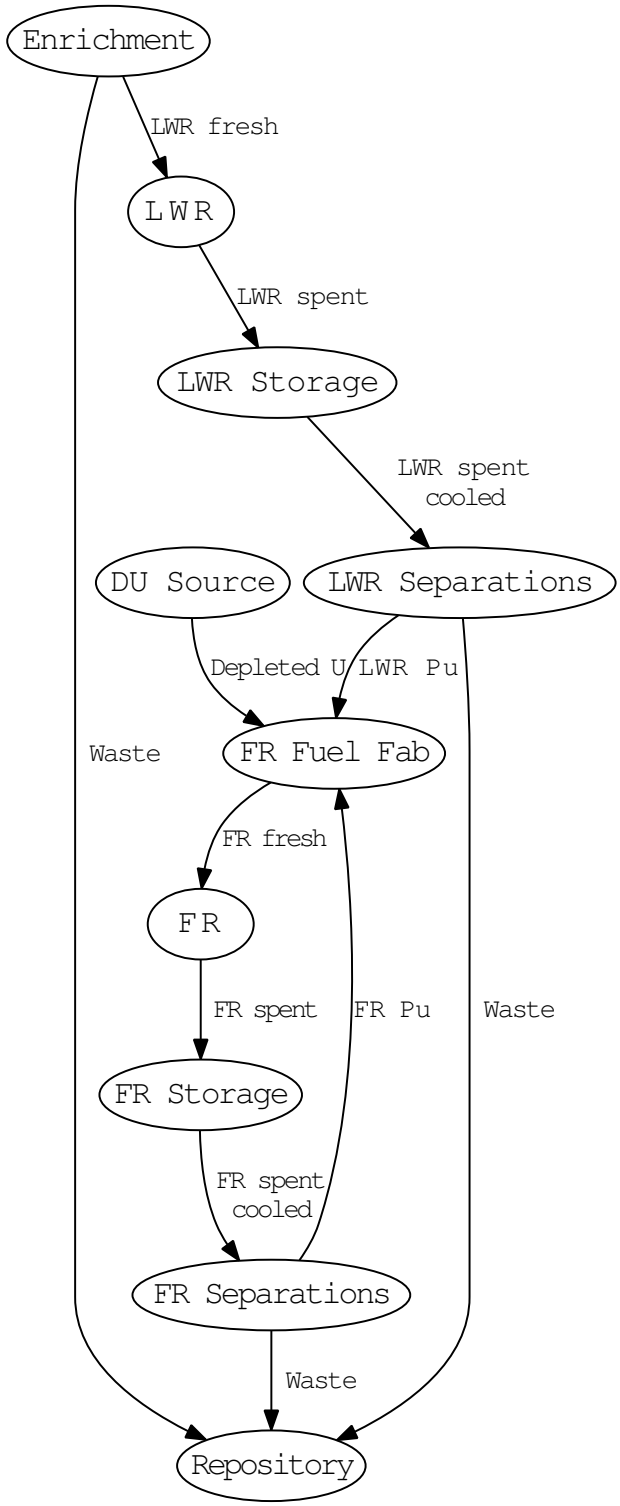

Figure 3.1: Potential material flow paths between facility types for Experiment 1.

Table 3.3 summarizes the results for all runs done with each optimizer. Only a single run was performed with the DIRECT algorithm because it is deterministic and multiple runs on the same problem will always give the same result. The optimizers are listed in order of performance/effectiveness with PSwarm being the best performer and DIRECT being the least effective.

	Run	# evaluations	# iterations	Best objective
PSwarm	1	60,046	356	0.1704
PSwarm	2	60,062	349	0.1670
PSwarm	3	60,153	349	0.1671
PSwarm-biannual	3	62,022	368	0.1423
JEGA	1	28,029	246	0.1884
JEGA	2	59,203	518	0.1848
JEGA	3	26,821	236	0.1889
HOPSPACK	1	60,012	310*	0.2816
HOPSPACK	2	59,840	310*	0.2637
HOPSPACK	3	59,956	310*	0.2900
SCOLIB EA	1	60,048	402	0.4742
SCOLIB EA	2	60,048	402	0.4871
SCOLIB EA	3	60,048	402	0.4871
SCOLIB DIRECT	1	58,260	16	0.5531

Table 3.3:
Experiment 1 summary results for all optimizer runs. The additional *PSwarm-biannual* run is identical to the other PSwarm runs except it had bi-annual instead of annual deployments resulting in only 200 optimization variables.

* The HOPSPACK optimizer did not have well-defined iteration boundaries.

The PSwarm and JEGA solvers were the strongest performers, with PSwarm beating out JEGA slightly in both convergence rate and best solution found. The other solvers performed unsatisfactorily. Figure 3.2 provides a good picture for roughly evaluating how each optimizer performed with respect to the following criteria:

1. How good was the best objective value found?

2. How quickly did the optimizer converge toward a good solution?

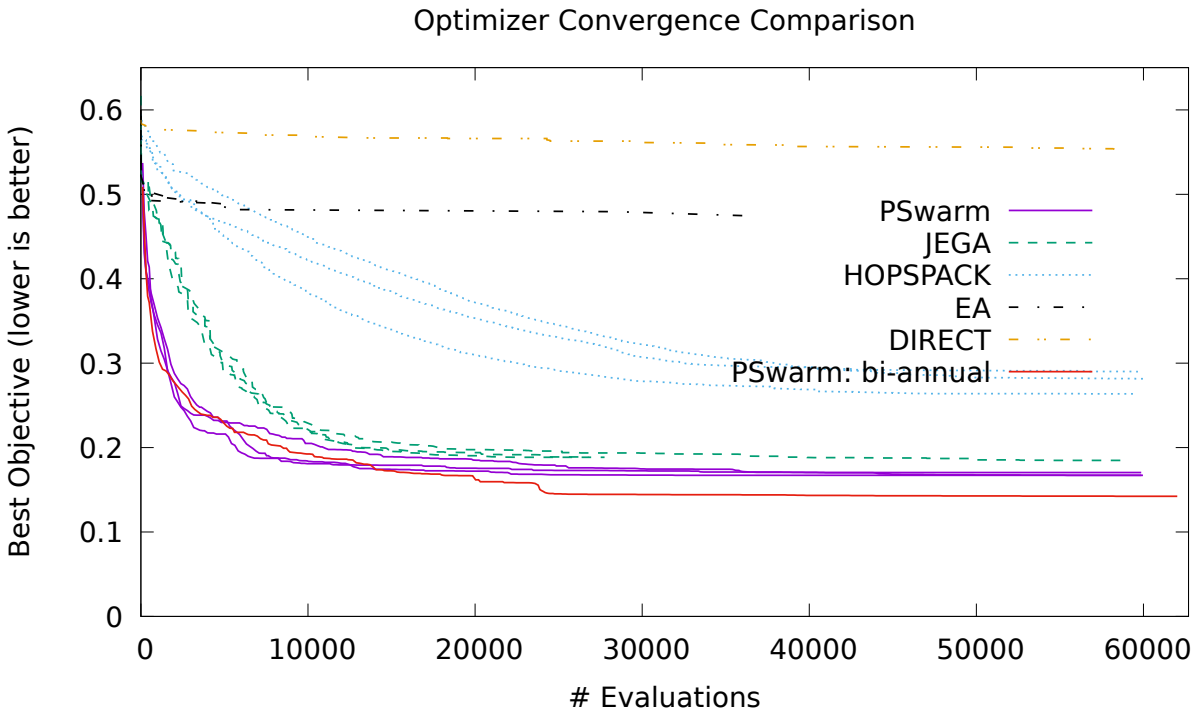

Figure 3.2: Objective value convergence curves for all optimizer runs.

3. How consistently does the optimizer perform on criteria 1 and 2 over multiple runs and/or problems?

In addition to the runs for each optimizer described above, Table 3.3 and Figure 3.2 contain data from one additional run using the PSwarm optimizer with bi-annual deployments instead of annual deployments. This run will be discussed more later.

The PSwarm implementation used was tailored somewhat to the overall fuel cycle transition deployment problem structure used, and so it is not surprising that it performs relatively well. The most significant modification involved changing how pattern search polling points were generated. Instead of polling in compass directions (along each dimension individually), polling directions were selected by randomly choosing a number of variables and stepping either forward or backward (randomly) simultaneously in each of those variables to generate a single polling point. On each polling iteration a number of random polling points were generated this way. This allows the pattern search to traverse

large portions of the solution space regardless of the number of variables in the problem.

HOPSPACK is a basic pattern search algorithm. It checks neighboring points to the current best point by modifying one dimension/variable at a time. This technique does not scale well with an increasing number of variables. As more variables are added to a problem, the upper and lower bounds for each variable remain fixed (for this problem) between zero and one. This means one of two things for the optimizer: either it takes on the order of as many iterations as there are variables to find an improving direction or the optimizer must limit its parallelism to very few evaluations per iteration. This can be seen in the slow convergence of the HOPSPACK runs. Also, due to the underlying discrete nature of the problem, it is possible for the optimizer step size to shrink below the distance required to drop over the next edge in the underlying problem that provides an objective improvement resulting in false/premature convergence.

The DIRECT algorithm suffers from similar problems to the HOPSPACK solver – each iteration tries new evaluations near good points by modifying one dimension at a time – except the DIRECT algorithm tracks several candidate points instead of working from a single best. This makes it even more difficult for it to generate long sequences of improvements deep into the problem space. And for this particular problem, the best solutions are near the boundaries of the problem space far away from the center where the DIRECT algorithm always starts. For these reasons, the DIRECT algorithm in its default configuration performs very poorly on this problem. Also notable was its large memory consumption; the DIRECT solver consumed a maximum of nearly 2 GB of memory and uses even more with an increasing number of variables.

Figure 3.3 shows both the total power and LWR power generated over time for the best deployment schedule found among all runs. Figure 3.4 shows both the total power and LWR power generated over time for the best deployment schedule found by the PSwarm optimizer using smaller $\pm 5\%$ bounds instead of $\pm 10\%$ bounds around the growth curve. This deployment schedule results in an objective value of 0.1557. This is noticeably

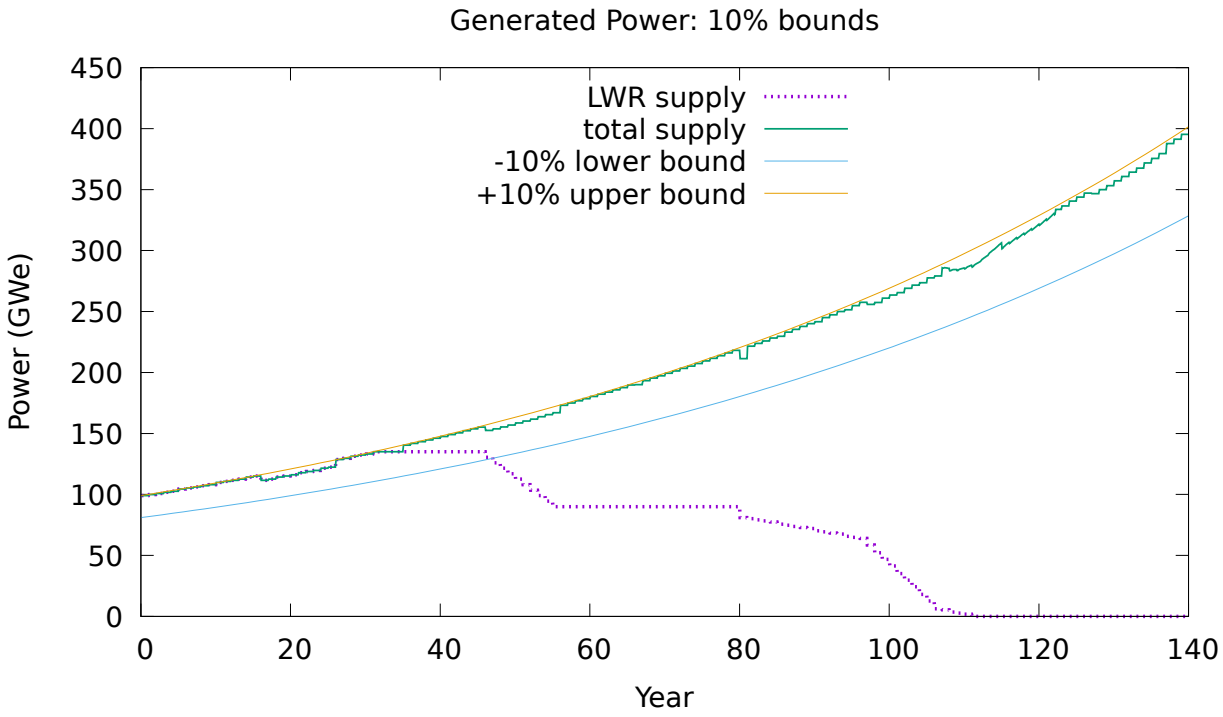

Figure 3.3: Both LWR and total (FR and LWR) generated power over time for the best deployment schedule found among all the primary experiment 1 optimization runs using the ±10% bounds around the 1% power growth curve and annual deployments.

better than the best result by other 400-variable optimizer runs which was 0.1671 from the PSwarm optimizer. The larger 10% power curve bounds effectively result in a solution space containing $[\frac{\text{new range}}{\text{prev. range}}]^n$ times as many possible solutions as the 5% bounds where range refers to the difference between upper and lower bounds and n is the number of variables in the problem. So doubling the size of the bounds for this 400-variable problem results in a solution space with 2^{400} times as many possible solutions – much larger than before.

Figure 3.3 shows that the optimizer found a solution near a corner of the parameter space, building along the upper power bound and only building fast reactors as soon as they become available. The solution shown in Figure 3.4 is similar in that it rides the +5% upper bound for nearly all of the 200 years. The only significant exception is a short period in the 10 years leading up to fast reactor availability in year 35. This makes sense - LWR

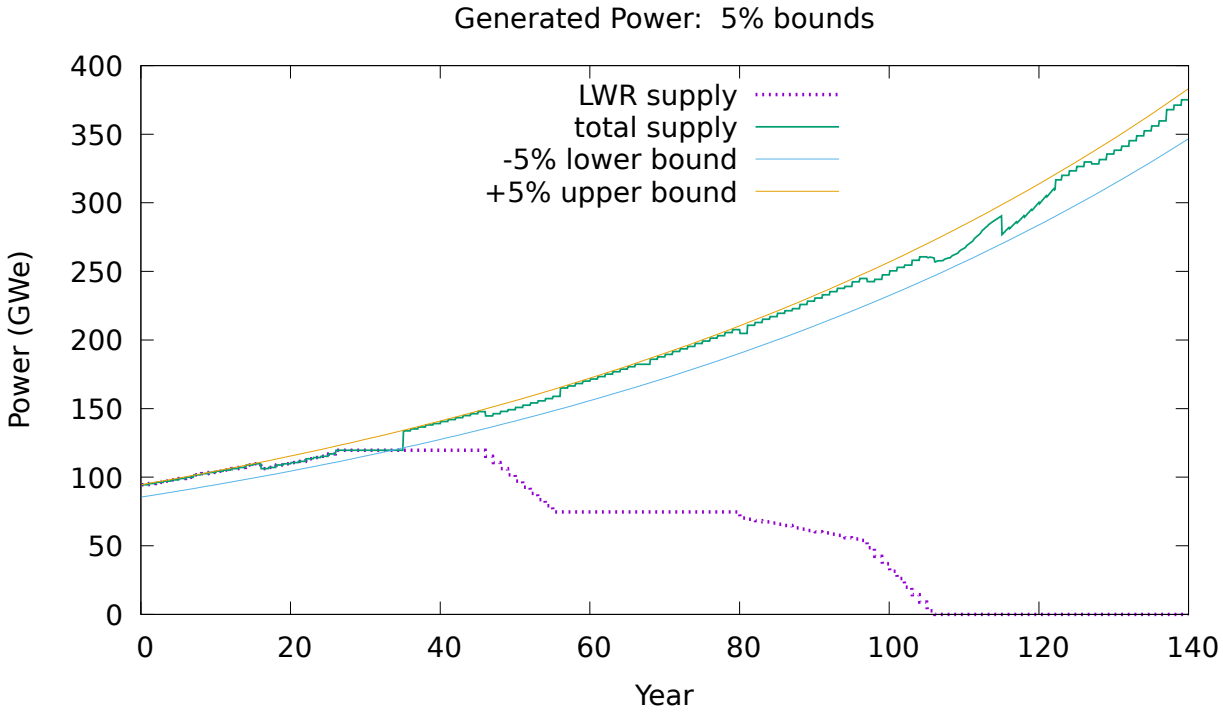

Figure 3.4: Both LWR and total (FR and LWR) generated power over time for the best deployment schedule found by the PSwarm optimizer using smaller ±5% bounds around the 1% power growth curve and annual deployments.

deployments are stopped and the optimizer utilizes its 5% flexibility to achieve a jump start in year 35 with a larger number of SFR deployments than would otherwise have been possible. This allows the LWRs to be phased out more quickly over all. This same principle applied to the ±10% case would result in even lower objective values and can actually be seen in Figure 3.5.

Figure 3.5 shows the best deployment schedule found by a PSwarm optimization run identical to all the ±10% runs except deployments occur bi-annually instead of annually. This results in an objective function with only 200 variables - half as many as before. In this coarser problem the optimizer converges more quickly and is able to find significantly better objective values than before. Although the deployments are coarser, solutions provided by the optimizer satisfy all the same constraints to roughly the same degree. There is a significant trade-off between the coarseness of the solution and the ability of optimizers

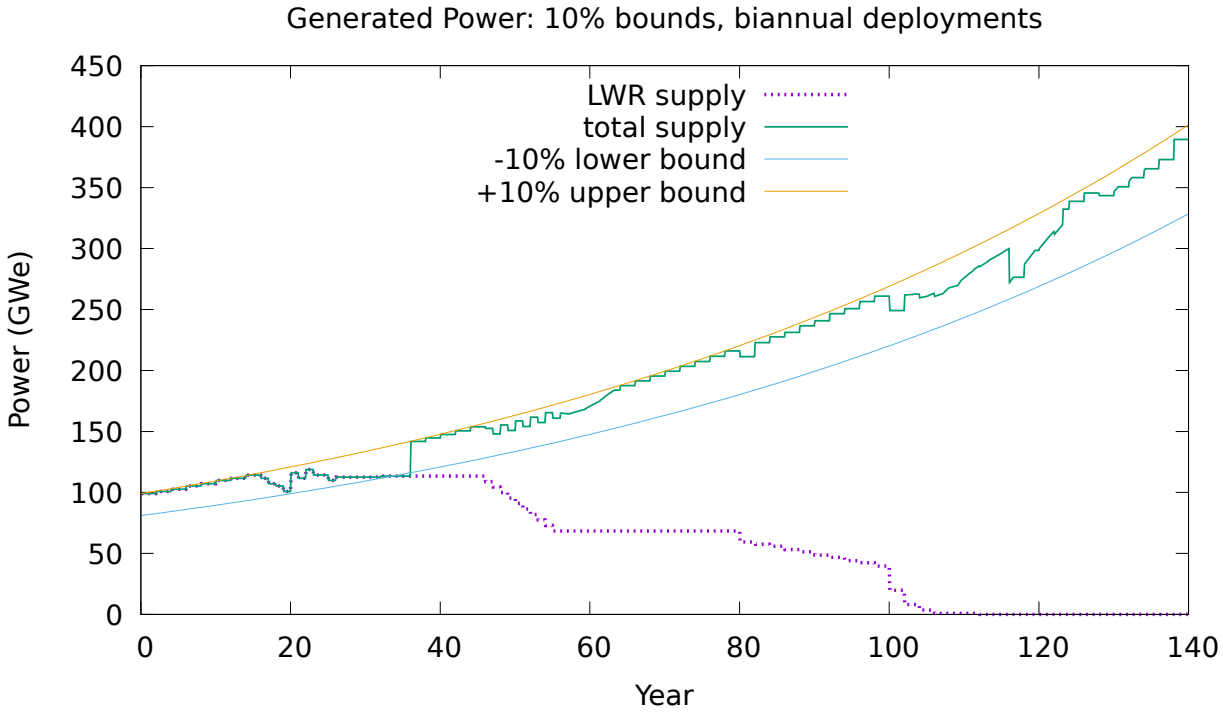

Figure 3.5: Both LWR and total (FR and LWR) generated power over time for the best deployment schedule found by the PSwarm optimizer using ±10% bounds around the 1% power growth curve and bi-annual deployments instead of annual.

to find good solutions. The results shown in 3.5 indicate that reducing variables when possible is strongly advisable, much more so than even adjusting the bounds of individual variables – which *does* have a strong impact on the constraints the solution satisfies.

The optimization results indicate potential for expediting a transition to a closed fast reactor fuel cycle by slowing/halting new LWR construction prior to SFR availability followed by an abrupt shift to SFR-only construction. Careful planning can help avoid undesirable shortages and enable the exploitation of opportunities that improve transition robustness and speed. The strategy identified by the bi-annual deployment optimization run has an objective value nearly 15% lower than naively deploying along an exponential curve, or rather it provides a 15% faster transition when treating the objective as a rough proxy for transition speed.

The custom PSwarm optimizer performs well and will be used for both Experiments 2

and 3. Recently, Chen et al. [6] have studied the effects of dimensionality on particle swarm optimization and provide some useful guidance. They find that increasing the swarm population size improves the required number of iterations for convergence relatively little for even simple objective functions (e.g. a multi-dimensional sphere). This finding is consistent with observations made throughout work done for Experiment 1. Increasing parallelism (i.e. adding more particles) provides quickly diminishing returns with respect to improving convergence, and minimizing dimensionality will remain an important aspect of continued fuel cycle deployment optimization.

Computing the primary results (excluding extra small-bounds and bi-annual PSwarm optimization runs) for Experiment 1 required about 700,000 Cyclus simulations that were run using the HT Condor high throughput computing resources at the University of Wisconsin - Madison. An additional 1 million Cyclus simulations were also run for optimizer testing/tuning and other related support activities. The optimization runs were performed with between 100 and 300 parallel computing nodes over the course of about 7 days clocking a total of around 30,000 CPU hours of work.

3.6 Summary

An effective technique for automatically determining desirable deployment schedules for fuel cycle scenarios has been developed and demonstrated using the Cyclus fuel cycle simulator. This involved developing a novel way to map optimization variables to a deployment schedule. Rather than having variables be uniform in meaning, characteristics of the fuel cycle facility deployment problem were encoded into the variable definitions. This resulted in semantically different types of variables – some representing simulation-wide power capacity levels and others representing facility ratios. This breakthrough allowed relationships between reactor facilities and power capacity/demand constraints to be represented implicitly in the definition of the variables. This not only enables optimizers

without constraint support to be used, but it also prevents wasting computational resources searching through many infeasible deployment schedules. With the simplified constraint handling, optimization can be used to analyze larger problems in addition to providing better solutions generally.

Further, the performance of several optimization algorithms was evaluated in order to identify optimizers that work well for discovering deployment schedules for fuel cycle analysis. A custom PSwarm-based optimizer was created that outperformed several other optimizers from the DAKOTA framework. This custom optimizer was selected for use in Experiments 2 and 3. The underlying discrete, discontinuous, non-convex nature of objective functions in this space makes evolutionary and other stochastic-type algorithms a good fit. Increasing parallelism provides quickly diminishing returns with respect to improving convergence. Minimizing dimensionality is important and has a large impact on the quality of answers optimization can provide for fuel cycle analysis.

The best, discovered deployment schedule indicated potential for expediting a transition to a closed fast reactor fuel cycle by slowing new LWR construction prior to an abrupt change to SFR-only construction. This deployment strategy had an objective value nearly 15% lower than naively deploying along the exponential demand curve. Although not necessarily a realistic scenario in many details, the results demonstrate the ability of these optimization techniques to discover interesting strategies and provide insight. Optimization analysis such as this is a useful tool for generally identifying potential pitfalls and opportunities in nuclear fuel cycle planning.

4 EXPERIMENT 2: FACILITY DISCRETIZATION AND TIME STEP EFFECTS

One of the major motivations for the development of Cyclus was creating a simulation environment that is more expressive and less limited than more traditional system dynamics based simulators. One of the defining differences between these two modeling paradigms involves the discretization of facilities. System dynamics simulators such as VISION simulate facility "fleets". In system dynamics fleet-based modeling:

- groups of facilities (usually all facilities of a given type) are treated as a single, aggregate entity.
- resource flows are continuous (limited by time step discretization)

Cyclus' agent-based architecture in concert with its dynamic resource exchange has the ability to model fleet-based facilities. However, Cyclus was designed with the intent of modeling facilities individually with discrete flows. The difference between these two modeling styles in terms of their effects on output of fuel cycle simulators in general is currently unknown. This experiment is designed to provide both quantitative and qualitative insight about how fleet-based continuous flow models differ from individual facility, discrete flow models with a particular focus on the deployment optimization domain.

Many fuel cycle simulators step through time in discrete steps. The duration of a single time step can have significant impact on the dynamics of a simulation. The time step duration can affect the magnitude of individual material transfers between facilities and correspondingly affects the noise level in facility material inventories. It can also act as a minimum bound for outage durations. The effects of the time step duration will also be investigated together with fleet vs facility modeling. Questions of interest include:

- How well do fleet-based models with continuous material flow capture optimal fuel cycle transitions compared to individual facility modeling?

- What effects are the primary drivers for differences in fleet-based and individual-based facility modeling?
- How sensitive are optimal fuel cycle transitions to time step duration?
- Does the impact of changes to time step duration differ in fleet-based and individual-based facility scenarios?

The results for Experiment 2 have two main sections: one dedicated to investigating simulation level differences between fleet vs. individual reactor modeling and time step duration and another focused on comparing differences between these modeling paradigms within a deployment schedule optimization context. The optimization analysis uses the same four scenarios as the simulation detail comparison with each case each optimized in exactly the same way as the scenario in Experiment 1 using the custom PSwarm optimizer.

4.1 Methodology

A single fuel cycle transition scenario is used with a few modeling variations resulting in four different cases:

1. *Case MI*: Monthly time steps with individual reactor modeling
2. *Case MF*: Monthly time steps with fleet reactor modeling
3. *Case QI*: Quarterly (3-month) time steps with individual reactor modeling
4. *Case QF*: Quarterly (3-month) time steps with fleet reactor modeling

The case labels consist of two letters each: "M" for monthly time steps, "Q" for quarterly time steps, "I" for individual reactors, and "F" for fleet reactors.

Conducting this analysis involved developing some additional capability in/around Cyclus. Three primary pieces to the methodology are:

- A fleet-based reactor model.
- A simulation scenario with both initial conditions and transition details.
- Theory and metrics for comparing differences between the four cases.

Because the standard package of archetypes included with Cyclus does not include a fleet-based reactor model, one was necessarily developed. The Experiment 1 scenario is used with appropriate adjustments for each of the four cases. A few phenomena related to the different modeling choices in each of the four cases are identified and used as a basis for comparison. These are described in more detail in the following sections.

4.1.1 Fleet Reactor Implementation

A fleet reactor archetype was created for Cyclus. The flexibility of Cyclus makes it relatively straight forward to create plug-in archetypes to model anywhere along the spectrum from individual, discrete facilities to continuous flow fleet facilities. In order to contrast well with individual, single-facility granularity modeling, the fleet archetype created here falls more toward the aggregate, continuous flow side of the spectrum.

The fleet reactor, although a single entity modeling many reactors, is actually designed to look like many individual reactors to the Cyclus simulator kernel. This "trick" allows other plugins for managing facility deployments to transparently work with the fleet reactor without having to know anything about the fleet paradigm. The fleet reactor has several important characteristics described below:

- Reactor capacity is deployed and retired in single reactor units (e.g. 1 GWe LWRs).
- Material is discharged from the fleet's reactor cores continuously (every time step) according to the following equation:

$$D = \frac{B}{L} \cdot N_r \cdot \frac{C_{inv}}{C_{cap}} \tag{4.1}$$

D is the discharge rate (kg per time step), B is the batch size (kg) for a single reactor's batch, L is the cycle length (including refueling time) in time steps, N_r is the number of reactors in the fleet, C_{inv} is the total amount of fuel in all reactor cores (kg), and C_{cap} is the total fuel capacity for all reactor cores (kg). The last fractional term accounts for lower fuel burning rates due to unfueled reactors.

- Refueling occurs continuously with as much fresh fuel as is available up to the total amount to fill all reactor cores in the fleet. No extra fuel is kept on-hand – it is all ordered just-in-time.
- Generated power is computed similarly to core discharge rates by multiplying the total fleet power capacity by the same last fractional term of Equation 4.1 to account for unfueled capacity in the fleet.
- When a reactor in the fleet is retired, a full core of material is discharged – even if there is some fraction of the fleet with unfueled cores.

One characteristic of this fleet reactor implementation is that it distributes fuel fuel sharing and perfect cycle staggering among the individual reactors it models — two effects that are described below in Section 4.3. The first discharge of spent fuel begins immediately on the first time step of operation rather than after a complete refueling cycle. Fuel shortages also only cause proportional loss of capacity rather than whole-reactor quantized shutdown.

4.2 Scenario Description

The same scenario is used for both the simulation detail analysis and the optimization analysis. The scenario and parameters from Experiment 1 are used with a few perturbations described below. A pre-release of Cyclus version 1.4 is used for this analysis. Archetypes used are the standard ones provided with Cyclus in the included Cycamore library – also a pre-release of version 1.4. The same custom storage archetype from Experiment 1 is used

because Cycamore does not yet provide one. The standard Cycamore reactor archetype is used for both LWR and SFR reactor types in cases MI and QI. Correspondingly the custom fleet-based reactor archetype just described is used for both reactor types in cases MF and QF. Additional configuration for the reactor types is shown in Table 4.1.

Table 4.1: Reactor Facility Parameters. Cycle length includes the refueling outage.

	LWR	SFR
Lifetime (yr)	80	80
Cycle length (months)	18	14
Batch size (kg)	29565	7490
Batches per core	3	5

The described scenario is used as closely as possible in each of the four cases. However, a few small case-dependent changes to the original Experiment 1 scenario relating to reactor configuration are necessary. For the cases using quarterly time steps with individual reactors, cycle and outage times must be adjusted to be a multiple of 3 months. In order to keep the cases as consistent as possible, the two with monthly time steps are also adjusted to have identical cycle and outage times. The invariants preserved with respect to reactor behavior between all four cases are shown in Table 4.2. Table 4.3 shows the computed/selected configuration for both LWR and SFR reactor types for all four cases.

In each of the four cases, the initial 100 LWRs are configured differently to have a zero-length refueling outage with an explicit 900 MWe net power output capacity. Because

	LWR	SFR
Discharge Rate ($\frac{\text{kg·HM}}{\text{month}}$)	1642.5	535
Burnup ($\frac{\text{MWe·month}}{\text{kg·HM}}$)	0.547945	0.672897
Effective Power (MWe)	900	360
Core Size (kg · HM)	88695	40125

Table 4.2: Reactor parameter invariants. Effective power accounts for refueling outages.

	LWR		SFR	
	Cases MI,QI	Cases MF,QF	Cases MI,QI	Cases MF,QF
Cycle length (months)	15	18	12	15
Refueling outage (months)	3	0	3	0
Batch size (kg · HM)	29,565	29,565	8,025	8,025
Power capacity (MWe)	1,080	900	450	360

Table 4.3: Reactor parameters by case.

the individual reactor model does not have any way to start a reactor mid-cycle, this was done to avoid having all initial LWRs refuel at the same time. The parameters for each case were also selected to keep capacity factors for individual reactors very roughly around 80-90%.

Instead of annual deployments, a longer 21-month deployment period is used because it provides better natural staggering for the reactor refueling cycles. This also reduces the dimensionality for the optimization portion of this experiment allowing the problems to be optimized more easily/robustly which in turn increases confidence that the results are caused by the perturbations of interest rather than by noise from the optimizer's inability to robustly converge in higher dimensional problems.

The deployment schedule used for the simulation detail comparison (all four cases) builds power capacity along the upper, +10% bound with no LWRs being deployed following SFR availability. This particular deployment schedule is chosen because it causes significant fuel shortages that provide interesting features to investigate and compare.

4.3 Modeling Effects

There are various low level phenomena that have potential to affect the larger outcome of simulations with respect to facility discretization and time step modeling decisions. Several effects arising from these different modeling choices are present in many fuel cycle simulators to varying degrees. Some such effects are described in detail below.

One difference between the fleet and individual facility modeling is the *quantized shut-*

down effect. If a fleet-based reactor modeling 3-batch cores is short $\frac{1}{2}$ batch of fuel, it will produce $\frac{2.5}{3} \cdot C$ of power for that time step where C is the power capacity of a single reactor of the fleet (e.g. 1000 MWe). If an individually modeled reactor is short $\frac{1}{2}$ batch of fuel, then it will produce no power ($0 \cdot C$) for that time step. Changes to time step duration can also have this effect. For both fleet and individual reactor modeling, power goes offline in full time step increments. For example, if a 3-batch fleet based reactor is short $\frac{1}{2}$ batch of fuel with a 3 month time step, then it will produce $\frac{2.5}{3} \cdot 3\text{mo} \cdot C$ of energy. In contrast, a monthly time step might allow the same reactor to produce $[\frac{2.5}{3} \cdot 1\text{mo} + 1 \cdot 2\text{mo}] \cdot C$ of energy if the fuel shortage actually ended after only 1 month. Quantifying this effect directly is difficult. One way of observing it is to compare fleet reactor scenarios that have different time step durations (i.e. cases MF and QF); relevant results discussion can be found in Section 4.4.1.1. Because the fleet reactors have perfect fuel sharing (described below) and no noise from refueling outages, differences observed will likely be due to quantized shutdown.

Individually modeled reactors each have their own refueling cycles. Depending on how the refueling cycles and outages are staggered, power production can vary significantly. For example, if all reactors refuel at the same time, then all reactors are offline at the same time. This is referred to as the *cycle staggering effect* and does not occur in fleet-based reactor models. An example of poor cycle staggering is shown in Figure 4.1. This effect can be naturally observed as the fluctuations in the difference between deployed power capacity and generated power. The figure shows power jumping above and below the installed net capacity (includes capacity factor) from time step to time step depending on how many reactors are online/offline together.

Poor reactor cycle staggering can also cause spikes of material supply and demand that can lead to suboptimal resource utilization. Consider a fuel fabrication facility that receives requests for new batches for every reactor all at once. The reactors draw out inventory from the fuel fabrication facility together in a large quantity of orders. If the fabrication facility does not have enough material on hand, some number of reactors will need to wait until

enough fuel can be fabricated. Avoiding such constraints would require the fabrication facility to maintain suboptimally large on-hand inventories as a contingency. Even with infinite material availability, poor staggering can result in supply constraints caused by finite facility throughput. Even if the fuel fabrication facility above has infinite material supply that it can keep on hand, it might not be able to fabricate fuel for all reactor requests all at once even though it has sufficient capacity if they request fuel uniformly over time.

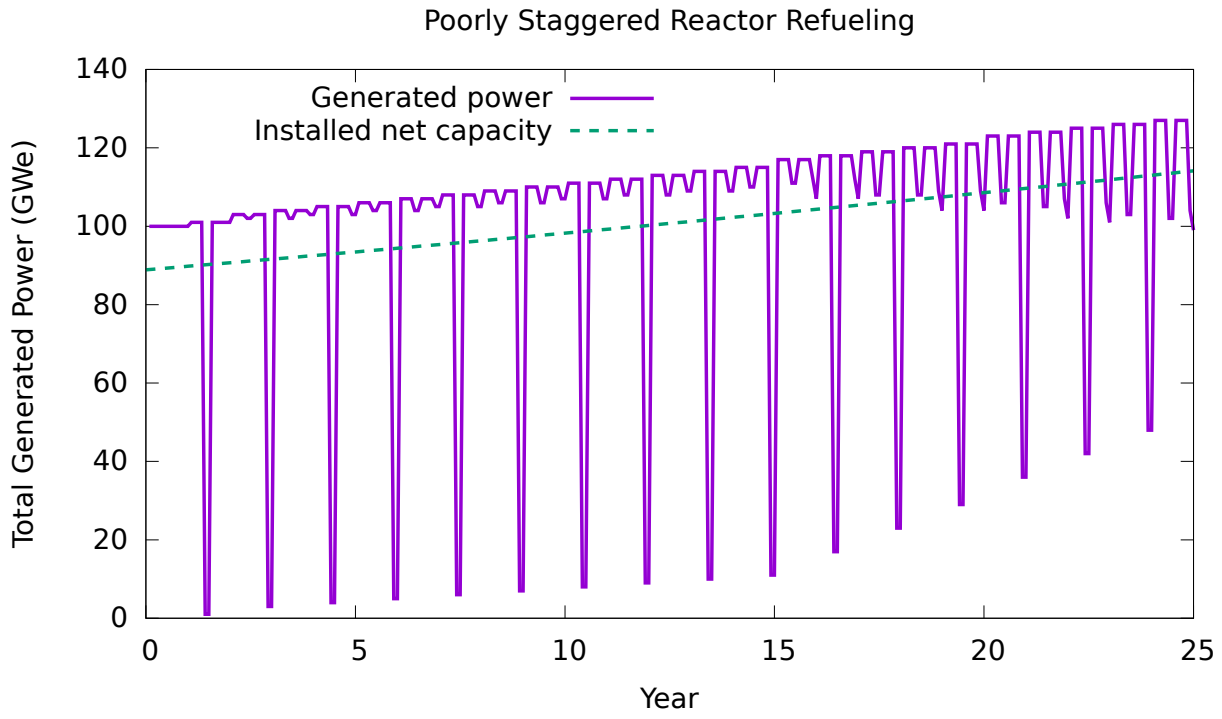

Figure 4.1: In this scenario, reactors are deployed annually with a refueling cycle length (including outage) of 18 months. As a result, all reactors deployed every 3rd year refuel together. All initial 100 LWRs start out with their cycles synchronized as well. The installed net capacity curve represents the desired generated power that would be approximately achievable if all reactor cycles were staggered appropriately.

Another difference between fleet-based and individual facility modeling involves fuel sharing. Power production by a group of individual reactors for a fixed amount of fuel shortage may vary depending on how available fuel is distributed. This is illustrated in Figure 4.2. A system being short 3 batches might mean that one reactor has no batches and isn't operating or that 3 reactors are each missing a single batch. This is referred to as the

fuel sharing effect. Fleet reactors, by design, have perfect fuel sharing - all fuel is distributed so that the minimum possible amount of capacity is offline during shortages. The most correct measure of fuel sharing inefficiency is the difference between the actual generated power and the maximum amount of power that could have been generated among all possible fuel distribution alternatives. This is difficult to actually compute. Serving as an upper bound on this inefficiency, one possible approximation is to count the number of fresh fuel batches distributed to reactors that ended up not having enough fuel to operate (i.e. a full core) that time step. This approximation assumes that every "wasted" batch of fuel, could have been given to a reactor that only needed one batch to operate. Equation 4.2 describes such an approximation.

$$N_{wasted}(t) = \sum_{r \in R_t} \left(H[t - S_{sched}(t,r)] \cdot [1 - O(t,r)] \sum_{t'=F_{prev}(t,r)}^{t} N_b(t',r) \right) \quad (4.2)$$

where R_t is the set of all reactors, H is the Heaviside function, $S_{sched}(t,r)$ is the original scheduled startup time of reactor r after the most recent refueling outage that started before t, $F_{prev}(t,r)$ is the start of the most recent refueling outage for reactor r before or on time t, $O(t,r)$ is 1 if reactor r is producing power at time t and zero otherwise, and $N_b(t,r)$ is the number of new fuel batches received by reactor r since time t. The Heaviside function is one when the reactor should be operating, and the $1 - O$ term is one when the reactor is not operating. The product of these two terms is one when the reactor is not operating but should be and zero otherwise.

One important phenomenon related to time step duration is the *inventory drawdown effect*. Increasing time step duration reduces the frequency over which refueling can occur, resulting in larger impulse drawdown on inventories. This by itself is not problematic because this is balanced by correspondingly larger inventory top-up quantities. However, another component of this effect is the simultaneity of incoming and outgoing material flows

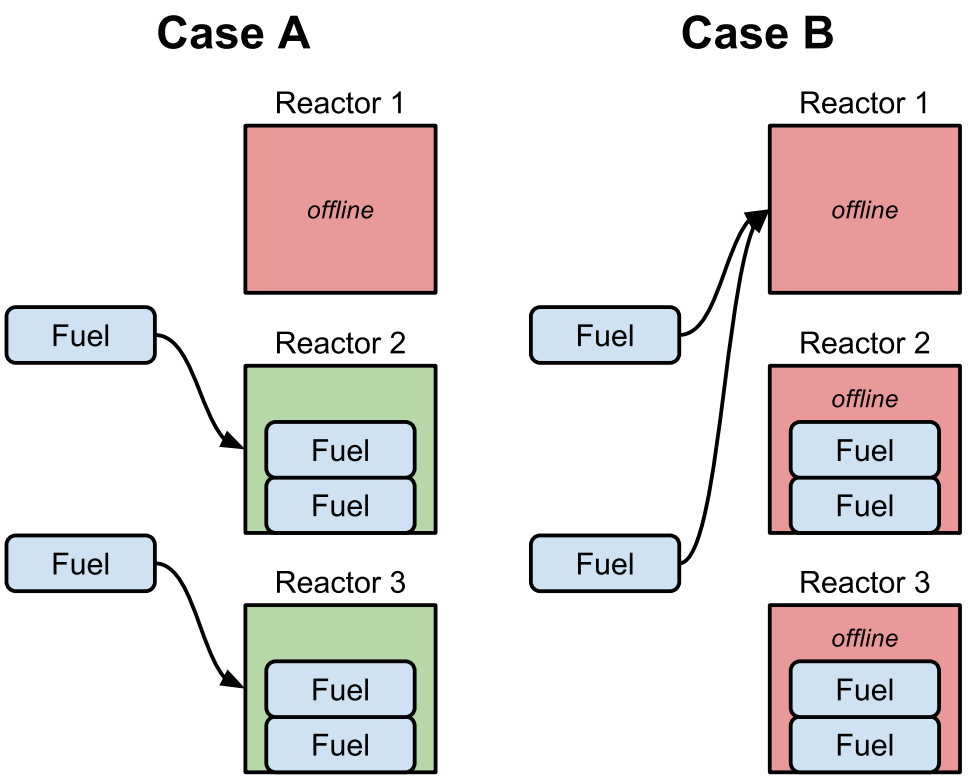

Figure 4.2: A simple diagram showing effects of constrained fuel supply distribution choices. In both cases A and B, reactor 1 needs 3 new fuel batches to operate and reactors 2 and 3 each need 1 fresh batch. In case A, the two batches are given to reactors 2 and 3, resulting in reactor 1 remaining offline. In case B, the available fuel batches are given to reactor 1 resulting in all three reactors being offline.

on a given time step. Within a particular time step facilities do not know about incoming material when they make commitments for outgoing material. This is a fundamental property of the resource exchange mechanism in Cyclus and is a restriction that applies generally to many fuel cycle simulators due to the nature of the fuel cycle analysis problem. For longer time steps, larger incoming quanta of material are not available for making offers. In general longer time steps create a need for larger floating inventory buffers in order to avoid material shortages. Another aspect of this effect is that at least one time step is required for a material object to move between two facilities. Larger time steps result in longer minimum bounds on time to traverse paths between facilities. Although direct measures of this effect are difficult, some of its consequences (e.g. higher standing

inventories during shortages) can be quantified by comparing scenarios with different time step durations (e.g. cases MF and QF) which is done in Section 4.4.1.3.

There are also performance differences between the two modeling paradigms. A single 200 year simulation with several hundred reactors takes about 2 seconds to run on a typical desktop machine with fleet-based modeling where the same simulation using individual reactor modeling takes about 20 seconds. This order of magnitude difference is an important trade-off to keep in mind when doing fuel cycle analysis in general, particularly when considering optimization or sensitivity studies where running thousands or millions of simulations may be desirable. The time step duration has less impact on performance because the amount of work being done in the simulation is similar; in the monthly time step cases most facilities (i.e. all reactors) only perform actions every refueling cycle rather than every time step.

4.4 Results

4.4.1 Simulation Detail Comparison

4.4.1.1 Power Production

Figure 4.3 shows an overall view of the generated power over time for each of the four cases. Figure 4.4 normalizes the Figure 4.3 curves to the expected exponential power curve magnifying some interesting differences between the four cases. For the first 100 years, the four cases behave somewhat similarly, although the individually modeled reactors in cases MI and QI actually go offline and come back online for refueling causing more variance. Around year 100, a fuel shortage begins and more significant differences between the four cases become apparent. This fuel shortage lasts until about year 140. The power generated in case QF is slightly lower than in case MF during the fuel shortage years. This is due to a combination of the *quantized shutdown effect* and the *inventory drawdown effect*. The longer

3-month time step in case QF results in reactor capacity going offline longer than necessary.

During the initial 100 years, cases MI and QI have larger variance in power output than cases MF and QF. This is caused by minor refueling cycle synchronization. The *cycle staggering effect* becomes much more visible during and after the fuel shortage as evidenced by the larger swings in power generation from time step to time step. Initially, the reactor cycles are staggered well. During the shortage, several reactors that previously had staggered cycles are all offline together waiting for fuel. On time steps where reactors retire (not just during the shortage), new deployments are made to replace them in addition to new deployments made to address power demand growth. Because the initial reactors retire in waves somewhat close together, there are corresponding waves of deployment, and these waves are echoed every 80 years (the reactor lifetime). These waves of deployment cause waves in recycled fuel availability that cause many of the reactors offline during the shortage to receive fuel and come online together. The net effect is that the fuel shortage degrades reactor cycle staggering overall. As the simulation continues, however, new reactors continue to be deployed and reactors with synchronized cycles retire resulting in a gradual return to better staggering visible by the end of the 200 years.

4.4.1.2 Fuel Shortage

For the fleet reactors, fuel shortage is exactly the difference between generated power and installed power capacity. Measuring the fuel shortage for individual reactor cases, however, requires careful accounting of the difference between reactors that are offline for a normal refueling outage and reactors that are offline because they have insufficient fuel. Equation 4.3 shows how this is calculated for cases MI and QI.

$$P_{outage}(t) = \sum_{r \in R_t} C_r \cdot H[t - S_{sched}(t, r)] \cdot [1 - O(t, r)] \tag{4.3}$$

where R_t is the set of all reactors, C_r is the power capacity of reactor r, H is the Heaviside

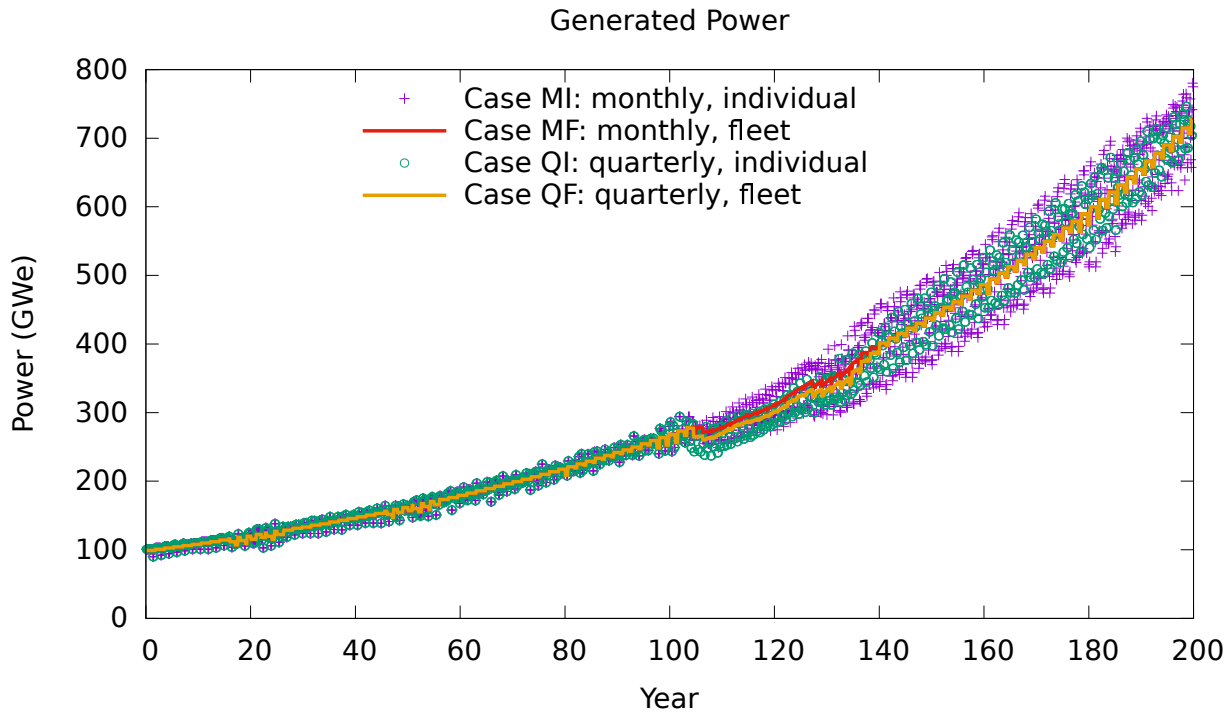

Figure 4.3: Generated power for all four cases. A fuel shortage occurs from about year 100 to about year 140. In cases MI and QI with individual reactor modeling, refueling cycles become much more synchronized during and after the fuel shortage resulting in large power fluctuations - an artifact of the *cycle staggering effect*.

function, $S_{sched}(t, r)$ is the original scheduled startup time of reactor r after the most recent refueling outage that started before t, and $O(t, r)$ is 1 if reactor r is producing power at time t and zero otherwise.

Figure 4.5 shows the fuel shortage more explicitly for each of the four cases with the case MI and QI results being the cumulative version of Equation 4.3. The individual reactor modeling cases result in more cumulative outage than corresponding fleet-based cases. The quarterly time step of cases QI and QF also make the outage significantly more severe. Cases MI and QI exhibit approximately twice as much offline power as their corresponding fleet based simulations in cases MF and QF. The monthly time step simulations also have roughly twice as much offline power as their corresponding quarterly cases.

Quantifying imperfect fuel sharing is a bit difficult, but Figure 4.6 provides one way to see the *fuel sharing effect* showing wasted batch-months computed using Equation 4.2. Fleet

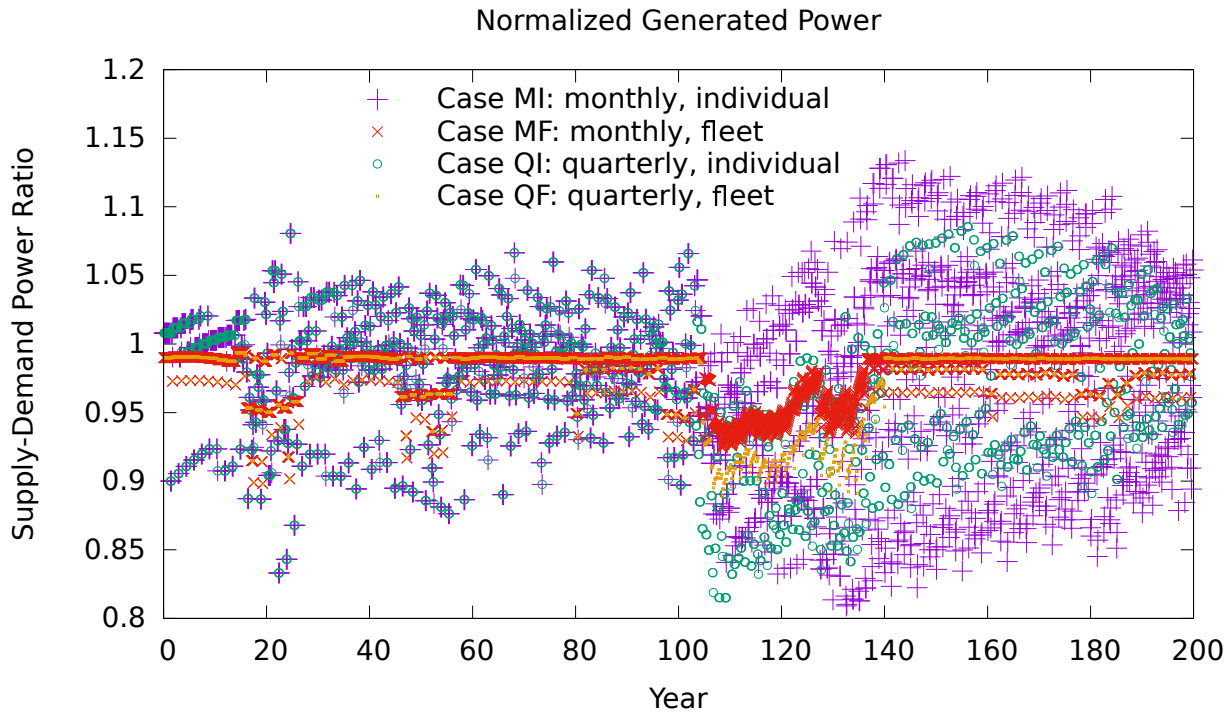

Figure 4.4: Generated power is normalized to the expected 1% exponential growth curve for all four cases. A fuel shortage occurs from about year 100 to about year 140. The *cycle staggering effect* can be seen as excessive divergence above and below 1.0 in cases MI and QI where the shortage increases cycle synchronization significantly. The *quantized shutdown effect* can also be seen in the discrepancy between quarterly and monthly time steps for fleet reactor modeling (cases MF and QF) during the shortage; a longer time step causes more reactor outage than necessary. The consistent jumping up and down for the fleet reactor cases (even pre-shortage) is due to the mis-alignment of reactors going offline (multiples of 12 months) and the build period (21 months).

reactor modeling by design exhibits perfect fuel sharing (zero inefficiency) and so is not included in this figure. Every batch that is given to a reactor that ends up not being able to start its cycle at the scheduled time (due to fuel shortage) adds one to this cumulative for each time step the cycle start is delayed.

One useful over-estimate of the fuel sharing inefficiency is to assume every one of the "wasted" batches could have been given to a reactor enabling it to operate. Multiplying each of these cumulative batches by the fast reactor power capacity (i.e. 450 MWe) provides a way to compare fuel sharing inefficiency with the overall fuel outage. This results in a cumulative energy deficit caused by poor fuel sharing of roughly 2,000 GWe · months

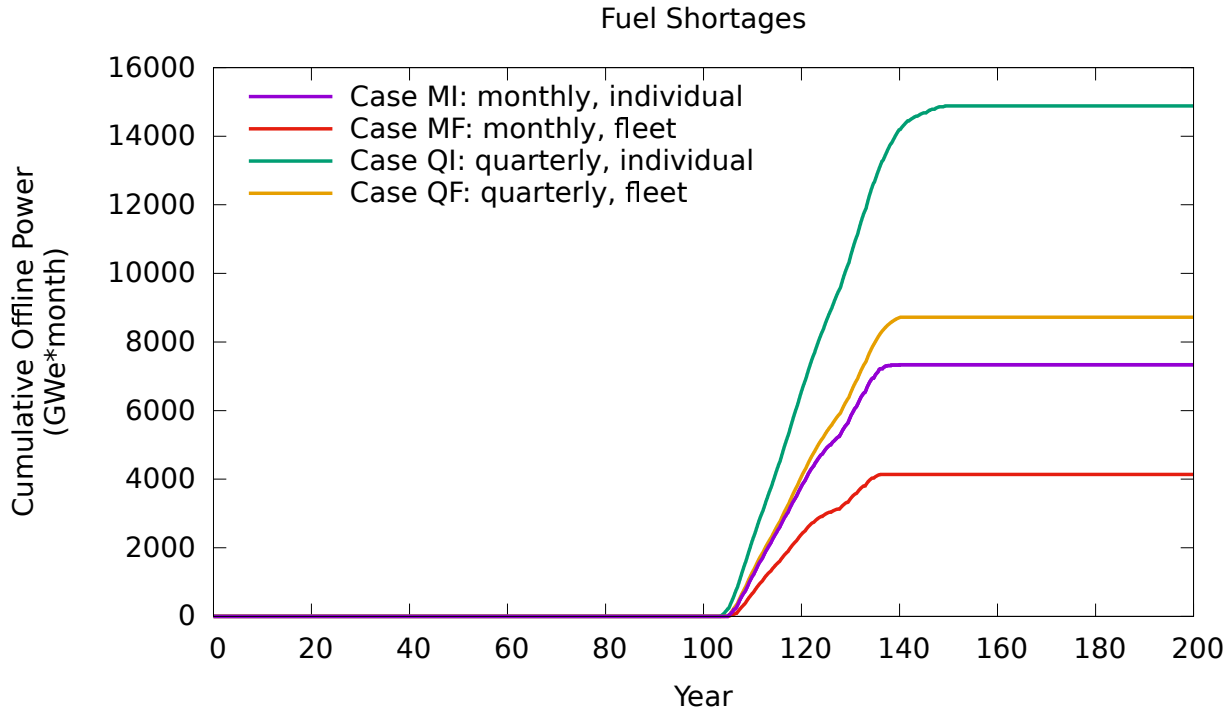

Figure 4.5: Cumulative offline power for reactors that had delayed cycle start due to fuel shortage. The difference between individual and fleet based modeling is primarily a result of the *fuel sharing effect*. The difference between monthly and quarterly time steps is primarily caused by the *inventory drawdown effect*.

and 6,600 GWe · months for cases MI and QI respectively. These approximations account for a significant portion of the difference between individual and fleet cases in Figure 4.5 suggesting that most of the discrepancy in cumulative outage between fleet and individual modeling is caused by inefficient fuel sharing in the individual reactor cases.

The poor fuel sharing exhibited in the individual reactor cases would not occur if there were no new reactors being built during the shortage. Previously operating reactors only ever need one batch - receiving only a single batch allows them to operate. Newly constructed reactors need multiple batches in order to begin operation and cannot operate if they receive less than a full core. The real world does not necessarily optimize for fuel sharing efficiency very cleanly. There are many factors that can affect real-world fuel sharing outcomes. A few include:

1. Multiple fuel quanta comprise a single batch (i.e. multiple assemblies per batch). This functionality is natively supported in the individual reactor model, although it is not used here.

2. On-hand fresh fuel inventory maintained at reactors. This functionality is natively supported in the individual reactor model.

3. Long-term fuel contracts between reactors operators and fuel suppliers.

Having multiple assemblies per batch (item 1 above) will further degrade fuel sharing efficiency. Even reactors that are short only one batch could potentially receive some fuel and still be unable to operate. In order to maintain optimal sharing, fresh fuel now needs to be sent in all-or-nothing multi-assembly quanta with higher preference to reactors that need fewer assemblies. Spare fresh fuel inventory (item 2 above) will have the global effect of increasing total shortage severity; on average more batches of fresh fuel will be sitting unused. This will always be true for cases where the system is resource constrained by the emergent dynamics (e.g. not necessarily when constrained by explicit limits such as facility throughput). However, holding spare fresh inventory can potentially reduce the frequency of outages for individual reactors in some cases.

If they were to ever occur, real world fuel shortages would likely not result in optimal fuel sharing. However, modeling these the causes of suboptimal fuel sharing is probably best accomplished with a more direct, intentional approach rather than as a modeling artifact as seen here in cases MI and QI. One way to alleviate the poor fuel sharing is to modify the individual reactor model to adjust the preference value on requests for fresh fuel depending on how many assemblies it needs to have a full core. The more assemblies it needs, the lower it will set its request preference. This will have the effect of allowing the DRE in Cyclus to prefer sending fuel to reactors that need less to operate.

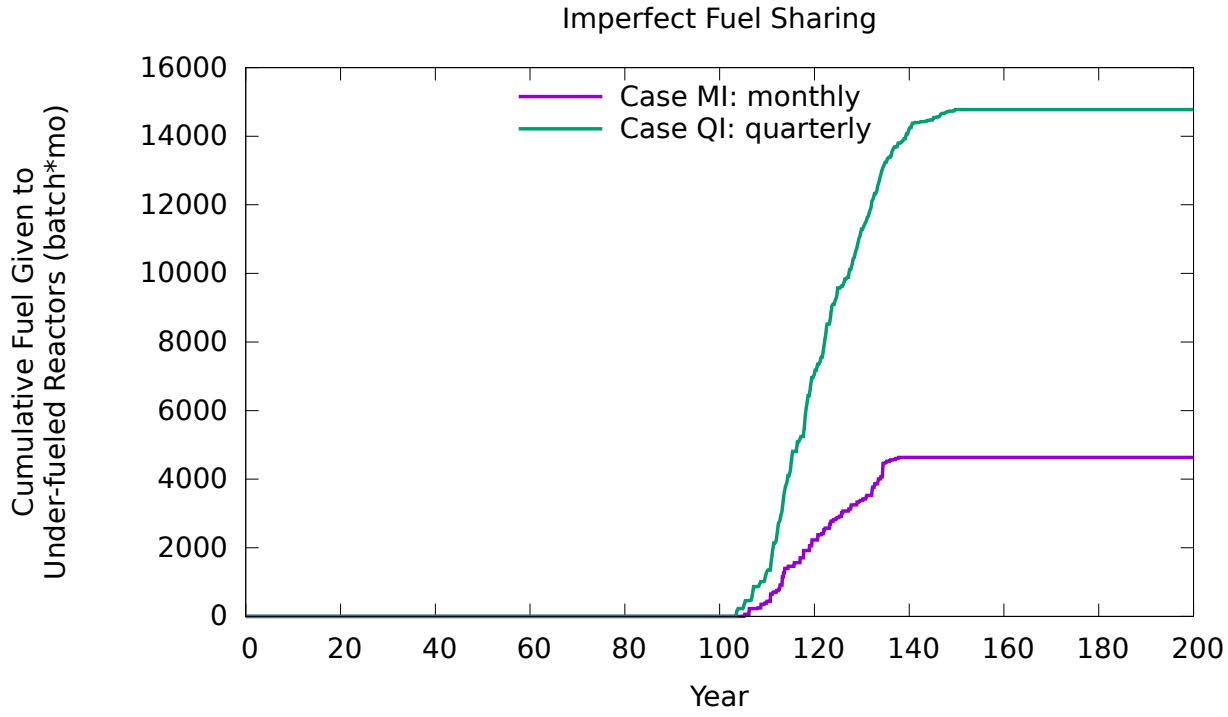

Figure 4.6: Fuel sharing inefficiency approximated by the cumulative number of fuel batches given to each reactor on each refueling period multiplied by the number of time steps that reactor had to delay the start of its next cycle (due to fuel shortage). Fleet reactors implicitly have perfect fuel sharing and so are not included here. The large difference between cases MI and QI is mostly due to the *inventory drawdown effect*.

4.4.1.3 Inventory Drawdown

The large differences seen between monthly and quarterly time step cases in Figures 4.5 and 4.6 are primarily caused by the *inventory drawdown effect*. Figure 4.7 helps to visualize this effect. The black dots in the figure represent impulse flows of available Pu from fuel fabrication and into fast reactors. Blue lines represent the flows into Pu inventory available for making fuel. The in-flows are not shown for the individual reactor cases because they are somewhat messy and obscure other information in the plots. The other colored curves show Pu inventory available for fabricating fast reactor fuel. When the inventory curves meet and force down the out-flows, fuel shortages occur. While this is most clear for cases MF and QF, the individual reactor scenarios (cases MI and QI) also have many points during the shortage where the outflow point lies exactly on top of the inventory point -

indicating that all available inventory is being transferred.

Quarterly time steps have less frequent but larger impulse material flows. The larger in-flows, however, do not compensate for the larger withdrawals because facilities do not know about incoming inventory when they resolve their outflow for a particular time step. This information lag effectively requires the floating Pu inventory to be maintained at a higher level during the shortage. This can be clearly seen in the Figure 4.7 case QF plot which has a higher standing inventory than case MF during the fuel shortage years. The case QF Pu inventory is unable to drop below approximately 100 tonnes, which indicates out-flows to be roughly equal to the Pu flow into inventory each time step during those years. This can be confirmed by the in-flow and inventory curves touching at those points. Case MF Pu inventory is similarly limited at about the 30 tonne level. As expected, the case QF withdrawals are also three times higher than the case MF withdrawals, exactly matching the time step duration ratio between the two cases.

Throughout most of the shortage, the aggregate outflow of Pu from separated inventory is roughly the same in cases MF and QF. However, due to the longer time steps in case QF, the shortage starts slightly sooner because Pu inventory must be maintained at a higher limit as described above. The small difference in outage start time has feed-back effects. Not only are extra reactors offline, but those extra reactors are also discharging less aggregate spent fuel for recycling. This increases the discrepancy between the two cases. Eventually the shortage in case MF ends slightly before it does in case QF. This feedback effect occurs because the reactors have a conversion ratio greater than 1.0.

The horizontal striations visible in the monthly flows for case MI in Figure 4.7 are a characteristic of the synchronization of reactor refueling caused by the shortage. A close-up view of these can be seen in Figure 4.8. They begin at the end of the fuel shortage window when many reactors come online together in groups and are resonances that the inventory level jumps between. This artifact is associated with poor cycle staggering like that shown earlier in Figure 4.1. These resonances gradually disappear as the simulation progresses

farther in time beyond the shortage and refueling cycles become more staggered.

The in-flows shown for cases MF and QF show a periodic pulsing that begins during the shortage in about year 120 and is shown in greater detail in Figure 4.9. This pulsing actually has nothing to do with the shortage and is caused by the retirement of fast reactors. The first fast reactors are deployed in year 35 and have an 80 year lifetime. When they retire, they discharge an entire core's worth of fuel rather than a single batch. After a 7-year cooling period, the first discharged full cores begin to make it through the recycle loop in year 122. Another interesting feature visible in all four cases in Figure 4.7 is the minority fraction of out-flow points that are higher than the bulk majority. These higher out-flows occur once every 21 months when new reactors are built because new reactors require a full core of fresh fuel rather than the single batch needed when just refueling.

Figure 4.10 shows the same inventory curves from Figure 4.7 superimposed. Perhaps counter-intuitively, lower separated plutonium inventory during the shortage generally indicates fewer unfueled reactors. Lower inventory levels mean available Pu is being utilized more efficiently rather than sitting idle. Cases QI and QF with their longer time steps show higher inventories indicating a more severe shortage. Cases QI and QF take longer to recover from the shortage and fall a bit behind with respect to building up Pu stocks due to the delayed contributions of more unfueled reactors to the separated Pu pool. This can be seen most clearly in Figure 4.11. Cases MI and QF recover Pu inventory post-shortage at about the same time and rate — this is consistent with their approximately equivalent cumulative offline power capacity curves in Figure 4.5.

The drawdown effect is more pronounced in the fleet-based cases partly because the noise from discrete reactor outages (normal refueling and otherwise) partially subsumes the drawdown penalty — discrete withdrawals from fabrication inventory are already somewhat aggregated into possibly uneven chunks. Also, because case QI pays the quantized shutdown penalty for both longer time step duration and individual reactor discretization, this effect is more significant here than in the other three cases. These observations help

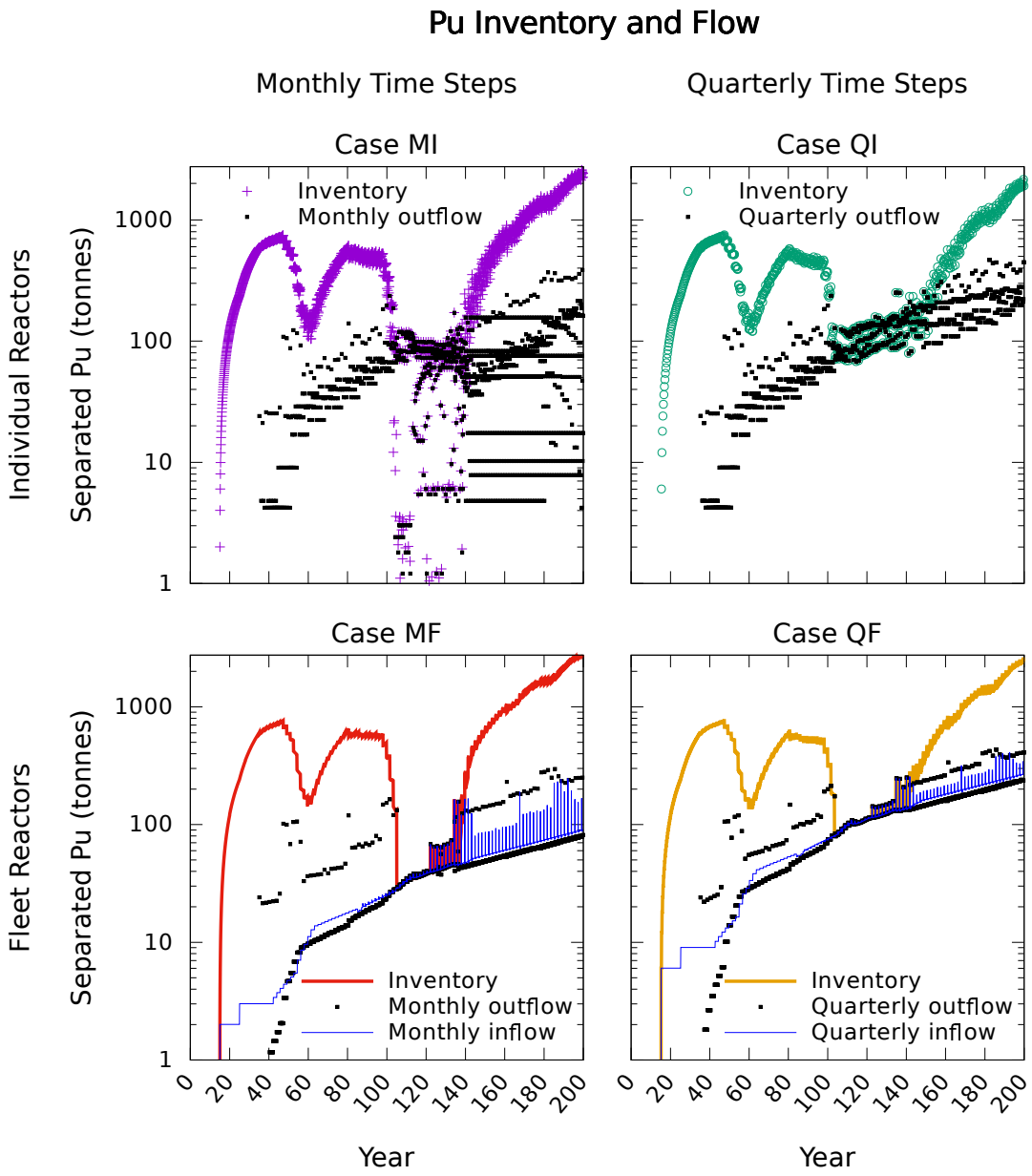

Figure 4.7: Separated Pu inventory and flows for all four cases. In-flows for cases MI and QI are omitted because they are noisy and obscure other useful information. When out-flows meet inventory, fuel shortages are taking place. A larger time step results in larger per time step material flows. These larger flows mean more material is not available for supplying to requesters. This information lag causes more supply constraints than occur with smaller time steps - illustrating the *inventory drawdown effect*. Another consequence of this effect is the larger standing Pu inventory during the shortage with larger time steps that can be seen in case QF than in case MF.

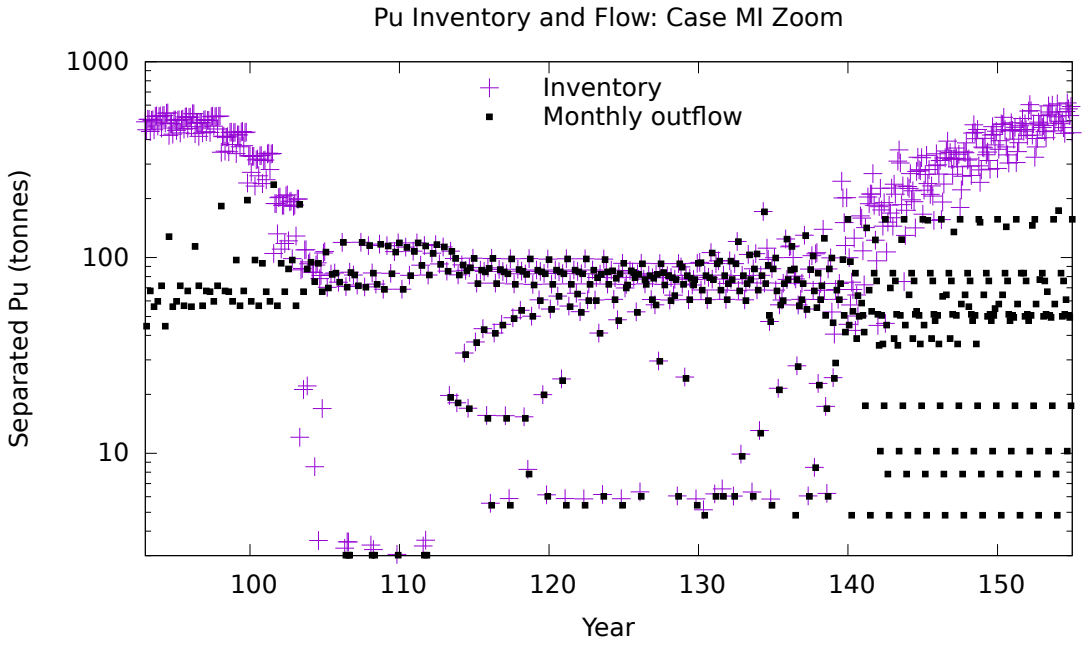

Figure 4.8: Zoom view of separated Pu inventory and flows from Figure 4.7 for case MI. At locations where the outflow dots lie on/near the inventory level, fuel shortages are taking place. This begins around year 105 and continues until about year 140 where Pu withdrawals become lower than standing inventory. The horizontal bands in out-flows starting around year 140 are caused by many previously offline reactors coming online together in groups. These fuel inventory level resonances gradually disappear as the simulation progresses.

to explain the smaller difference between cases MI and QI for fuel-shortage Pu inventory levels than the difference between cases MF and QF despite cumulative offline power differences being relatively consistent across fixed time step duration cases.

Also notable is that different modeling choices (e.g. time step duration, facility discretization, etc.) may have varying levels of significance depending on the scenario/context they are operating in. As shown in Figure 4.10 before the fuel shortage begins, differences between each of the four cases are relatively minor. However, after the shortage begins near year 100, separated Pu inventories vary much more significantly between the different cases.

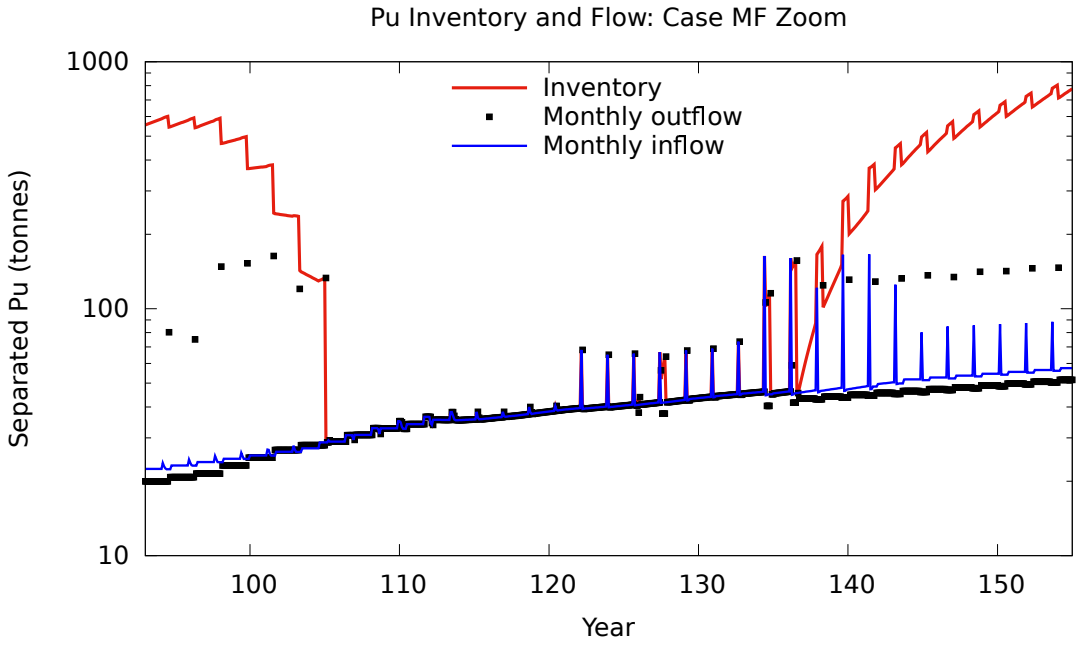

Figure 4.9: Zoom view of separated Pu inventory and flows from Figure 4.7 for case MF. At locations where the outflow dots meet the inventory level, fuel shortages are taking place. This begins around year 105 and continues until about year 140 where Pu withdrawals become lower than standing inventory. The periodic pulsing that begins just after year 120 is caused by the first fast reactor retirements; the first fast reactors deployed starting in year 35 retire annually after their 80 year lifetime and discharge an entire core's worth of fuel rather than the usual single batch for refueling.

4.4.2 Optimization Comparison

For each of the four cases, an optimization run was performed using the custom PSwarm algorithm used in Experiment 1. The convergence (i.e. objective value vs evaluations) for each case is shown in Figure 4.12. The difference between the lowest (case MF) and highest (case QI) objective values is about 0.005 or approximately 3.5% of the best case MF objective of 0.1406. We expect lower objective values to be achievable in the shorter time step scenarios (cases MI and MF) and this is supported by the figure.

In order to provide a more insightful comparison of optimization runs of the four cases, the (approximately) optimum deployment schedule found in each of the four runs was used to run a simulation in each of the four scenarios. For example, the best build schedule found by the case MI optimization was run in the case MF scenario. The objective value

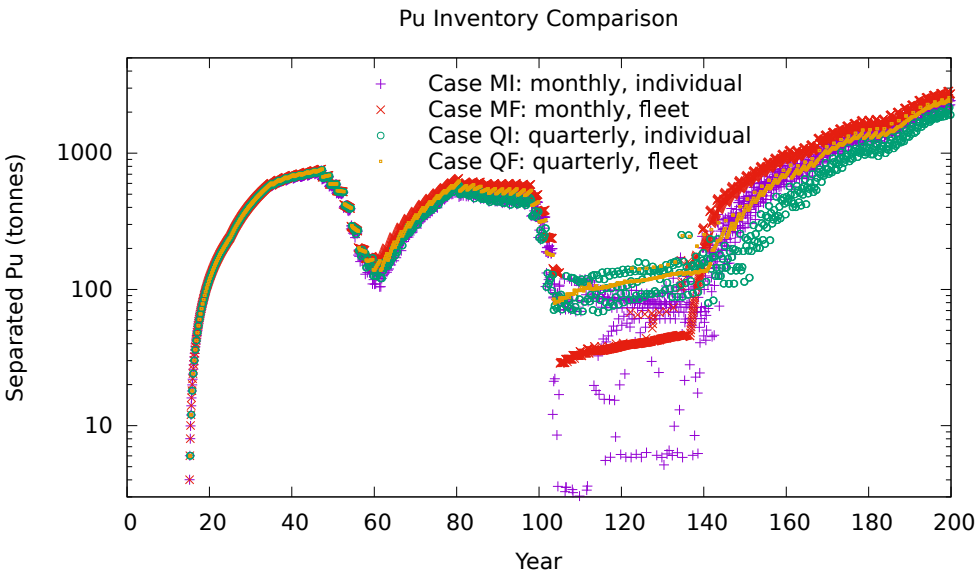

Figure 4.10: Separated Pu inventories for all four cases. Differences in modeling choices are not particularly significant until about year 100 when fuel shortages begin. These differences gradually decay away starting in year 140 after fuel shortages have ebbed. Higher aggregate Pu inventory during the shortage is correlated with both a longer shortage and a longer post-shortage recovery period.

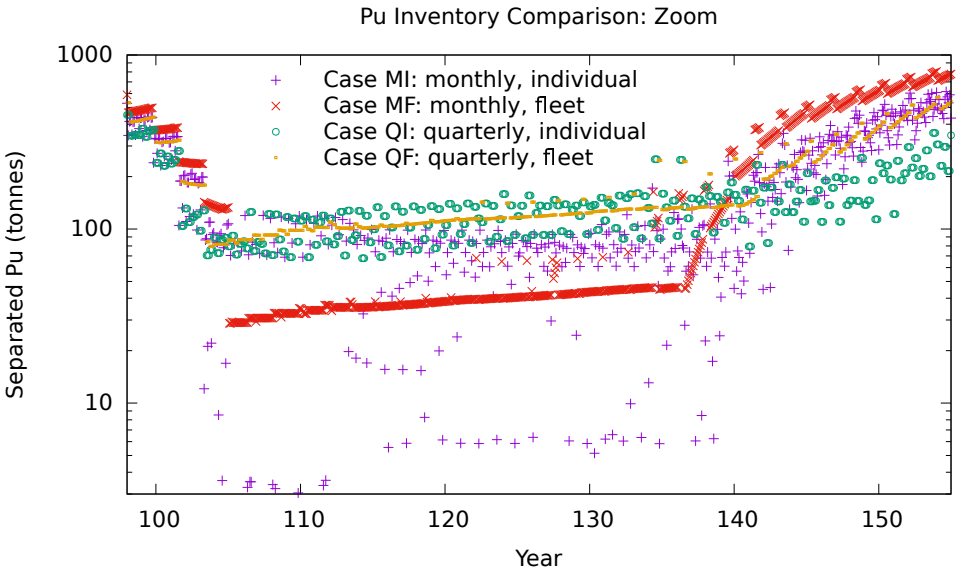

Figure 4.11: Separated Pu inventories for all four cases zoomed in on the fuel shortage years. Higher aggregate Pu inventory during the shortage is correlated with both a longer shortage and a slower post-shortage recovery of Pu inventory.

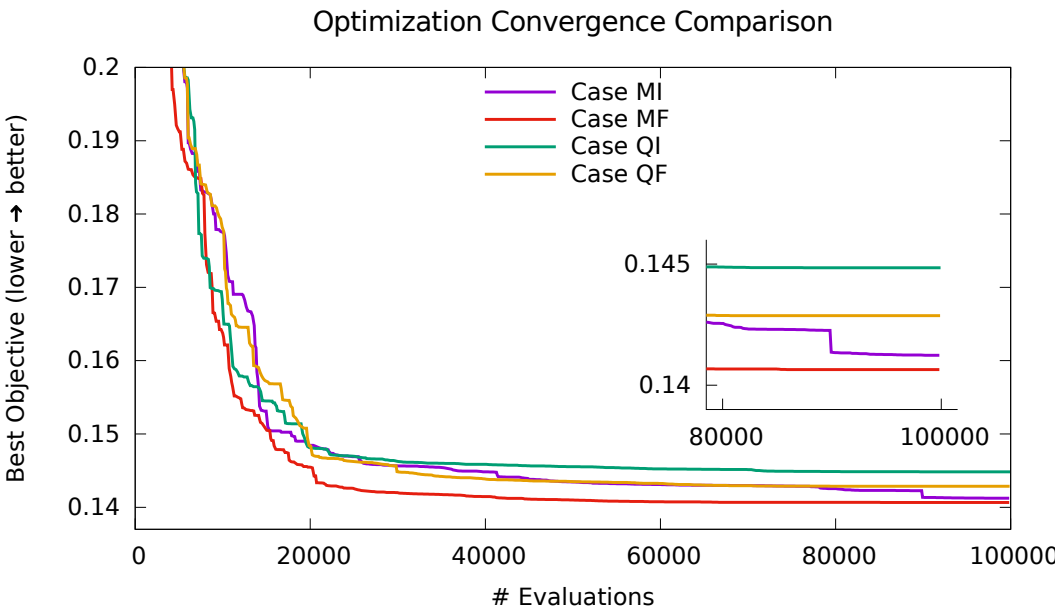

Figure 4.12: Objective value convergence curves of the optimization run for each of the four comparison cases.

was computed for all sixteen scenario combinations, and the results are shown in Figure 4.13.

Most notable, perhaps, is that the optimum build schedule from the case MF optimization performs better than all other "best" build schedules. This indicates that, at least for cases MI, QI, and QF, the optimizer failed to converge near the true optima. This indicates that the smoothness provided by the smaller time step and fleet-based reactor modeling provides an easier objective function landscape for the optimizer to navigate effectively.

It is interesting to note that the optimizer managed to make a significant jump toward a better objective value near the end of the 100,000 evaluation limit suggesting that the algorithm could potentially provide convergence to lower optima in scenarios similar to these four cases if given more than the roughly 500 iterations. Such improvements may have also been possible for the optimization runs of cases MI, QI, and QF if they had continued to run beyond 100,000 evaluations, bringing the best objectives found closer to corresponding values for the case MF optimum build schedule as shown in Figure 4.13.

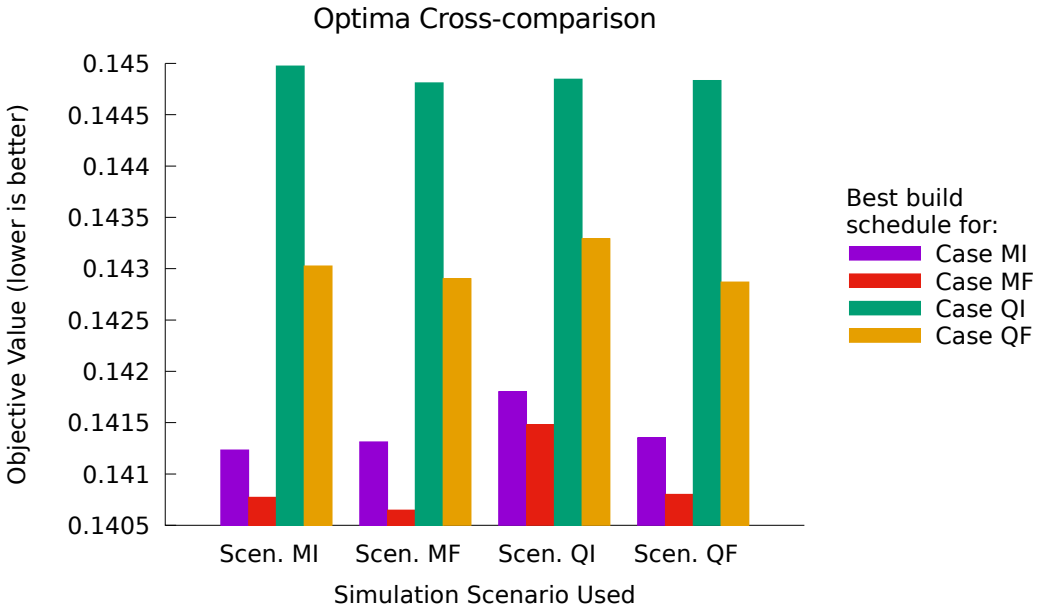

Figure 4.13: Cross comparison of objective values using the best build schedule from each of the four optimization runs inside each of the four scenarios. The build schedule from the MF case performs best in all four case scenarios, demonstrating incomplete convergence of the other three runs.

4.5 Summary

Different modeling choices can have a significant impact on the outcome of a simulation. With discrete reactor modeling many factors must be considered in order to ensure the integrity of conclusions. Individual reactor outage modeling can be very useful for certain analyses, but it can reduce result quality if things such as cycle staggering are not handled appropriately. The fleet and discrete models used here are just two points in a multi-dimensional spectrum of modeling choices. For example, if fuel sharing is of interest but cycle staggering is not, individual reactors could be used with the power capacity lowered and the outage reduced to zero duration, including the capacity factor in the reactor's power capacity.

Modeling reactors individually can provide valuable insight not possible with the fleet-based reactor modeling. For example, bringing reactors back online following shortage-induced outages should not necessarily follow the natural pacing of fuel availability, other-

wise refueling cycles become unsatisfactorily synchronized. If individual reactor modeling fidelity is not required, then performance benefits in both simulation run time and data analysis may suggest a fleet-based modeling approach.

Increasing time step duration increases facilities' standing inventory requirement during supply constrained periods. Facilities can provide no more inventory than they can hold on a single time step. Facilities with large inventory buffers that are unconstrained are not affected significantly by the time step duration when operating in isolation. However, even when facilities do not have individual inventory or throughput limits, the time step can still have a significant effect. The *inventory drawdown effect* can impact the aggregate throughput of supply-constrained recycle loops by affecting minimum bounds on idling material quantities.

The impact of modeling assumptions depends heavily on the metrics of interest. For example, someone interested in fuel-shortage driven reactor outages would observe a factor of four span among the four cases shown in the detail comparison. However, someone more interested in the overall transition mechanics would not see as significant a discrepancy. In an unconstrained supply setting, varying the time step duration or discretization of facility modeling have a relatively minor impact on the overall simulation results. Differences become more significant in material-constrained regimes. Robust behavior in constrained regimes is especially useful in the context of sensitivity and optimization analysis.

The value of realism in fuel cycle modeling can be greatly enhanced by an associated understanding of how outcomes are affected by changes in fidelity. Cyclus' flexibility for accommodating different modeling choices uniquely enables many interesting comparisons. When doing fuel cycle analysis it is important to be cognizant of how modeling choices are affecting results and conclusions through exercises like the one here. Even if the outcome is that a particular modeling decision has little impact on results, work in this direction has great potential to improve the integrity of fuel cycle analysis generally.

5 EXPERIMENT 3: HEDGING STRATEGIES FOR DISRUPTION

The goal of this experiment is to explore fuel cycle transitions while accounting for uncertainties that may be resolved at unknown times during the simulation. For example, there might be some per time step probability that a separations facility enters a long unplanned outage. It may go offline in year 2, or it may never go offline at all. Two questions that arise related to such uncertainties are:

1. How can/should the hedging effectiveness of particular deployment strategies be quantified?

2. How can we find good good hedging strategies?

In order to answer these and related questions, a disruption methodology was developed along with ways for measuring hedging effectiveness. Some extra scenario details relating to the uncertain disruption also become necessary. These details are discussed in the sections that follow. The chapter concludes with a presentation of results and a brief summary.

5.1 Disruption and Hedging Methodology

In order to facilitate unambiguous discussion about the many layers of this experiment, some important definitions must be established:

- t_d represents the time step on which a disruption occurs.

- D is a full deployment schedule for all facilities spanning the duration of an entire simulation.

- $D^*(t_d)$ is the optimum deployment schedule for a disruption occurring at time t_d.

- $O(D, t_d)$ is the objective function value for the deployment path D used in a simulation where the disruption actually occurs at time t_d. Note that the disruption may cause a perturbation in the actual objective function as well as in the deployment schedule.

- $O^*(t_d) = O[D^*(t_d), t_d]$

  This represents the best achievable objective value among all possible deployment schedules where the disruption occurs at time t_d.

In the real world, behavior changes dynamically in response to unexpected events. A deployment schedule must be allowed to be modified post-disruption. To capture this, a couple more definitions are necessary:

- $R(D, t_d)$ is any arbitrary map from D to D' where D' differs from D only post-disruption.

- $R^*(D, t_d)$ is a particular R where post t_d deployments are optimal for the objective function and scenario being considered.

- $H(D)$ is a measure of the hedging effectiveness (i.e. objective value) of deployment schedule D.

- D^*_H is the best hedging deployment schedule for a disruption scenario encoded in H with all its associated uncertainty and possible futures.

- $H^* = H(D^*_H)$ represents the best achievable hedging objective among all possible deployment schedules for the disruption scenario encoded in H.

With these definitions in place, the hedging effectiveness of deployment schedules can begin to be estimated by calculating a hedging sub-objective S^* for each of the possible disruption times t_d:

$$S^*(D, t_d) = O[R^*(D, t_d), t_d] \tag{5.1}$$

S is really the same as the single-simulation O objective, but it is helpful to make the distinction that, in the hedging analysis, S is just an intermediate result rather than the final objective value of interest. For a particular deployment schedule, the hedging sub-objectives can then be combined for all possible disruption times. There are many possible choices for computing the total hedging value of a deployment schedule. One possibility is to prefer deployment schedules that provide the best (weighted) average outcome:

$$H(D) = \sum_{i=1}^{n} p_i \cdot S^*(D, t_{d,i}) \tag{5.2}$$

n is the number of possible disruption times and p_i is the probability of the disruption occurring at $t_{d,i}$. Disruption times on or after the last moment of the simulation (i.e. $t_{d,i} \geqslant t_{end}$) are beyond the scenario's planning horizon and effectively represent no disruption occurring. The variance of outcomes is also a good candidate for explicit inclusion in calculating $H(D)$, although it will not be used for this analysis. For this analysis the expected or mean outcome is used for measuring hedging effectiveness (i.e. Equation 5.2).

Equation 5.2 can be generalized for a continuous distribution of disruption times. Even though time in the simulator is actually discrete, because time steps are much closer together then a feasible number of samples for approximating S^* treating time as continuous can be a useful approach. The continuous form for computing H is shown in Equation 5.3:

$$H(D) = \int_0^{\infty} S^*(D, t_d) \cdot p(t_d) dt_d \tag{5.3}$$

where $p(t_d)$ is the disruption probability density function (PDF). The upper bound is infinity to capture the full probability of the no-disruption case represented by disruption times after the end of the simulation. The no-disruption probability can be calculated using Equation 5.4:

$$p_{none} = \int_{t_{end}}^{\infty} p(t_d) dt_d \tag{5.4}$$

The value of S^* for all $t_d \geqslant t_{end}$ are equal to each other and are computed using by running a simulation using the full deployment schedule D (i.e. with no post-disruption mapping R). The PDF must, of course, integrate to unity over all physical times (i.e. positive t_d):

$$\int_{0}^{\infty} p(t_d) dt_d = 1 \tag{5.5}$$

Evaluating Equation 5.3 requires having a continuous form for S^* which does not exist. Approximations using sampled points of S^* can be used. While closed-form approximations enable analytic integration, because the PDF and simulation scenario details generally involve many assumptions and much uncertainty, less sophisticated techniques involving numerical integration will provide satisfactory results. The simplest approximation is perhaps to generate a piece-wise function of linear interpolations between sampled points of S^* of the form given in Equation 5.6:

$$S^*(D, t_d) \approx \begin{cases} S^*(D, t_0) + \frac{t_d - t_0}{t_1 - t_0}[S^*(D, t_1) - S^*(D, t_0)] & : t_0 \leqslant t_d < t_1 \\ S^*(D, t_1) + \frac{t_d - t_1}{t_2 - t_1}[S^*(D, t_2) - S^*(D, t_1)] & : t_1 \leqslant t_d < t_2 \\ \ldots & \\ S^*(D, t_{n-1}) + \frac{t_d - t_{n-1}}{t_n - t_{n-1}}[S^*(D, t_n) - S^*(D, t_{n-1})] & : t_{n-1} \leqslant t_d < t_n \end{cases} \tag{5.6}$$

where n is the number of sampled points. Then, by dividing the time axis into many small sub-intervals and using the mid-point rule, a numerical approximation of 5.3 can be calculated using Equation 5.7:

$$H(D) = \sum_{i=0}^{m} S^*(D, t_i) \cdot p(t_i) \cdot \Delta t \tag{5.7a}$$

$$t_i = (i + 0.5) \cdot \Delta t \tag{5.7b}$$

$$\Delta t = \frac{t_{end}}{m} \tag{5.7c}$$

where m is the number of sub-intervals. While more sophisticated numerical integration techniques can certainly provide much more precise results, the uncertainty associated with the PDF and other scenario details renders such high precision unnecessary.

Optimization can be used to search for good/optimal hedging strategies. The optimizer chooses hedging deployment schedules and the objective function returns the expected/mean outcome across the distribution of possible disruption futures. In order to accomplish this, calculating the objective function requires running several simulations corresponding to alternative disruption futures. Fully calculating the expected outcome requires computing the best achievable objective value for each alternative disruption future requiring post-disruption optimal deployments (i.e. finding $R^*(D, t_d)$ for each disruption time. Each evaluation of a single hedging objective becomes its own set of optimization problems (one for each disruption).

5.1.1 Approximating the Hedging Sub-objective S^*

R^* is often computationally expensive to compute, requiring an entire optimization run itself. Because of this, fully calculating R^* repeatedly while searching for H^* (i.e. optimization) is likely infeasible. In order to find H^*, simpler approximations for R^* become necessary. As a reference approximation Equation 5.8 can be used to compute a corresponding approximation of S^*.

$$R(D, t_d) = D \tag{5.8}$$

While this approximation is clearly "wrong", it can aid in evaluating the quality of other, more effective approximations. Another slightly better but still simple method is to

approximate S^* directly, eliminating the need to approximate R^* altogether. One way to do this involves calculating the weighted linear combination of the objective value from the deployment schedule being analyzed with the objective value for the best known schedule for that particular disruption time. This approximation is shown in Equation 5.9.

$$S^*(D, t_d) \approx \frac{t_d}{t_{end}} \cdot O(D, t_d) + \frac{t_{end} - t_d}{t_{end}} \cdot O^*(t_d) \tag{5.9}$$

If the objective function can be meaningfully computed on portions of the simulation scenario (i.e. pre and post disruption independently), then it may be possible to make an even better approximation of S^*. Like Equation 5.9, pre-computed optimum objective values for specific disruption times are combined with the objective values using the deployment schedule being evaluated. The improvement involves computing the objective value over partial scenarios (i.e. before and after t_d). This technique is represented in Equation 5.10.

$$S^*(D, t_d) \approx w_{pre} \cdot O_{pre}(D, t_d) + w_{post} \cdot O^*_{post}(t_d) \tag{5.10}$$

O_{pre} is the objective value computed using D on the partial simulation up to time t_d, and O_{post} is an estimate of the objective function on the partial simulation following time t_d. For objective functions that generate ratio-based values, non-unity weights indicate the importance of each chronological portion of the simulation. The objective function used in this analysis if based on power generation, so the weights would be chosen using Equation 5.11. For strictly sum-based objective functions (e.g. total power generated, total accumulated Pu, etc.), the weights are implicitly included in the partial objective calculation and should simply be set to one. Another incremental improvement on these approximation techniques would be to compute the raw component metrics of the objective value over partial simulations, and combine them directly into a single, full objective value rather than doing a weighted linear combination.

$$w_{pre} = \frac{1}{E_{tot}} \sum_{t=0}^{t_d} E_t \tag{5.11a}$$

$$w_{post} = 1 - w_{pre} \tag{5.11b}$$

$$E_{tot} = \sum_{t=0}^{t_{end}} E_t \tag{5.11c}$$

where E_t is all energy produced on time step t.

5.1.2 Finding and Comparing Hedging Strategies

Optimization can be used to search for good/optimal hedging strategies. The optimizer chooses hedging deployment schedules and the objective function returns the expected/mean outcome across the distribution of possible disruption futures. Since calculating the hedging sub objective S^* requires performing an optimization run itself to determine optimal post-disruption deployments, a single objective evaluation (i.e. calculating H) when searching for optimal hedging strategies would require running nested sub-optimizations for each disruption time. So if the inner disruptions each take 500 iterations to converge, and the outer optimization takes 500 iterations to converge, the full optimization would effectively become 250,000 non-parallelizable iterations. In such a scenario, even a short iteration time of 1 minute would result in multi-month run times. Approximation techniques for S^* and/or R^* as described above can be used to overcome this problem.

Using Equation 5.9, each hedging objective evaluation is transformed from a set of full optimization runs into a set of single simulation runs plus the cost of pre-computing the O^* values for each disruption time t_d via single optimization runs. Since the O^* values can be computed via parallel, the entire hedging strategy search will only take as long as 2 full optimization runs.

After a hedging strategy search has been completed, it is necessary to evaluate the actual hedging value $H(D)$ provided by the discovered deployment schedule, rather than

the approximations used during the search. It is also desirable to compute actual full deployment schedules using the hedging strategy coupled with appropriate disruption responses. For this, it is necessary to do the full optimization runs to compute R^* for each disruption time $t_{d,i}$ and then to compute each respective sub-objective S^*. This adds a third set of optimization runs that must be done following the hedging optimization itself. The actual H values can be computed in parallel for other deployment schedules (i.e. each $D^*(t_d)$) via additional optimization runs and then compared to the H^* discovered in order to either strengthen or contradict the discovered deployment schedule's label as a strong hedging strategy.

5.1.3 Objective Distributions

In order to compare the hedging value of different build schedules, factors other than the mean/expected outcome may be important. It can be useful to see the frequency distribution of outcomes for deployment schedules across the full range of possible disruption outcomes. Doing this requires using both the disruption PDF together with the disruption-objective mapping function $S^*(D, t_d)$.

Similar to the numerical integration techniques for estimating H described above, the PDF can be discretized into many small intervals (many more than before) and sampled stochastically. The sampled disruption times can then be run through the piece-wise linear approximation of $S^*(D, t_d)$ (i.e. Equation 5.6) to generate a series of objective value outcomes which can then be plotted as a frequency distribution or histogram. Because this operation is computationally cheap, a suitably fine discretization of the PDF (e.g. 100s to 1,000s of bins) can enable deterministic sampling of interval midpoints to adequately represent the continuous distribution of disruptions. This means multiplying the probability associated with each disruption time interval on the PDF by a fixed, large integer number of "samples", the result of which represents the number of times a particular disruption was sampled. This becomes the number of times each disruption time is run through $S^*(D, t_d)$

generating the set of sampled objective outcomes.

With a fixed number of these samples, direct comparison of the resulting frequency distributions or histograms from several deployment schedules is possible. This enables diverse hedging characteristics of arbitrary deployment schedules to be compared and visualized. This can answer quantitative and qualitative questions such as:

- Are the outcomes bimodal in nature?
- Which deployment schedules have lower variance in outcomes (i.e. tighter distributions)
- What are the best/worst cases for each deployment schedule and how do they compare?

5.2 Scenario

The scenario used is based on the one from Experiment 1. The same 21-month deployment period used in Experiment 2 is also used in the optimization runs here. In order to accommodate scenarios with many LWRs later in the simulation, the LWR reprocessing limit in the Experiment 1 scenario is no longer appropriate; this limit is lifted and an infinite LWR reprocessing throughput capacity is used. For this experiment, the objective function is modified slightly from the one used in experiment 1. Equation 5.12 shows the new objective function:

$$O_{sim} = \frac{\sum_{t \in sim} E_{t,\,LWR}}{\sum_{t \in sim} E_{t,\,tot}} \cdot \frac{\sum_{t \in sim} C_{t,\,tot}}{\sum_{t \in sim} E_{t,\,tot}} \tag{5.12}$$

where O_{sim} is the objective function value for an entire simulation, $E_{t,\,LWR}$ is the energy produced by all LWRs in time step t, $E_{t,\,tot}$ is the energy produced by all reactors in time step t, and $C_{t,\,tot}$ is the installed generating capacity of all reactors at time step t. The

difference from Experiment 1 objective is the second ratio that serves as an explicit, extra penalty for installed, unfueled reactor capacity. This is important because the disruption can have the effect of causing a sudden, severe reduction in fuel availability for fast reactors. So the previous implicit penalty is no longer sufficient to generate reasonable deployment schedules from the optimization process with the externally imposed disruption.

The final difference from the experiment one scenario was the addition of the disruption itself. The disruption used is designed to simulate a 33% reduction of Pu generated/available for use in SFR fuel. The disruption occurs instantly and the reduced Pu generation remains in place for the rest of the post-disruption simulation time. This could represent a sudden policy shift away from breeder reactors to burner reactors. In general it could represent any kind of event that affects the ability of a significant portion of SFRs to operate. The disruption is implemented by deploying a sink facility that requests all Pu once every third time step (i.e. every 3 months). The disruption time is adjusted by simply adjusting the time of this facility's deployment. This facility was configured to have infinite lifetime, infinite throughput, and infinite max inventory size.

5.2.1 Disruption Probability and Sampling

A disruption distribution was chosen to very roughly approximate a somewhat realistic set of possible circumstances. Current policy is unlikely to change very soon. A little into the future, technological progress and an increased understanding of climate change make a policy change relating to nuclear energy more likely. However, farther into the future, the industry becomes more invested in a particular way of doing things and it gradually becomes more difficult to diverge from the status quo. A gamma distribution (Equation 5.13) is used as the base for this probability shape:

$$p(x) = \frac{1}{\Gamma(k) \cdot \theta^k} x^{k-1} e^{-\frac{x}{\theta}} \tag{5.13}$$

Using $k = 1.5$ and $\theta = 2$, scaling out along the x axis to cover a 2400 month range using $t_d = 600 \cdot x$, and then normalizing to the new area so that the integral from zero to infinity is equal to 1.0 gives the following equation as the PDF:

$$p(t_d) = \frac{600}{\Gamma(1.5) \cdot 2^{1.5}} \left(\frac{t}{600}\right)^{0.5} e^{\frac{-t}{600 \cdot 2}} \tag{5.14}$$

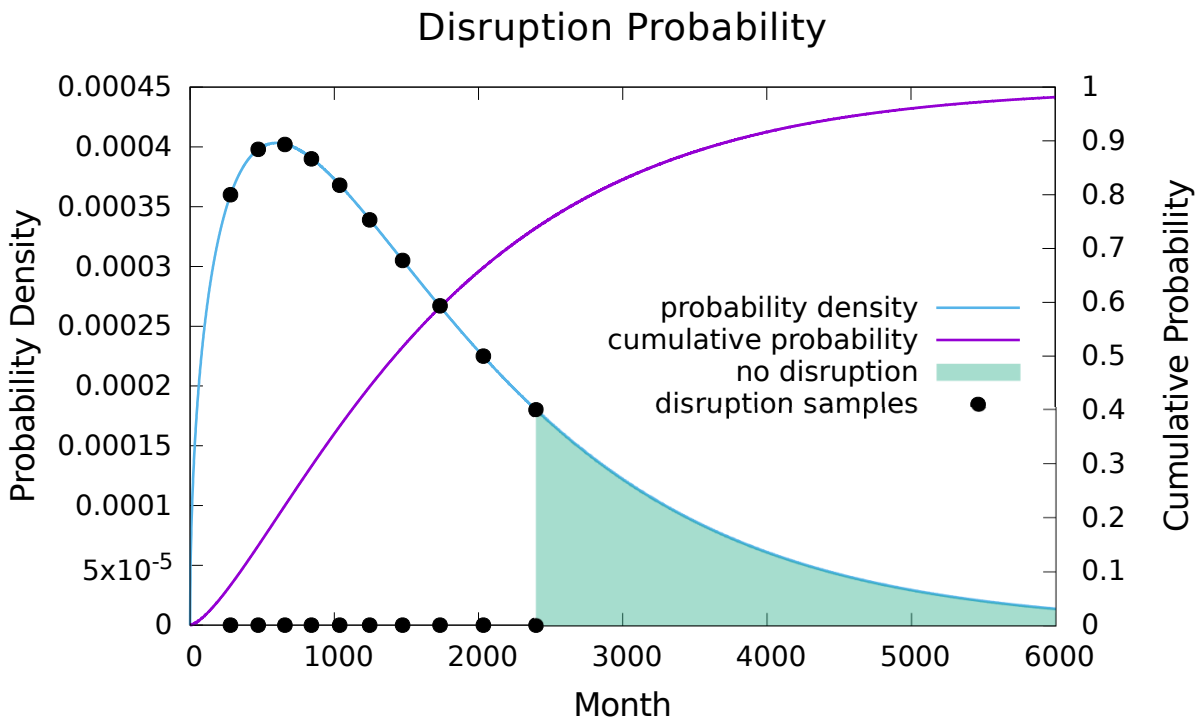

Figure 5.1: The disruption probability density function is a scaled gamma distribution (i.e. Equation 5.14). Note that the simulation duration is 2,400 months; times after month 2,400 are beyond the planning horizon and represent no disruption occurring.

The integral for $0 \leqslant t_d < 2400$ is approximately 0.739. Thus, for this scenario, there is approximately a 26.1% chance that the disruption occurs outside the planning horizon (i.e. no disruption occurs). The distribution, both density and cumulative, are shown in Figure 5.1 along with the no-disruption probability represented after the simulation ends.

Computing H for a continuous disruption PDF for a deployment schedule D is done with Equations 5.7 and 5.6. The PDF was divided into ten equi-probable sections for which the right end-points were used as sampling in Equation 5.6 to approximate $S^*(D, t_d)$. The

Disruption Time Samples

Sample	Disruption Time (month)
1	280
2	473
3	657
4	842
5	1037
6	1245
7	1474
8	1733
9	2034
10	2400 (no disruption)

Table 5.1: Right endpoints from ten equi-probable sections of the PDF are selected for use in the analysis. Only the partial PDF prior to the end of the 200-year time horizon is used (i.e. excluding the no-disruption region). The final sample point at month 2400 is equivalent to no disruption occurring.

selected points are shown in Figure 5.1 and listed in Table 5.1. Computational resources required for the analysis scale linearly with the number of these sample points. This and the relatively tame curvature of S^* make a modest sampling density appropriate for this work. For Equation 5.7, a simple midpoint rule numerical integration is performed with the domain divided into several thousand sub-intervals since evaluation of both S^* and the PDF are computationally cheap operations.

5.3 Results

The data and analysis for Experiment 3 consists primarily of the best deployment schedules and objective values from several optimization runs and associated details from using these deployment schedules in various disruption scenario simulations. The results are presented in 3 main sub-sections:

- **Single Disruption Optima** (5.3.1): Details for optimum deployment schedules for each of the ten disruption time sample points. These optima are used as a reference

for approximating and judging hedging effectiveness of deployment schedules.

- **Hedging** (5.3.2): Quantification and discussion of deployment schedules' hedging value within the disruption probability distribution.

- **Alternate Objective Optimization Runs** (5.3.5): Investigation of effects of modifying the objective function in an attempt to avoid unrealistic artifacts in optima.

Several optimization runs were performed as part of this experiment:

- One optimization run for each of the ten sampled disruption times (see table 5.1) to determine each of the ten $D^*(t_d)$ schedules and their respective $O^*(t_d)$ values. Each objective evaluation consisted of a single simulation with a single disruption at the determined time. Equation 5.12 was used as the objective function.

- One hedging optimization run searching for D^*_H using the $O^*(t_d)$ values found from the just mentioned single-disruption optimizations for approximating S^* in Equation 5.9. Each objective evaluation consisted of ten simulations – one for each sampled disruption time – whose results were combined to evaluate H using Equation 5.7.

- One optimization run to find the best post-disruption deployments in each sampled disruption time for all deployment schedules being compared for hedging effectiveness – each $D^*(t_d)$ and the D^*_H from the hedging optimization totaling 11 optimization runs. This resulted in $([\#disruptimes] - 1) * [\#deploymentschedules] + 1 = 9 * 11 + 1 = 100$ more optimization runs. The results of these runs were used to determine optimal deployments $R^*[D, t_d]$ and associated $O(R^*[D, t_d], t_d)$ value for a more accurate approximation of H for each of the 11 deployment schedules 10 disruption times. In cases for D^* build schedules, because $R^*[D^*(t_d), t_d] = D^*(t_d)$ only 9 optimization runs were required.

- One optimization run for disruptions 5–10 using an alternate objective function. Each objective evaluation consisted of a single simulation with a single disruption at the determined time. Equation 5.15 was used as the objective function.

In total, 116 optimization runs were performed comprising over 7 million simulations and over 100,000 CPU-hours of runtime. Including troubleshooting and other re-runs brings the experiment 3 totals to approximately 13 million simulations and 200,000 CPU-hours.

5.3.1 Single Disruption Optima

Each $D^*(t_d)$ deployment schedule found was run in a simulation with a t_d disruption time. The details from each of these ten simulations are shown in Figures 5.2–5.11 and are discussed below. The disruption time for each scenario/figure is indicated by a vertical black bar. The associated $O^*(t_d)$ values for each of the ten disruption time sample points are summarized in Table 5.2.

Single Disruption Objective Values

Disruption Year (t_d)	Best Objective Value
23	0.654
39	0.634
55	0.618
70	0.586
86	0.545
104	0.481
123	0.384
144	0.209
170	0.152
200	0.142

Table 5.2: The $O^*(t_d)$ objective values for the best deployment schedules from the optimization runs for each sampled disruption time. Disruption times have been rounded to the nearest whole year.

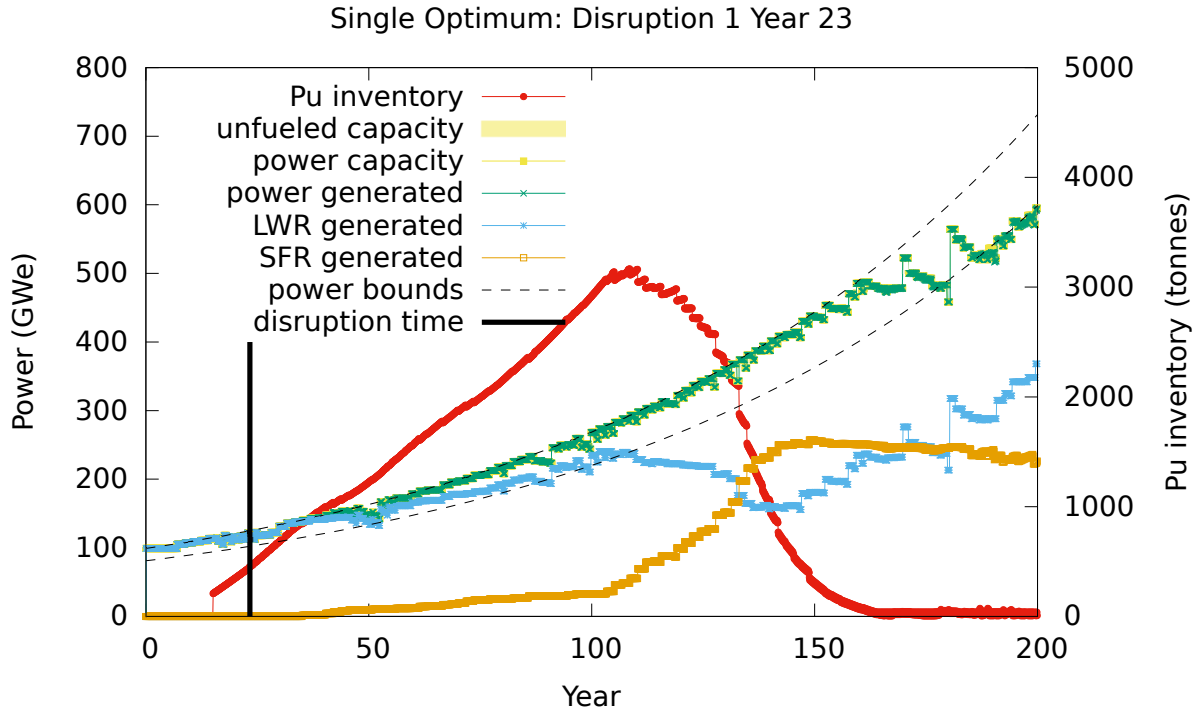

Figure 5.2: Power capacity and generation (y1 axis) plotted with Pu inventory available for use in fresh SFR fuel (red curve, y2 axis). The objective value is 0.654. Uniquely, the strategy used here delays SFR deployments until half-way through the simulation, at which point it builds beyond the equilibrium LWR-SFR support ratio temporarily until gravitating toward an equilibrium ratio with zero excess Pu inventory. The power generation drop from the upper bound to the lower capacity bound is a boundary effect that occurs in many D^* schedules and is discussed in the section text.

Because of the nature of the disruption, given fore-knowledge of the disruption time the best achievable objective value monotonically decreases with increasing disruption time; earlier disruptions result in worse outcomes. This must be the case because the disruption serves to reduce the system's capacity to generate the benefit portion of the objective function (i.e. SFR energy) with no other compensatory effect. The year 200 disruption time scenario effectively has no disruption and represents a best-case outcome. A scenario consisting entirely of LWRs would have an objective value of 1.0.

Figures 5.2 and 5.3 both show the disruption occurring very early – years 23 and 39 respectively. Despite this, however, the optimum deployment schedules for each follow qualitatively different strategies. $D^*(t_d = 23)$ holds off on SFR deployments until about

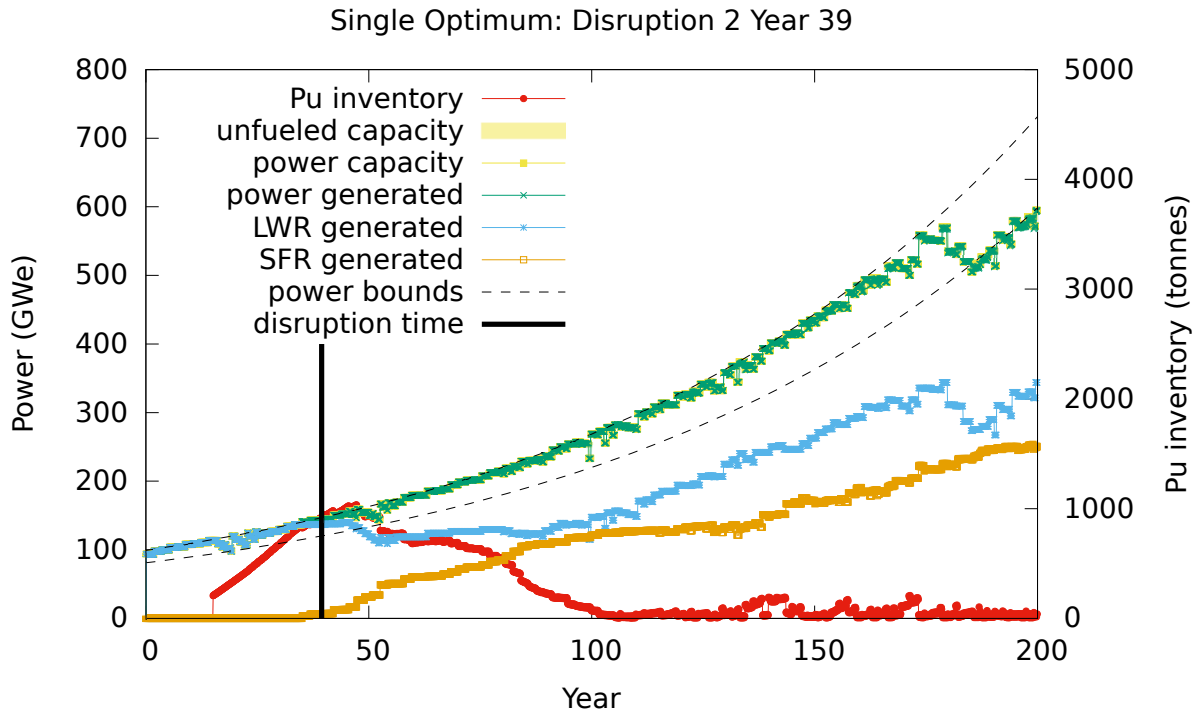

Figure 5.3: Power capacity and generation (y1 axis) plotted with Pu inventory available for use in fresh SFR fuel (red curve, y2 axis). The objective value is 0.634. The Figure 5.2 strategy of delaying SFR deployments is only optimal for a narrow range of early disruption times. For a year 39 disruption, it is better to approach and maintain an equilibrium LWR-SFR support ratio as quickly as possible.

year 100, at which point it rapidly builds SFRs beyond LWRs' ability to supply fresh fuel to them. While the accumulated Pu reserves are being depleted during and following these SFR builds, new deployments shift more heavily to LWRs approaching an equilibrium support level between the two which is not quite reached by simulation end. In $D^*(t_d = 39)$, rather than the Pu stock-piling and rapid SFR deployment, the strategy is to build SFRs at the maximum rate possible as soon as they become available; this maintains Pu reserves at or near zero for the entire simulation.

In many of the early-disruption scenarios, total deployed power capacity runs along the +10% upper bound until the last few decades after which it drops down to the lower bound. Figures 5.2–5.7 all show this quite clearly. This is caused by both decommissioning echoes of LWRs and the scenario's finite time-horizon. As reactors are decommissioned,

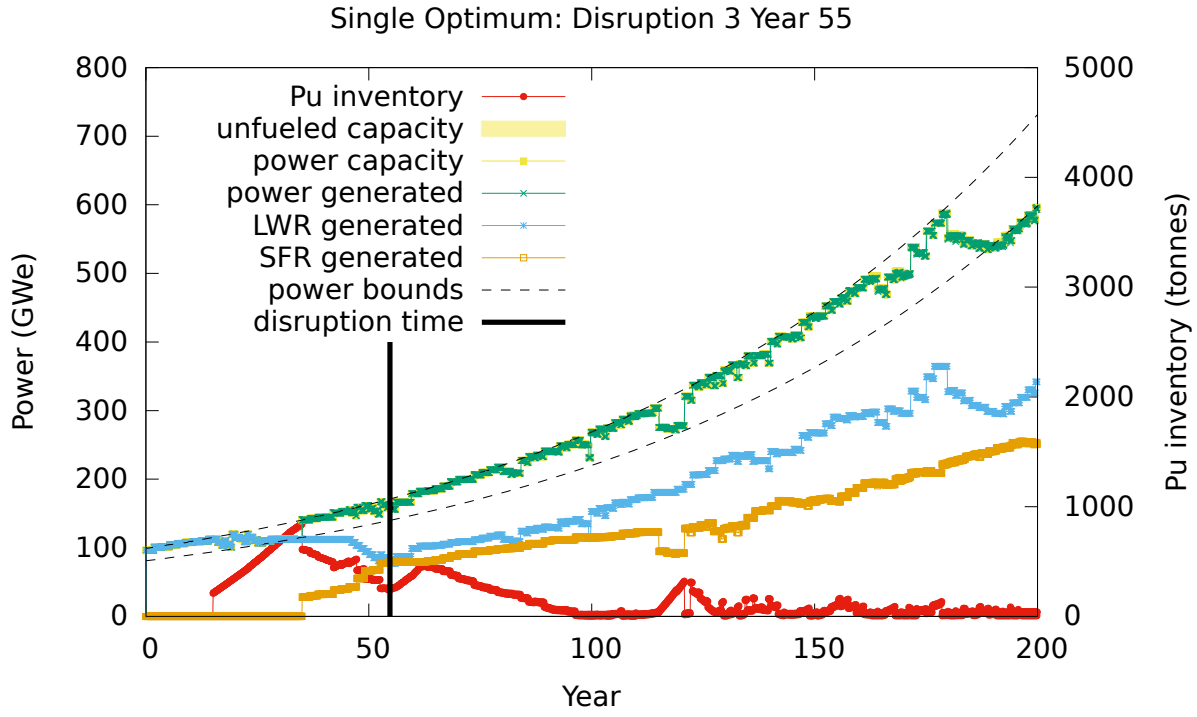

Figure 5.4: Power capacity and generation (y1 axis) plotted with Pu inventory available for use in fresh SFR fuel (red curve, y2 axis). The objective value is 0.618. The small discontinuity/drop in SFR deployments that occurs in year 115 occurs exactly 80 years (1 reactor lifetime) after SFRs become available for deployment. This can be seen in Figures 5.5–5.11. In many of the figures, a corresponding sudden increase in SFR generation occurs around year 120 introducing reactors that won't have to be replaced until after the simulation ends. This is a boundary effect that the optimizer utilizes to improve the objective slightly.

they discharge a full core of fuel rather than the usual 1-batch. This extra fuel allows delaying the replacement of decommissioned LWRs without forcing dependent SFRs offline immediately. Since the initial LWRs are decommissioned relatively close together, the echoes of the initial decommissioning wave continue throughout the simulation. The optimization process is able to capitalize on such a decommissioning wave around years 150 to 175 depending on the scenario. The optima effectively drive toward an SFR deployment deficit - meaning that the delayed power capacity deployment will cause many new reactors to need to be deployed around the same time which will require a higher LWR-SFR ratio to support new SFR full cores. But since the scenario ends at year 200, the optimizer "cheats"

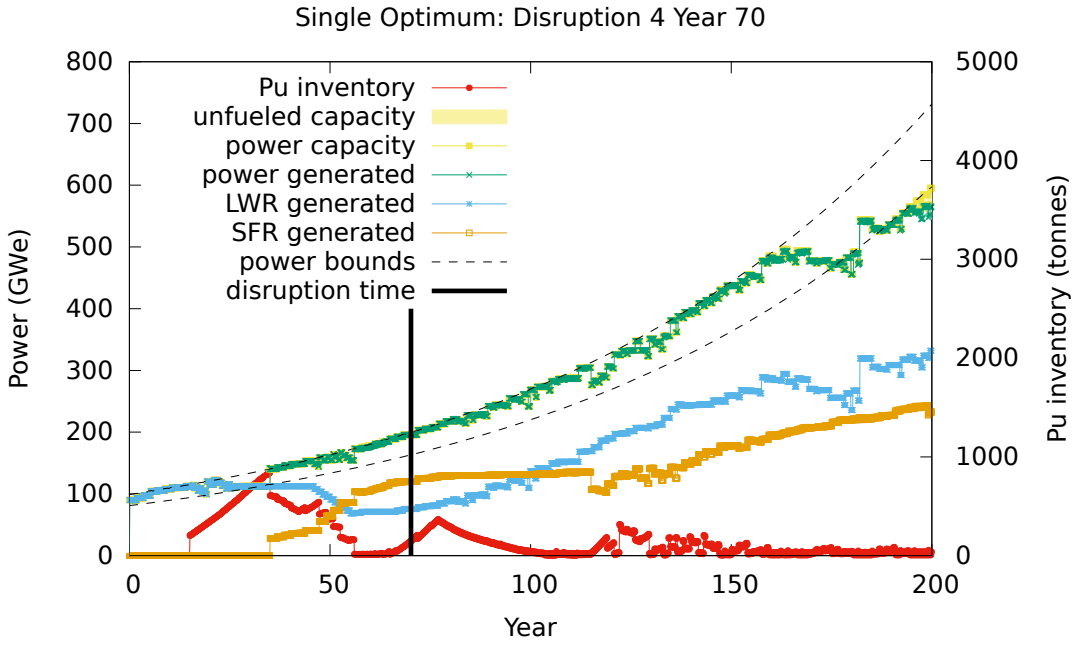

Figure 5.5: Power capacity and generation (y1 axis) plotted with Pu inventory available for use in fresh SFR fuel (red curve, y2 axis). The objective value is 0.586.

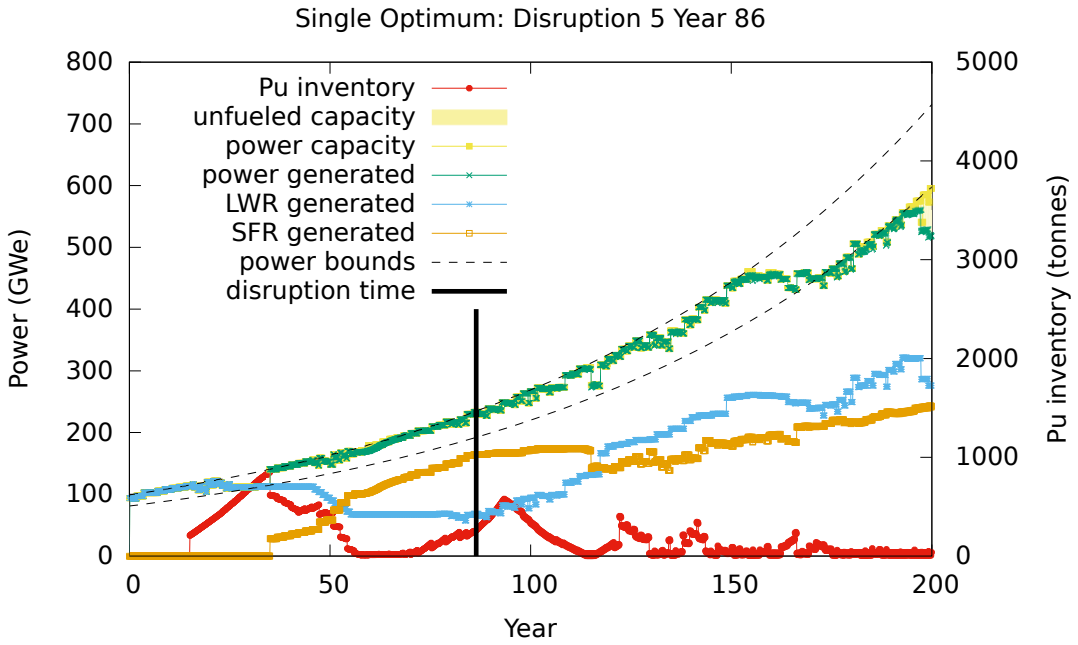

Figure 5.6: Power capacity and generation (y1 axis) plotted with Pu inventory available for use in fresh SFR fuel (red curve, y2 axis). The objective value is 0.545.

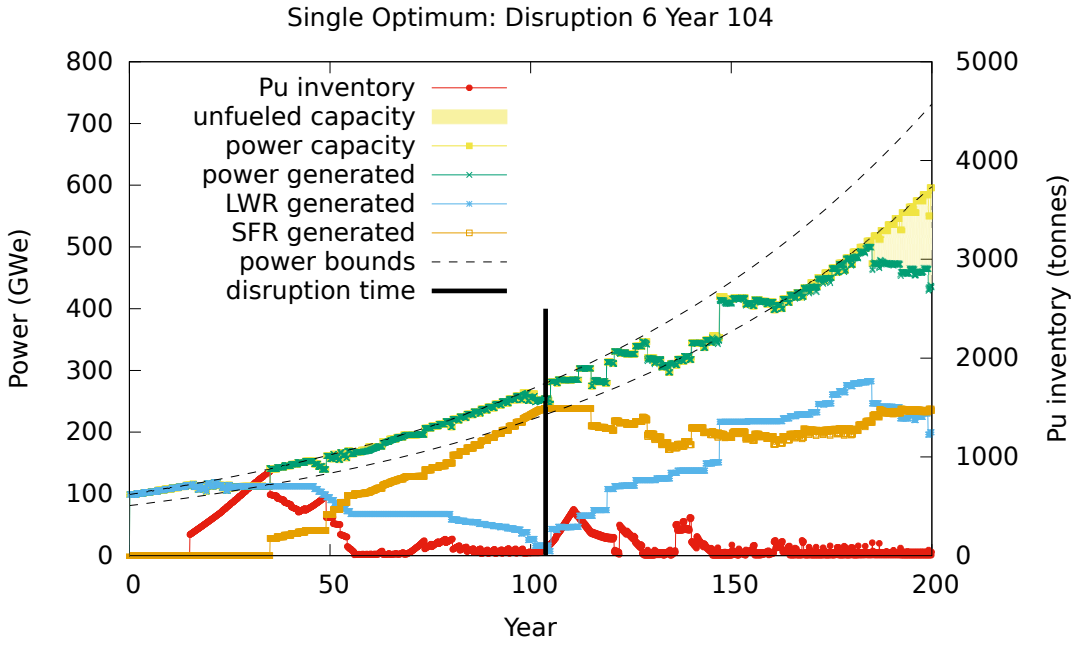

Figure 5.7: Power capacity and generation (y1 axis) plotted with Pu inventory available for use in fresh SFR fuel (red curve, y2 axis). The objective value is 0.481.

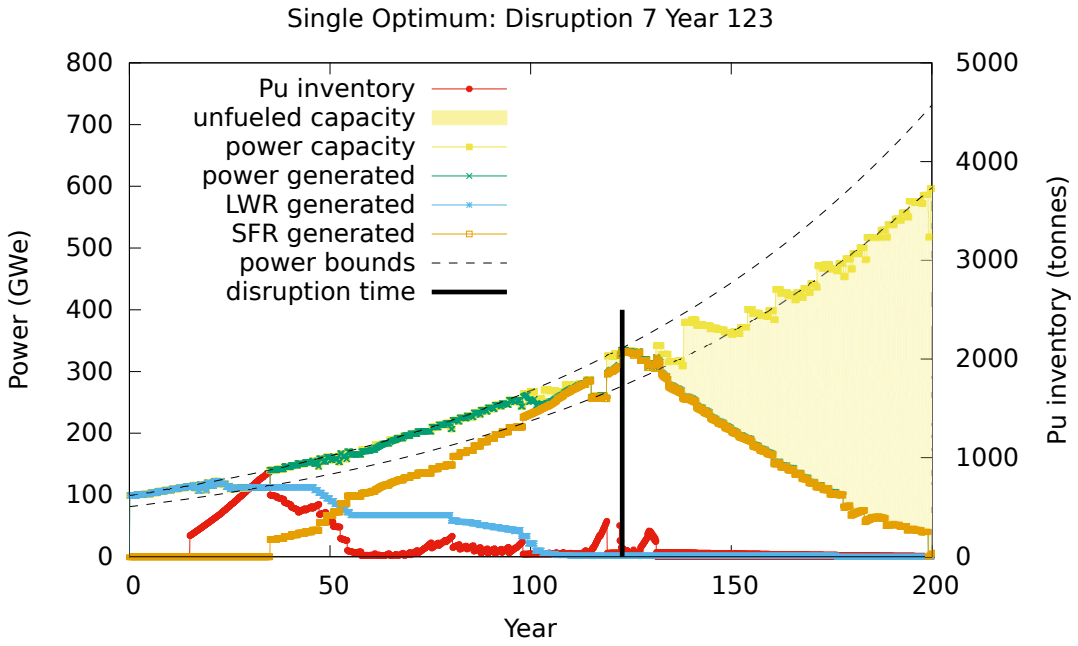

Figure 5.8: Power capacity and generation (y1 axis) plotted with Pu inventory available for use in fresh SFR fuel (red curve, y2 axis). The objective value is 0.384. Generated power falls far below the capacity bound. This is a limitation of the objective function and is discussed in the section text and investigated further in Section 5.3.5.

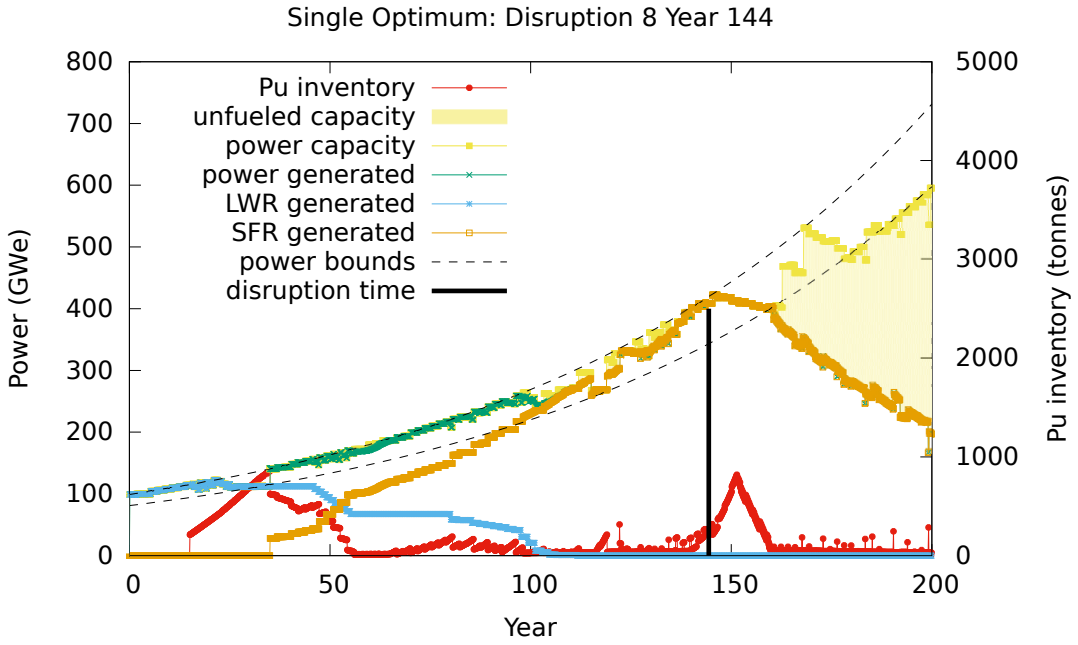

Figure 5.9: Power capacity and generation (y1 axis) plotted with Pu inventory available for use in fresh SFR fuel (red curve, y2 axis). The objective value is 0.209.

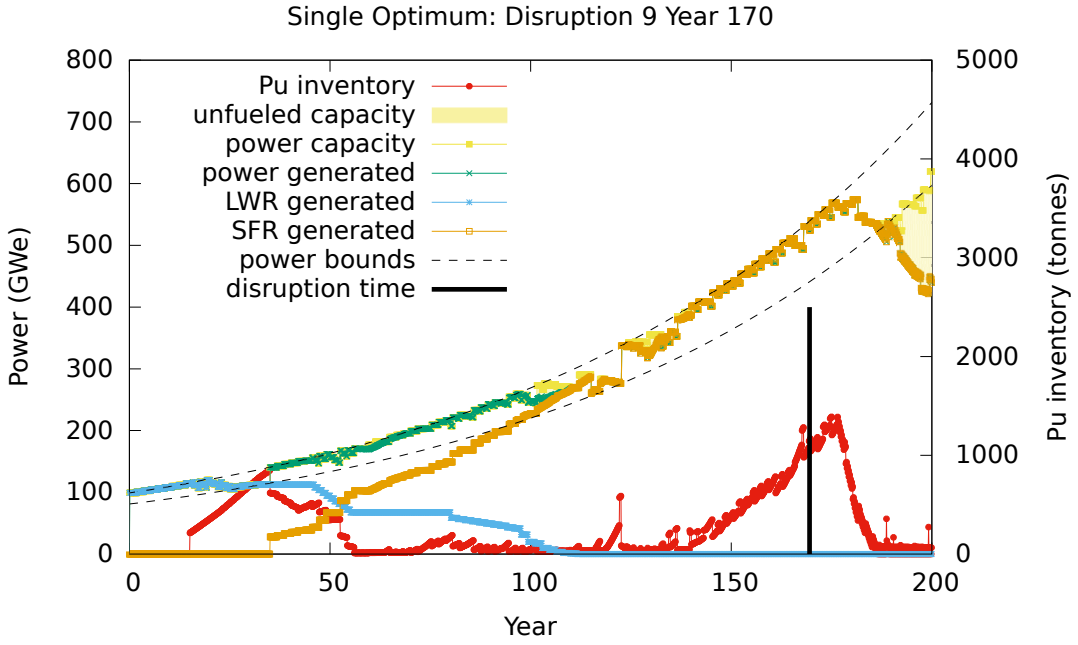

Figure 5.10: Power capacity and generation (y1 axis) plotted with Pu inventory available for use in fresh SFR fuel (red curve, y2 axis). The objective value is 0.152.

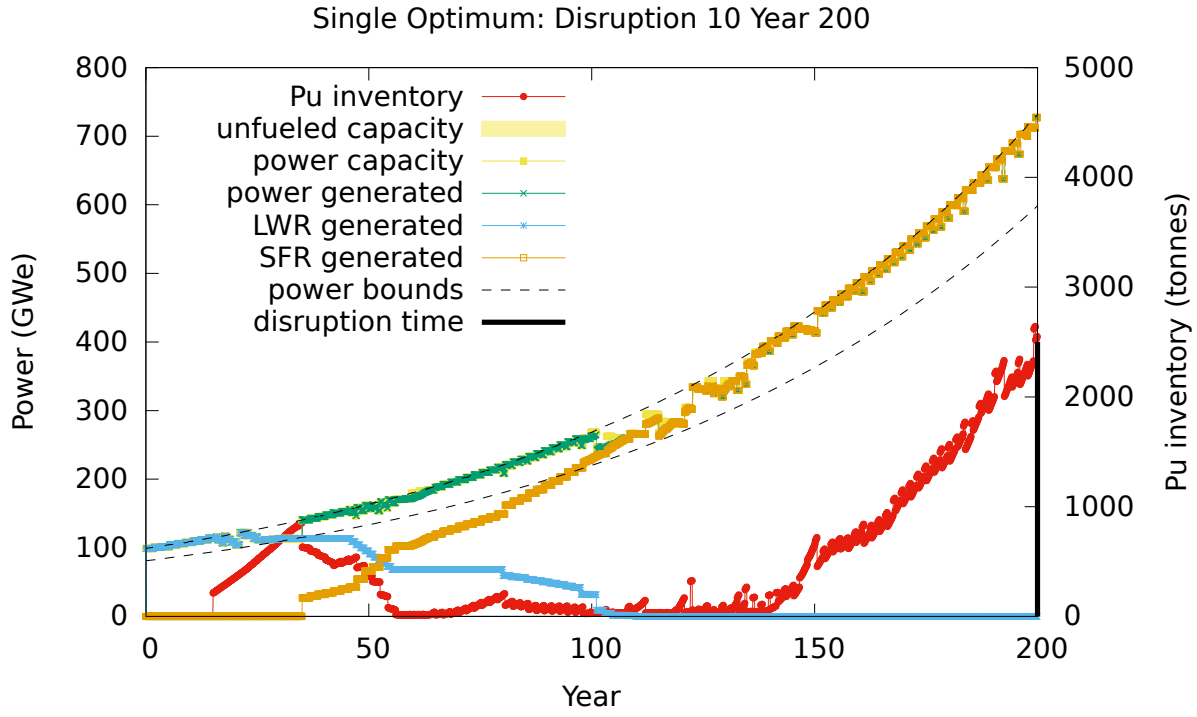

Figure 5.11: Power capacity and generation (y1 axis) plotted with Pu inventory available for use in fresh SFR fuel (red curve, y2 axis). The objective value is 0.142. With no disruption, the optimal strategy is to transition to SFRs as quickly as possible and maintain deployed capacity along the upper bound.

and takes advantage of this finite time-horizon.

This effect can be confirmed by looking at the age-distribution of reactors under each of these disruption scenarios at different snapshots in time as shown in Figures 5.12, 5.13, and 5.14. For each of the disruption scenarios represented in those figures, there is a shift toward an older age distribution for SFRs that occurs during the last several decades of the simulation. And the average SFR age also increases significantly. The most natural way to avoid this effect would be to increase the simulation duration a number of decades beyond the time frame of interest — at least 1 reactor lifetime (i.e. 80 years); the final decades could then be ignored during analysis. This artifact illustrates the importance of considering the age distribution of facilities in general and can provide useful insights.

Due to the simplicity of the disruption and objective function, another unrealistic artifact presents itself strongly in the optimum build schedule for a year 123 disruption time. Details

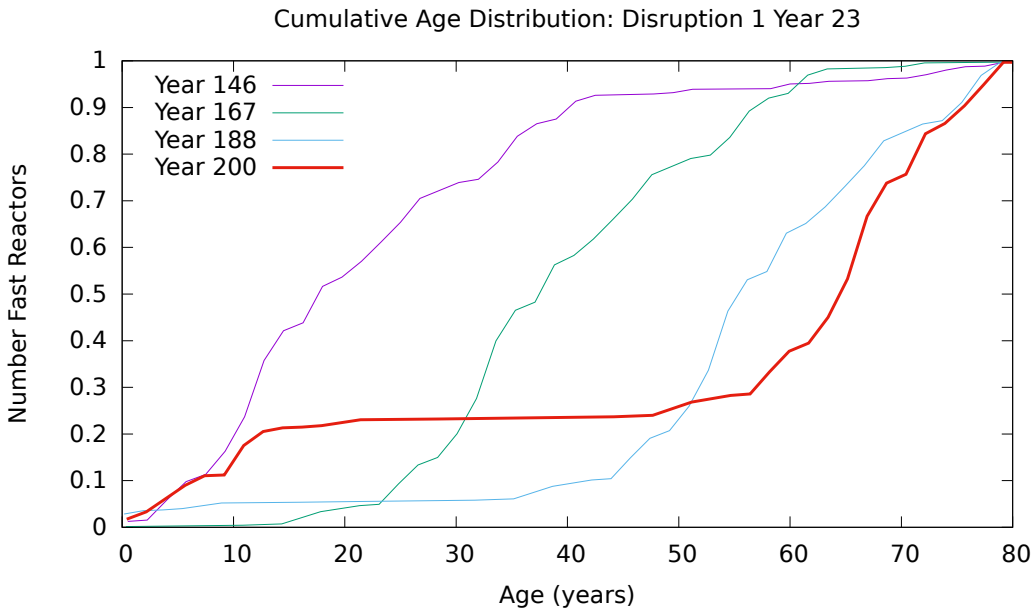

Figure 5.12: SFR age distributions at several time snapshots for the simulation using the best deployment schedule for disruption 1 in a disruption 1 scenario. The distributions at later times in the simulation (i.e. years 188 and 200) are skewed toward more older reactors, illustrating the finite time horizon boundary effect

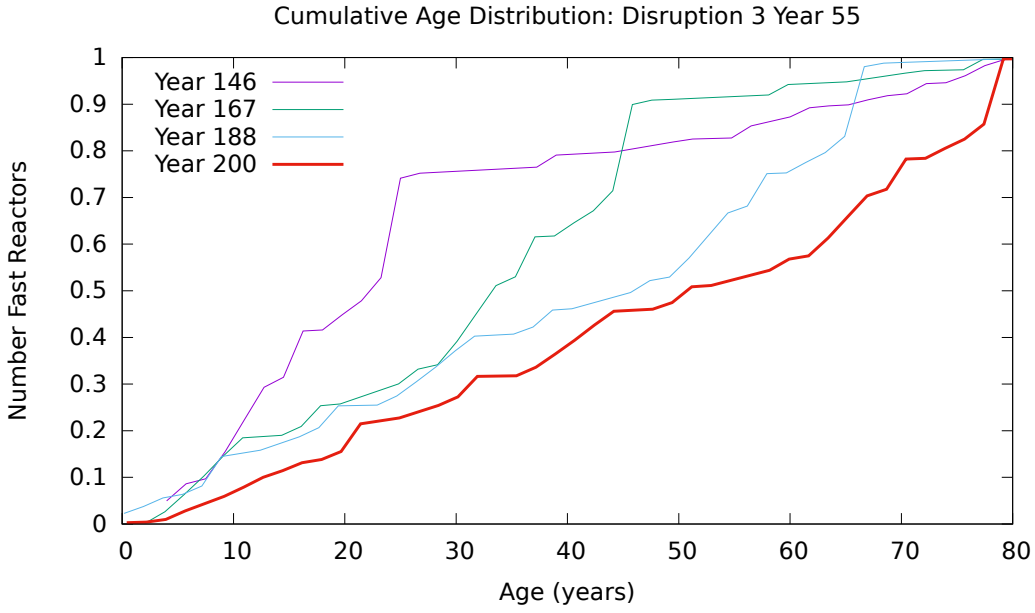

Figure 5.13: SFR age distributions at several time snapshots for the simulation using the best deployment schedule for disruption 3 in a disruption 3 scenario. The distributions at later times in the simulation (i.e. years 188 and 200) are skewed toward more older reactors, illustrating the finite time horizon boundary effect

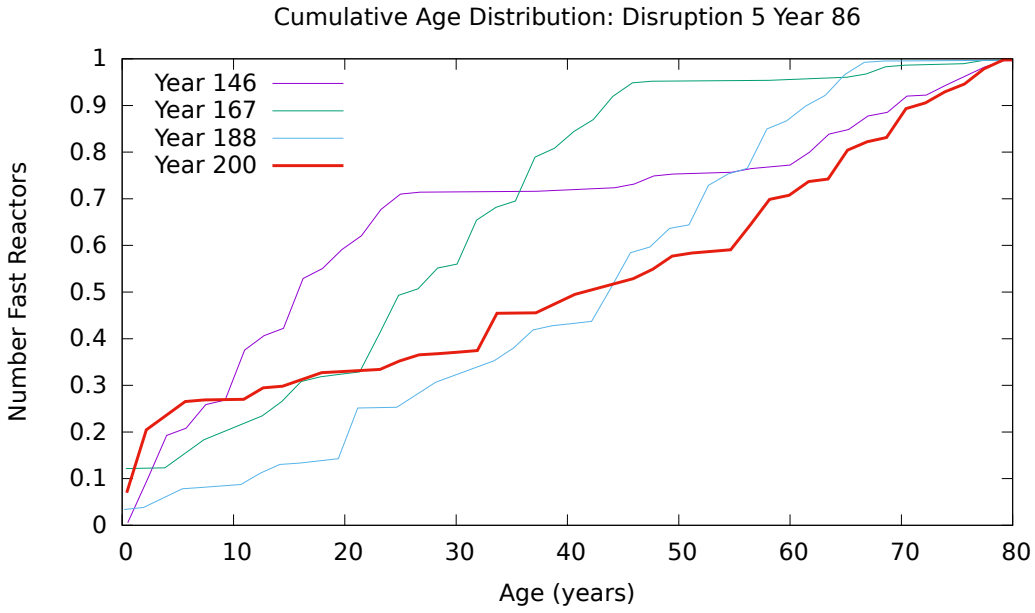

Figure 5.14: SFR age distributions at several time snapshots for the simulation using the best deployment schedule for disruption 5 in a disruption 5 scenario. The distributions at later times in the simulation (i.e. years 188 and 200) are skewed toward more older reactors, illustrating the finite time horizon boundary effect

for this disruption scenario and build schedule are shown in Figure 5.8. The optimum build schedule generates a large number of unfueled fast reactors. This occurs because the penalty for unfueled reactor capacity is not large enough to overcome the penalty for generating LWR energy. So despite the large reduction in available fuel for SFRs post disruption, the optimum strategy involves "coasting" on good objective components (i.e. SFR energy) accumulated prior to the disruption. A modified objective value was used in a few optimization runs to investigate this artifact in greater detail in order to determine potential effects on the hedging analysis. This is discussed later in Section 5.3.5.

5.3.2 Hedging Strategy Deployments

A hedging optimization run was performed using Equation 5.7 as the objective and Equation 5.9 as the S^* approximation. This generated a candidate D^*_H deployment hedging strategy. This discovered deployment strategy is shown in Figures 5.15 and 5.16 along with

the single-disruption $D^*(t_d)$ deployment schedules.

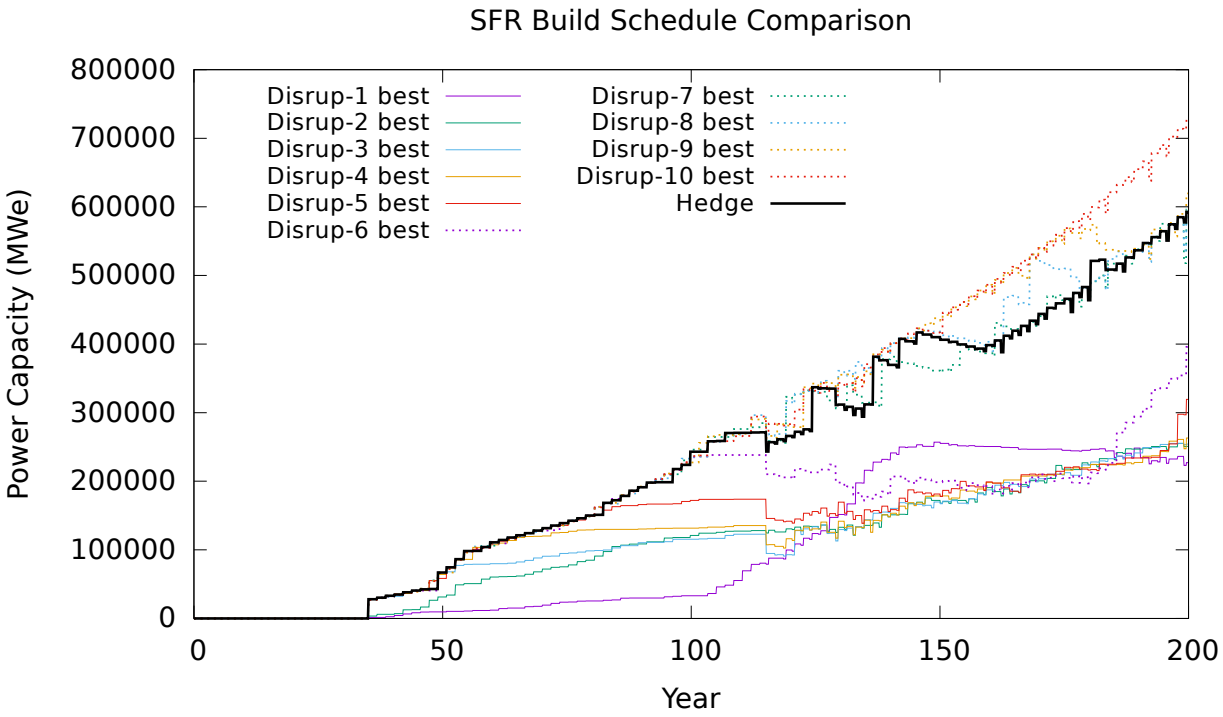

Figure 5.15: Installed SFR power capacity over time for the best deployment schedules in each of the disruption sample points along with the best optimized hedging deployments. Optimum SFR deployments for disruptions 7–10 are quite similar to the optimized hedging schedule. Three strategy groups are visible: the disruption 1 schedule delays SFR deployments prior to a sudden overbuild, the disruptions 2–6 schedules gradually introduce SFRs maintaining an approximately equilibrium LWR-SFR support ratio, and the disruptions 7–10 and hedging schedules transition to full SFRs as quickly as possible.

Figure 5.15 shows the deployed SFR capacity over time for the best deployment schedule for each single disruption time. Note that there is a great deal of similarity between the $D^*(t_d = 123 \text{ to } 200)$ deployment schedules; they do exhibit some differences toward the end of the 200 years due to the age-distribution shifting discussed earlier and shown in Figures 5.12–5.14. Corresponding LWR capacity curves are shown in Figure 5.16. This figure makes it more obvious that the best build schedules for the several disruption times are extremely similar prior to their respective disruptions — at which point LWR deployments stop their downward trend and begin increasing. And the best hedging strategies just follow the downward trend toward LWR phaseout, only diverging if/when

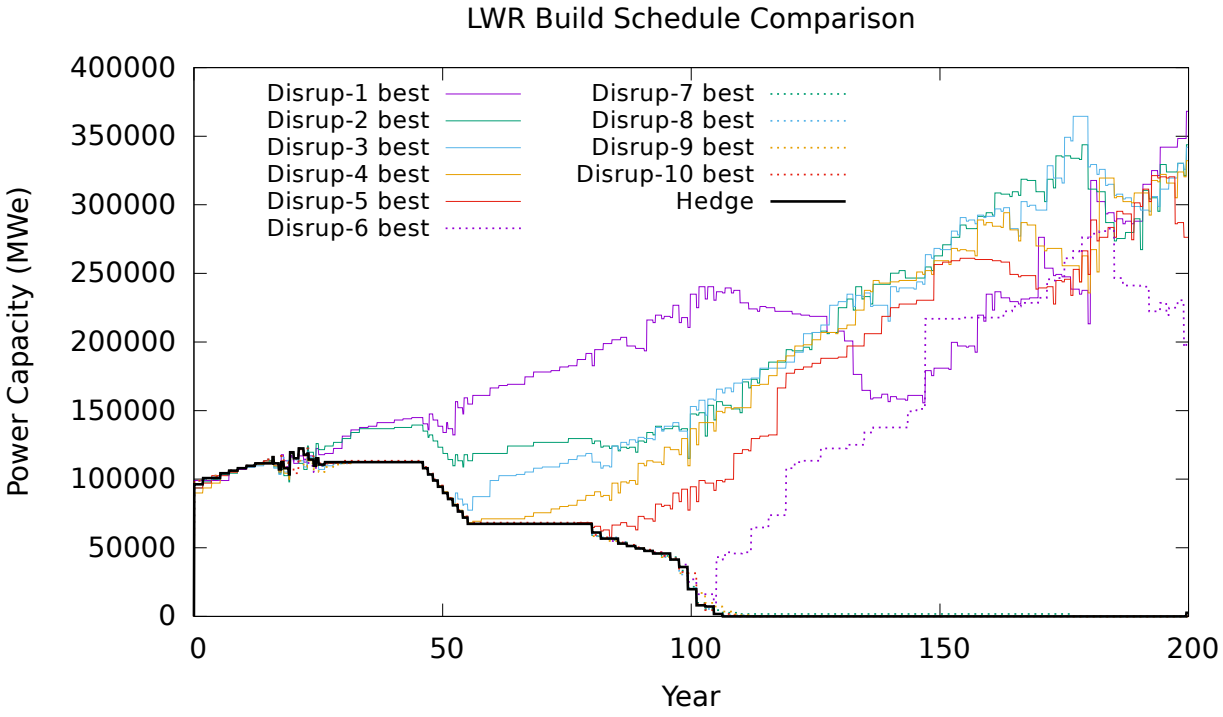

Figure 5.16: Installed LWR power capacity over time for the best deployment schedules in each of the disruption sample points along with the best optimized hedging deployments. The best hedging strategy is quick phase-out of LWRs and toward an all SFR fleet. The best schedules for disruptions 7–10 follow the hedging schedule closely pre-disruption. Best schedules for disruptions 1–6 also follow the hedging strategy pre-disruption before reversing toward an increase in LWR deployments, but due to the early disruption time frame the post-disruption behavior diverges too far from the hedging strategy to provide good all-round outcomes.

a disruption occurs.

5.3.3 Hedging Strategy Evaluation

The hedging optimization process generates a strategy that specifies pre-disruption deployments. Post-disruption, deployments are allowed to and expected to diverge from the ones specified by the hedging strategy. This is not only realistic, but is part of the hedging methodology defined and used in this exercise. Determining the optimal post-disruption responses is necessary to fully assess the hedging effectiveness of a deployment strategy. For the D^*_H strategy identified through the hedging optimization run the post-disruption

deployment response was computed through a series of optimization runs for each of the ten disruption times. This is effectively determining $R^*[D^*_H, t_{d,i}]$ for each disruption time i which can then be used to calculate corresponding actual S^* values. The computed S^* values are shown in Table 5.3. By the definitions of $D^*(t_d)$, D_H, and R^* the true $O(R^*[D^*_H, t_{d,i}], t_{d,i})$ objective value can never be better (i.e. lower) than the true $O(D^*(t_{d,i})$ objective value. This is because the $D^*(t_{d,i})$ schedules are each optimized individually with a priori knowledge of a single, specific disruption time. Any observations contrary to this would indicate imperfect optimization convergence.

Hedging Sub-objective Values

Disruption Year (t_d)	Best Achievable Objective	
	$O(D^*(t_d), t_d)$	$O(R^*[D^*_H, t_d], t_d)$
23	0.654	0.654
39	0.634	0.634
55	0.618	0.618
70	0.586	0.586
86	0.545	0.545
104	0.479	0.481
123	0.384	0.388
144	0.209	0.214
170	0.152	0.160
200	0.142	0.155

Table 5.3: The $O^*(t_d)$ objective values for the best deployment schedules compared to the best achievable objective using the D^*_H deployment schedule from the hedging optimization with post-disruption optimized deployments. Disruption times have been rounded to the nearest whole year.

As both a reference and because the work had already been done to generate them, the $D^*(t_d)$ deployment schedules were also selected for use and comparison in this hedging effectiveness evaluation. Like the D^*_H deployment schedule, each of the $D^*(t_d)$ schedules were also used in a set of ten optimization runs each to determine $R^*[D^*(t_{d,i}), t_{d,j}]$ for each disruption time combination $t_{d,i}, t_{d,j}$. The generated data were then used to determine S^* values for each of the disruption time sample points for each of these ten deployment schedules. The computed S^* values are shown in Table 5.4 and corresponding $S^*(t_d)$ curves

generated for each of these deployment strategies are shown in Figure 5.17. The curve for the best discovered strategy from the hedging optimization run (generated using the procedure from Section 5.1.2) is included as the last row in the table. Off-diagonal objective values represent the outcome of using a deployment schedule optimized for a particular disruption time in a scenario for which the disruption occurred at a different time while still allowing post-disruption modification of deployments. Among all deployment schedules considered, the best hedging strategy is actually one of the $D^*(t_d)$ schedules from the initial single-disruption optimization runs — namely disruption 10 in year 200 (i.e. no disruption).

The curves in Figure 5.17 clearly show that a hedging strategy exists that achieves close to the best possible outcome for all disruption times among all possible deployments for each disruption time (i.e. a dominating hedging strategy exists). That is roughly, $O(D^*(t_d = 200), t_{d,i}) \leqslant O(D_{any}, t_{d,i})$ for any disruption time $t_{d,i}$. This result is also reflected in Table 5.5 where the best hedging objective H (i.e. mean/expected outcome) is found belonging to $D^*(t_d = 200)$. The hedging objective value H for each deployment schedule in this analysis is shown in Table 5.5. The rows in Table 5.5 correspond directly to the rows in Table 5.4. Lower hedging objective values in this table indicate better hedging strategies.

When we look at the details of the optimum for each disruption time closely (e.g. in Figures 5.15 and 5.16), pre-disruption optimum deployments appear to be independent of the actual disruption time. For example, $D^*(t_d = 104)$ deployments and $D^*(t_d = 123)$ deployments are approximately the same between years 0 and 104. In general, scenario-objective configurations that exhibit this property will have ideal hedging strategies that are similar to $D^*(t_d)$ deployments for a larger t_d. This will not be the case for more sophisticated objectives for which optimal outcomes require pre-disruption preparation. This, in large part, explains why the hedging optimization and its associated D^*_H deployment schedule were not appreciably different from the other investigated $D^*(t_d)$ optima.

Recall that the deployment strategy is allowed to change post-disruption. Because of this, dissimilarity between a scenario A with disruption at time t_1 and scenario B with disruption at $t_2 > t_1$ does not require a (pre-disruption) hedging strategy to compromise between post t_1 scenario A ideal deployments and pre t_2 scenario B ideal deployments as long as A and B both have the same deployments before time t_1. Some of the analysis and discussion later in Section 5.3.5 shows how pre-disruption deployments might be different for different disruption times.

Best Achievable Objectives

Deployment Strategy	Disruption Time (year)									
	23	39	55	70	86	104	123	144	170	200
$D^*(t_d = 23)$	**0.654**	0.635	0.636	0.634	0.627	0.624	0.616	0.561	0.561	0.654
$D^*(t_d = 39)$	0.654	**0.633**	0.629	0.613	0.590	0.557	0.553	0.534	0.576	0.634
$D^*(t_d = 55)$	0.655	0.635	**0.618**	0.599	0.570	0.539	0.534	0.509	0.557	0.619
$D^*(t_d = 70)$	0.655	0.632	0.619	**0.586**	0.551	0.515	0.514	0.483	0.513	0.586
$D^*(t_d = 86)$	0.656	0.637	0.618	0.587	**0.545**	0.494	0.490	0.449	0.459	0.542
$D^*(t_d = 104)$	0.654	0.634	0.618	0.588	0.545	**0.479**	0.452	0.366	0.406	0.458
$D^*(t_d = 123)$	0.656	0.633	0.617	0.588	0.547	0.482	**0.384**	0.214	0.163	0.158
$D^*(t_d = 144)$	0.653	0.634	0.618	0.588	0.544	0.480	0.383	**0.209**	0.156	0.151
$D^*(t_d = 170)$	0.654	0.633	0.617	0.586	0.547	0.483	0.387	0.209	**0.152**	0.147
$D^*(t_d = 200)$	0.653	0.634	0.618	0.588	0.545	0.480	0.384	0.209	0.152	**0.142**
D^*_H	0.654	0.634	0.618	0.586	0.545	0.481	0.388	0.214	0.160	0.155

Table 5.4: For each listed deployment schedule, pre-disruption builds remain unchanged while post-disruption builds are re-optimized (i.e. uncertainty resolution enables departure from predetermined deployments). Disruption times are specified in years rounded to the nearest whole. Green background indicates the best objective value for a particular disruption time is achieved with the deployment schedule optimized for that disruption time (i.e. on-diagonal). A yellow cell background indicates the best objective value for a particular disruption time occurring in a deployment schedule optimized for a different disruption time (i.e. off-diagonal). Diagonal entries are printed in bold text. For the duplicate optimization runs for on-diagonal cases, the best objective value from either the original single disruption runs or the post-disruption optimization runs was chosen.

The reduced Pu caused by the disruption increases (i.e. makes worse) the best achievable objective, and earlier disruptions result in worse objectives. Given this fact, it might be surprising to see that some of the S^* curves are not monotonically decreasing with disruption time. Although changes are allowed post-disruption, late disruption times

Hedging Objective Values

Deployment Strategy	Mean Objective Outcome
$D^*(t_d = 23)$	0.720
$D^*(t_d = 39)$	0.665
$D^*(t_d = 55)$	0.655
$D^*(t_d = 70)$	0.644
$D^*(t_d = 86)$	0.630
$D^*(t_d = 104)$	0.562
$D^*(t_d = 123)$	0.385
$D^*(t_d = 144)$	0.381
$D^*(t_d = 170)$	0.379
$D^*(t_d = 200)$	0.373
D^*_H	0.383

Table 5.5: The mean objective outcomes for each deployment schedule as calculated with Equations 5.7 and 5.6. Disruption times (t_d) have been rounded to the nearest whole year. Notably, the deployment schedule D^*_H discovered is a less effective hedging strategy than some deployment schedules that were simply optimized for specific, single disruption times.

prevent diverging from a strong mismatch between a schedule optimized for one disruption time and used in a very different disruption time. This causes the non-monotonically decreasing behavior for the disruption 1–6 curves. Another way to think about these non-decreasing points on the disruption 1–6 curves is their correspondence to the upper-triangular entries in Table 5.4. They represent scenarios where the deployment schedules were developed with responses to a phantom disruption when the actual disruption happened later.

It is also of interest to take a more detailed look at some of the scenarios corresponding to these off-diagonal cells in Table 5.4 in order to show what happens when a build schedule optimized for a particular disruption time is used in a scenario with a different disruption time (while allowing for post-disruption deployments to change). Figure 5.18 shows simulation detail for a deployment schedule optimized for a late disruption (year 170) used in an early-disruption scenario (year 23). Since deployments are allowed to adjust following the disruption, the $D^*(t_d = 170)$ build schedule was easily able to match the build trajectory

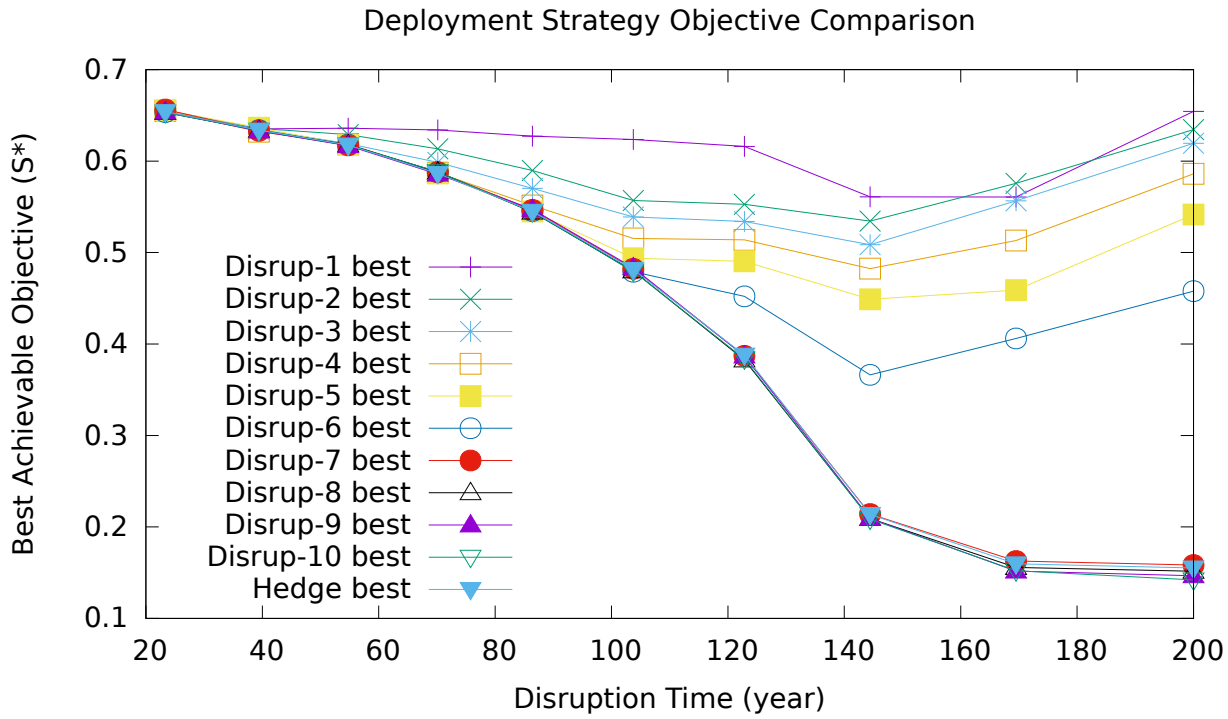

Figure 5.17: Best achievable objective (lower is better) for a simulation in each disruption sample time (including post-disruption optimized deployments) for the best deployment schedules in each of the disruption sample points along with the hedging deployment strategy. This is effectively a plot of the data contained in Table 5.4 where each table row corresponds to a series in the plot.

of the $D^*(t_d = 23)$ and little significant difference is observed — reinforcing the idea that optimizing for a later disruption is a good hedging strategy. Figure 5.19 shows simulation detail for the opposite – a deployment schedule optimized for an early disruption (year 23) used in a late-disruption scenario (year 170). The resulting poor Pu utilization can be seen as a large available Pu inventory buildup. Deployments respond post-disruption with the construction of many SFRs, but it is too late to recover from the poor LWR-SFR ratio that existed during most of the simulation.

One of the most useful capstone artifacts of this disruption/hedging analysis is the outcome frequency distribution for different hedging strategies. Outcome distributions were generated using the best deployment schedule from each of the ten disruption times under the PDF of all possible disruption times and are shown in Figure 5.20. Each of the

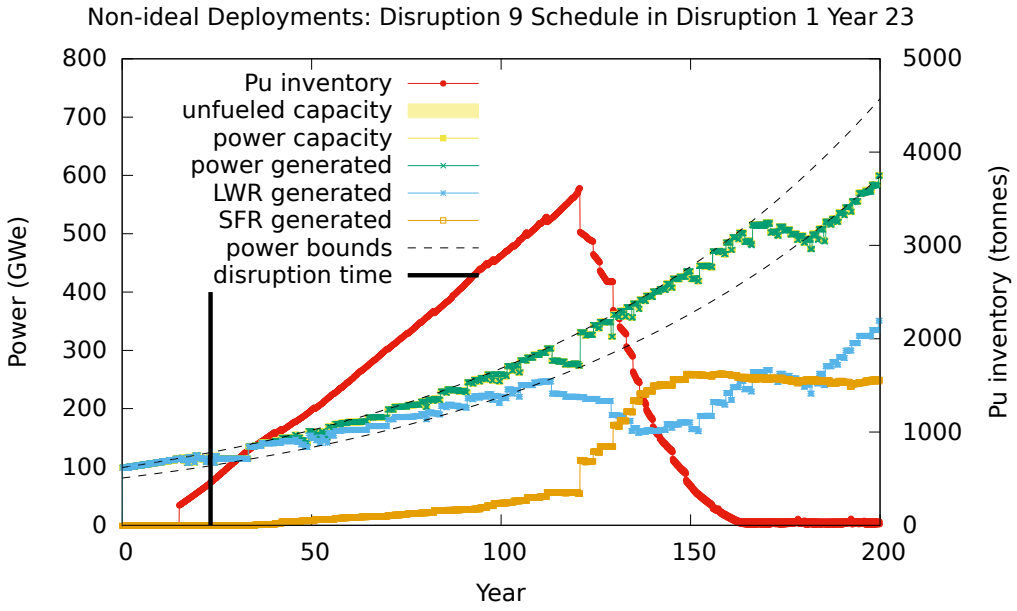

Figure 5.18: Power capacity and generation (y1 axis) plotted with Pu inventory available for use in fresh SFR fuel (red curve, y2 axis). The objective value is 0.654. The schedule is used in a scenario with a disruption time different from the one for which it was optimized.

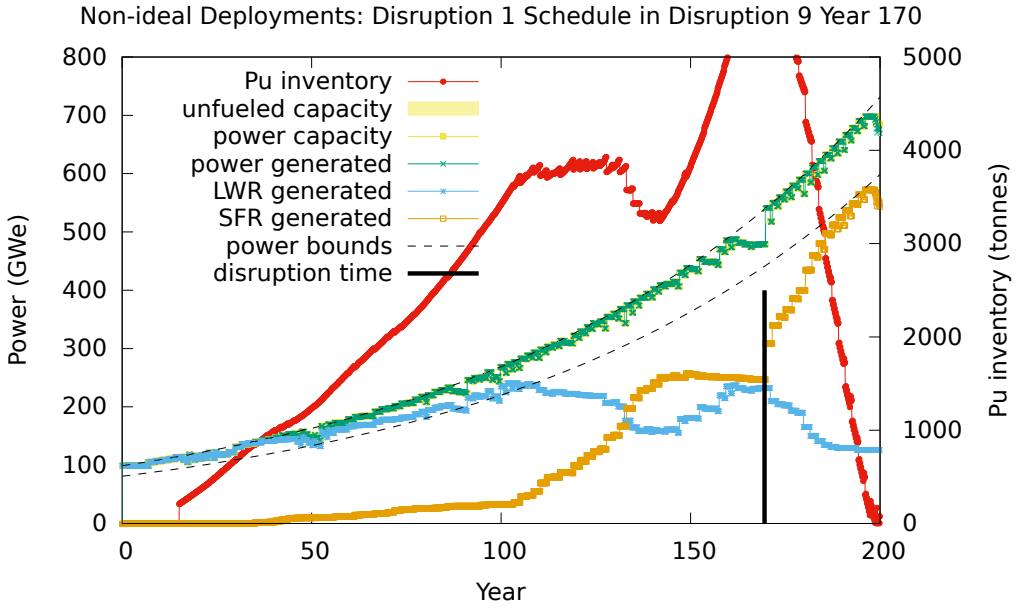

Figure 5.19: Power capacity and generation (y1 axis) plotted with Pu inventory available for use in fresh SFR fuel (red curve, y2 axis). The objective value is 0.561. The schedule is used in a scenario with a disruption time different from the one for which it was optimized. The y2 axis is maintained at the same scale as all other detail plots despite truncating some results in order to maintain consistency for comparison.

outcome distributions in the figure are generated directly from the S^* curves shown in Figure 5.17 using the disruption PDF and deterministic sampling method described in Section 5.1.3. These distributions are somewhat bimodal meaning that there are many early and many late disruption times that generate similar objective values with a relatively quick transition between worst and best objectives as the disruption time increases; Figure 5.17 shows this transition well for the $D^*(t_d = 123\text{to}200)$ and D^*_H deployment schedules.

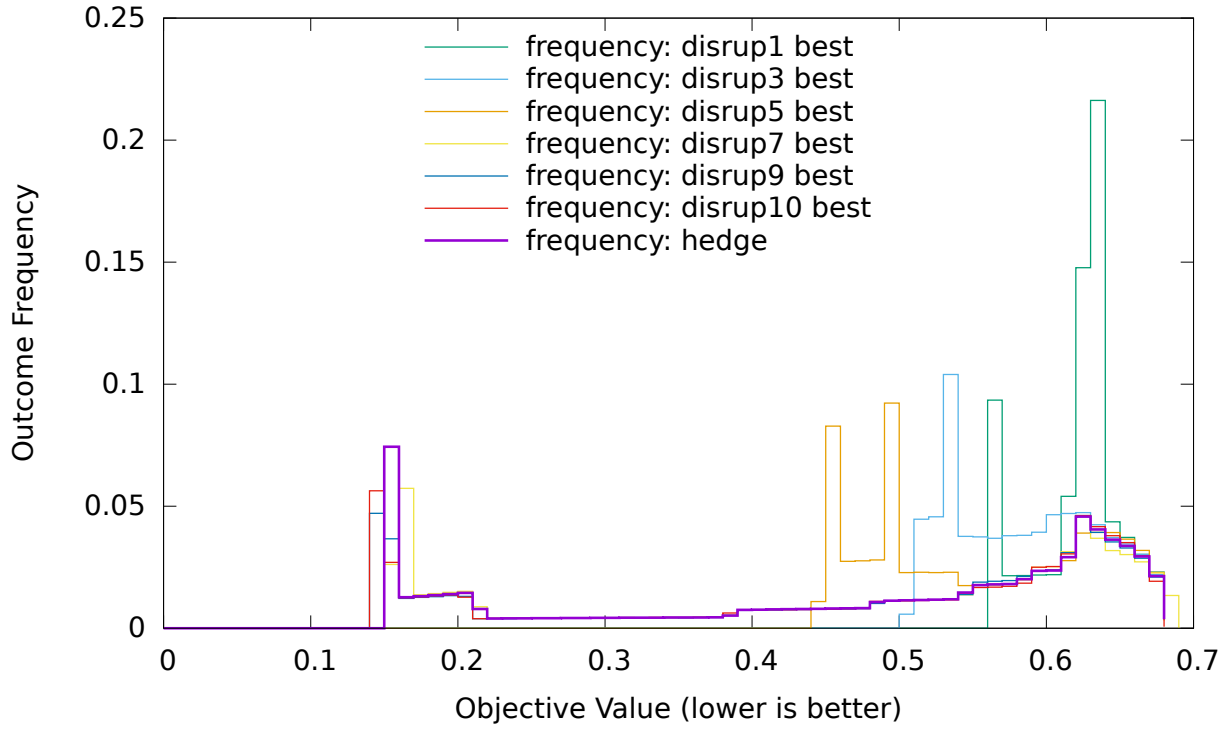

Figure 5.20: The best deployment schedules from select disruption times and the hedging schedule were used in to sample disruption times from the PDF. Best achievable objectives were then collected and the outcome frequency for each is shown. Better hedging strategies shift more outcomes to the left toward better values (e.g. hedge, disruption 10, etc.). The 26.1% no-disruption probability causes a spike of outcomes all at the left/lower edge of each distribution curve. Because this obscures other data, the no-disruption outcomes are omitted from the plot.

5.3.4 S^* Approximation Effectiveness

The hedging optimization used Equation 5.9 in order to reduce the nested component of the optimization, avoiding solving for post-disruption optimal deployments for each objective evaluation. Measuring the effectiveness of this approximation requires comparing approximated S^* values to ones using optimal solutions for post-disruption deployments (i.e. requiring a full optimization run to check a single $S^*(D, t_d)$ value. The results for such a comparison with D^*_H are shown in table 5.6 for each of the ten disruption sample points. For completeness, the table also includes S^* values using an Equation 5.8 style approximation — where post-disruption deployments are not modified at all.

Notably, the reference approximation (i.e. Equation 5.8) definitely draws an upper bound over the t_d-weighted approximation (i.e. Equation 5.9). Since the reference approximation is effectively "changing nothing", theoretically, it provides an extreme response that serves as an upper bound against which all approximations can be compared. And although modest in quantity, the data shown in table 5.6 seem to support this idea. Since the actual S^* values are optima (i.e. minima) by definition, they serve as an effective lower bound on the approximation values.

The t_d-weighted approximation values lie between the reference and actual objective values for all disruption times and are generally closer to the actual S^* values. This approximation seems to be much better than the reference, although still not particularly good – generating several values in excess of 10% above actuals. Also notable is that this approximation has inconsistent error across different disruption times (at least for this deployment schedule).

As the disruption time becomes later, the difference between post-disruption related approximations and actual post-disruption optimum decisions shrinks with the declining number of post-disruption variables. This causes error associated with most forms of approximation (including all ones discussed here) to generally decrease with increasing disruption time for *any* deployment schedule and objective function combination. This

D_H^* **Sub-objective Approximations**

Disruption	S^* Approximations		
Time (year)	Reference (Eq. 5.8)	t_d interp. (Eq. 5.9)	Actual
23	2.269	0.843	0.654
39	1.614	0.828	0.634
55	1.372	0.826	0.618
70	1.019	0.740	0.586
86	0.723	0.623	0.545
104	0.485	0.483	0.481
123	0.393	0.390	0.388
144	0.214	0.213	0.214
170	0.161	0.160	0.160
200	0.155	0.155	0.155

Table 5.6: Comparison of $S^*(D_H^*, t_{d,i})$ values computed using approximations from Equations 5.8 and 5.9 to actual S^* values computed from post-disruption optimization runs on D_H^*. Disruption times have been rounded to the nearest whole year.

phenomenon is clearly present in the data of Table 5.6.

Better quantifying this approximation error as a function of deployment schedule in addition to disruption time would help provide a better ability to quantify general approximation goodness — especially in the context of optimization. This would require many optimization runs (several for each deployment schedule sampled) sampling approximation error associated with many different deployment schedules and disruption times. One important feature to investigate would be any consistencies (or lack thereof) in the error magnitudes across these samples. Inconsistent error magnitudes and polarity will adversely affect any sort of hedging optimization attempts.

5.3.5 Alternate Objective Optimization Runs

In order to investigate artifacts present in results from Section 5.3.1 (particularly one shown in Figure 5.8), a few extra optimization runs were performed. These runs were identical to the other single disruption runs used for finding $D^*(t_d)$ build schedules except the objective function fractional component that penalized unfueled reactor capacity was squared. Equation 5.15 shows this alternate objective function. Corresponding detail results for the best deployment schedule found in each of these runs are shown in Figures 5.21, 5.22, and 5.23.

$$O_{sim} = \frac{\sum_{t \in sim} E_{t,\,LWR}}{\sum_{t \in sim} E_{t,\,tot}} \cdot \left(\frac{\sum_{t \in sim} C_{t,\,tot}}{\sum_{t \in sim} E_{t,\,tot}} \right)^2 \tag{5.15}$$

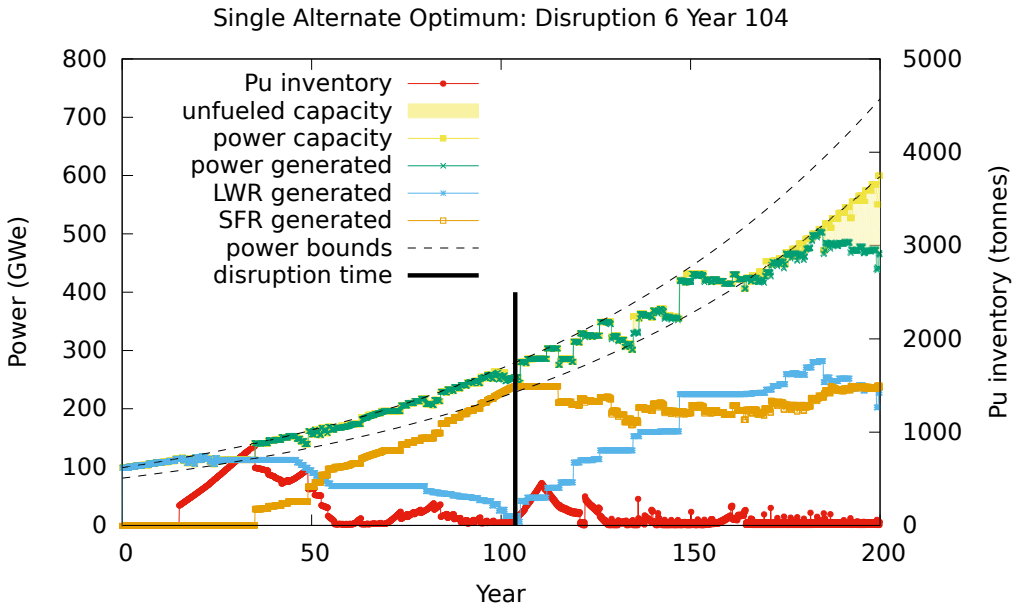

Figure 5.21: Power capacity and generation (y1 axis) plotted with Pu inventory available for use in fresh SFR fuel (red curve, y2 axis). The objective value is 0.484. This build schedule was generated using an objective function with a heavier penalty on unfueled reactors.

The optimum build schedules generated by the modified objective are compared to those generated using the original objective function in Figures 5.24 and 5.25.

Differences between the original best deployment schedules and the best ones under the

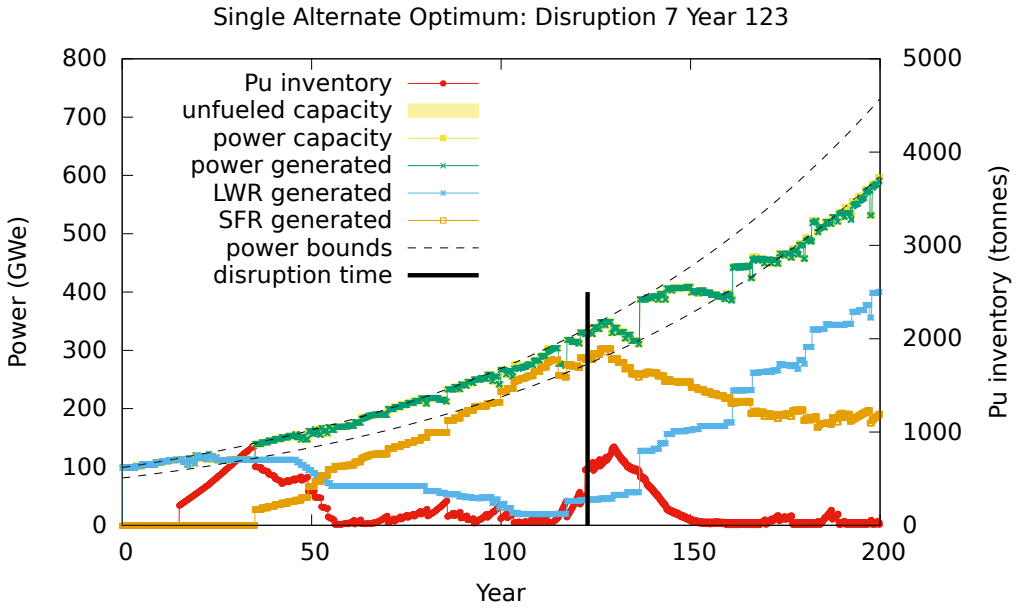

Figure 5.22: Power capacity and generation (y1 axis) plotted with Pu inventory available for use in fresh SFR fuel (red curve, y2 axis). The objective value is 0.454. This build schedule was generated using an objective function with a heavier penalty on unfueled reactors. This is the only disruption time for which the alternate objective generates significantly different objective value and deployments.

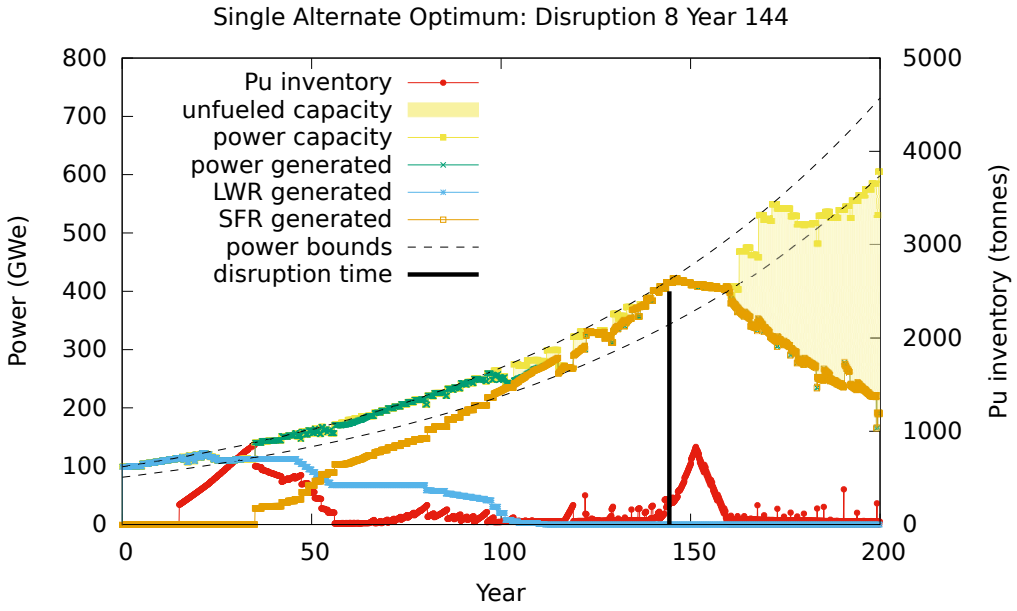

Figure 5.23: Power capacity and generation (y1 axis) plotted with Pu inventory available for use in fresh SFR fuel (red curve, y2 axis). The objective value is 0.214. This build schedule was generated using an objective function with a heavier penalty on unfueled reactors.

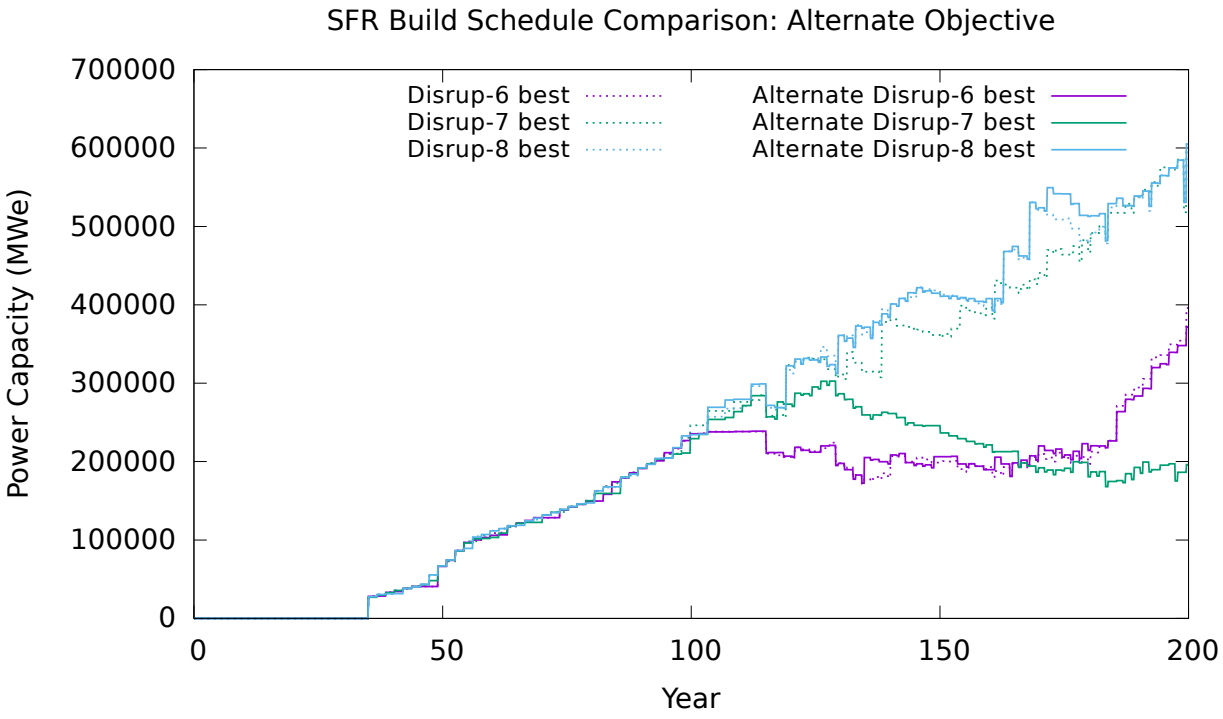

Figure 5.24: Installed SFR power capacity over time for the best deployment schedules for disruption times 6–8 for both the original and alternate objective functions. Deployments for each single disruption time for both objective functions are very similar for all disruption times except disruption 7 (year 123).

alternate object function are most clearly seen in Figure 5.24. Among all three disruption times shown, the only significant difference between best schedules from the two objective functions occurs for disruption 7 in year 123. In fact, the objective values for corresponding disruption times are nearly the same between both objectives except for the year 123 disruption. These values are shown in Table 5.7. The original objective function optimization runs used approximately 33% more total function evaluations which may account for the small discrepancies in the table's values for the year 86, 104, and 144 disruptions.

The alternate objective function results in a qualitatively different optimum for disruption 7 in year 123. In Figure 5.22 for disruption 7 the alternate objective causes optimal deployments to diverge from best deployments for other disruption times *prior* to the year 123 disruption. While small, the deviation is significant. One could reasonably imagine other objective functions and scenarios that more consistently produce deployment sched-

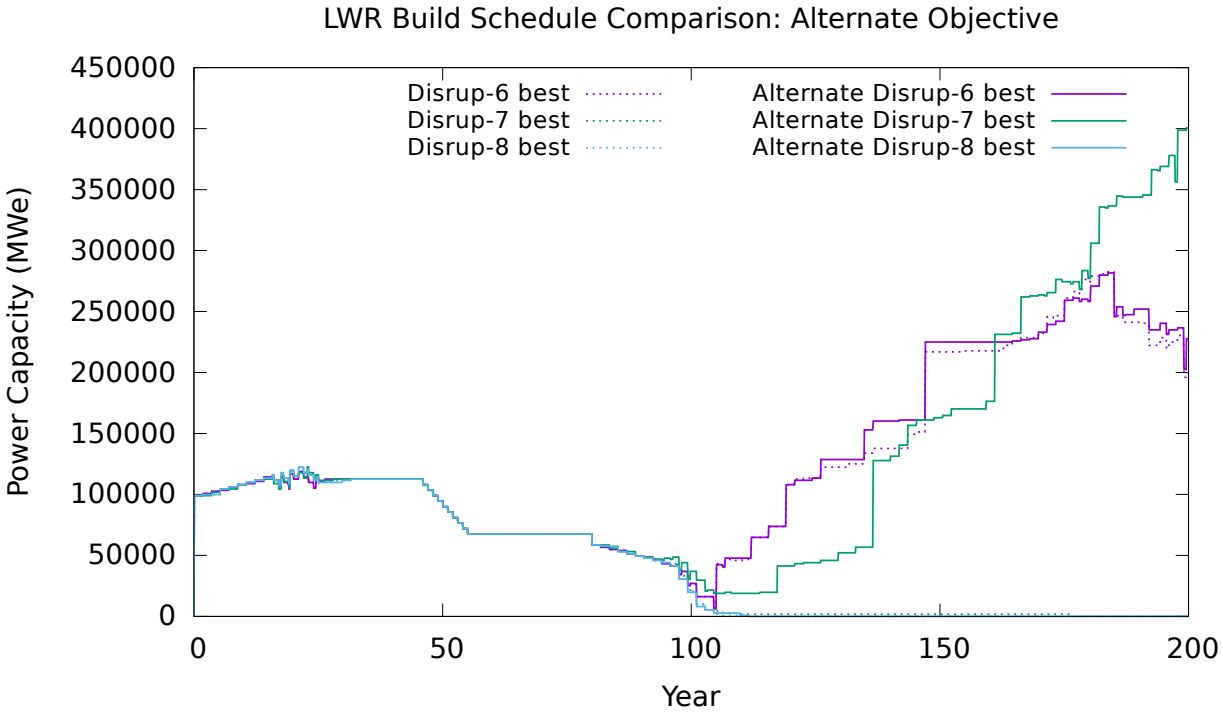

Figure 5.25: Installed LWR power capacity over time for the best deployment schedules for disruption times 6–8 for both the original and alternate objective functions. Deployments for each single disruption time for both objective functions are very similar for all disruption times except disruption 7 (year 123). Notably, the alternate objective also causes LWR deployments to begin to increase slightly *prior* to the disruption.

ules involving larger pre-disruption preparations across a variety of disruption times. The difference seen here, however, only occurs for a narrow window of disruption times around year 123, and the disruption PDF is much more heavily skewed toward earlier disruptions. For these reasons, the unfueled reactors artifact seen in the original $D^*(t_d = 123)$ deployment schedule (i.e. Figure 5.8) is not likely to have had a significant impact on the overall hedging analysis.

Although modifications to the objective function like those done in this exercise can make intermediate results more realistic (i.e. preventing unfueled reactors), such modifications do not, in this case, affect the optimum hedging strategy. As discussed earlier in Section 5.3.3, the best hedging strategy is to proceed as if no disruption will occur (i.e. disruption at year 200) and respond accordingly if a disruption occurs.

Alternate Objective Comparison

Disruption Year (t_d)	Best Objective Value	
	Orig. Func.	Alternate Func.
86	0.545	0.548
104	0.481	0.484
123	0.384	0.454
144	0.209	0.214

Table 5.7: The objective values for the best deployment schedules as determined by optimization runs for disruption sample times 5, 6, 7, and 8 under the original and the alternate objective functions. The year 123 disruption produces qualitatively different optima for the two objective functions resulting in a significantly different objective value. Disruption times have been rounded to the nearest whole year.

5.4 Summary

The hedging methodologies developed and demonstrated here show promise as tools for understanding how the effects of uncertainty can be measured and mitigated. This experiment builds on the theme of Experiment 1 enabling bigger questions to be asked in fuel cycle analysis. Rather than assuming an exact, particular future (which is very unrealistic), deployment strategies were evaluated under a probability distribution of futures involving a time-varying disruption that permanently reduces Pu availability. Evaluating the hedging effectiveness of a particular deployment schedule involved several optimization runs in order to determine reference outcomes and post-disruption deployment schedule adjustments.

Discovering good hedging strategies is much more difficult than measuring hedging effectiveness. Because computing the hedging effectiveness of a single deployment schedule requires several optimization runs itself, running an optimization to find an optimum hedging strategy becomes a nested problem where each objective evaluation requires an entire sub-optimization. Such a nested approach is not feasible in practice, requiring far too long to solve. To overcome this, a few methods for quickly approximating the best achievable objective for a particular deployment schedule and disruption time were

developed. Such approximations can reduce the complexity of the hedging optimization by eliminating the previously required, nested sub-optimizations. These methods were used to seek an ideal hedging strategy.

The hedging optimization was moderately successful, finding a deployment schedule that performed quite well across the full range of disruption times but not performing as well as the best schedule from the individual disruption time optimization runs. The simple nature of the objective function and disruption resulted in the ideal hedging strategy to be assuming no disruption will occur. Optimal pre-disruption deployments were nearly the same among all single-disruption cases meaning that each disruption time's optimal deployment strategy did not involve preparing for the disruption in advance. This resulted in a hedging strategy that was able to achieve the nearly the best possible outcome across all disruption times without compromise. More realistic or sophisticated objectives and disruptions can easily force optimal deployment schedules to require pre-disruption preparation. Hedging strategies for such scenarios would require compromise of performance for certain disruption times in order to improve the expected/mean outcome.

Two key visualizations were used to compare hedging properties of deployment schedules. The best achievable objective was plotted against disruption time allowing direct comparison of deployment schedules' performance for specific disruption times. The other key figure showed outcome distributions weighted with the disruption time PDF providing a big picture view of the risk-reward trade-off for different strategies.

Some interesting artifacts appeared in some of the investigated deployment schedules. One deployment schedule resulted in many unfueled reactors in some simulations. This artifact was caused by limitations of the objective function and did not have a particularly significant impact on the overall hedging analysis. Another artifact present in many deployment schedules was a drop in deployed power capacity from the upper down to the lower demand bound during the last third of the simulation. This can be explained as a boundary effect. The optimizer was able to capitalize on the finite simulation time horizon

achieving better objective values by trading SFR fleet youthfulness for better Pu utilization near the end of the simulation.

6 CONCLUSION

To date, nuclear fuel cycle analysis has mostly involved hand-crafting many details of fuel cycle scenarios for analysis. There is great potential for computational resources to improve both the quality of answers and the scope of questions that can be asked. Cyclus is one of the first nuclear fuel cycle simulators to strongly accommodate larger-scale analysis with its free availability, liberal open-source licensing, and first-class Linux support.

In Experiment 1, an effective technique for automatically determining desirable deployment schedules for fuel cycle scenarios is developed with the Cyclus fuel cycle simulator. This involved developing a novel way to map optimization variables into a deployment schedule. This structuring of the optimization variables allows fuel-cycle-specific details (i.e. relationships between reactor types and power capacity/demand constraints) to be represented implicitly in the definition of the variables. This not only enables optimizers without constraint support to be used, but it also prevents wasting computational resources searching through many infeasible deployment schedules. With simplified constraint requirements, optimization can be used to analyze larger problems in addition to providing better solutions generally.

Further, Experiment 1 investigated optimization algorithms in search of ones that worked well for discovering deployment schedules for fuel cycle analysis. A custom PSwarm-based optimizer was created that outperformed several other optimizers from the DAKOTA framework. The underlying discrete, discontinuous, non-convex nature of objective functions in this space makes evolutionary and other stochastic-type algorithms a good fit. Increasing parallelism provides quickly diminishing returns with respect to improving convergence. Minimizing dimensionality is important and has a large impact on the quality of answers optimization can provide for fuel cycle analysis.

The best deployment schedule discovered in Experiment 1 indicated potential for expediting a transition to a closed fast reactor fuel cycle by slowing new LWR construction

(i.e. taking advantage of the $\pm 10\%$ flexibility) preempting a change to fast-reactor-only construction. This deployment strategy had an objective value nearly 15% lower than naively deploying along the exponential demand curve. Although not necessarily realistic in many details, the results demonstrate the utility of these methods and tools for discovering interesting strategies and providing insight. Optimization analysis such as this is a useful tool for generally identifying potential pitfalls and opportunities.

The heavy Cyclus usage and scenario development in Experiment 1 established a good foundation for investigating impacts of some common modeling assumptions made in fuel cycle analysis. Cyclus' unique ability to easily swap different facility models in and out of scenarios makes investigating the effects of assumptions much more feasible. Experiment 2 investigated effects of some simple things that are often taken for granted in fuel cycle analysis: time step duration and discretization of facilities (e.g. individual vs. fleet reactors).

Modeling reactors individually can provide valuable insight not possible with the fleet-based reactor modeling. Using individual reactors, for example, it became apparent that bringing reactors back online following shortage-induced outages should not necessarily follow the natural pacing of fuel availability, otherwise refueling cycles become unsatisfactorily synchronized. Although individual reactor outage modeling can be very useful for certain types of analysis, it can reduce result quality if things such as cycle staggering are not handled appropriately. If individual reactor modeling fidelity is not required, however, then performance benefits in both simulation run time and data analysis may result in a fleet-based modeling approach being more appropriate. Due to the *inventory drawdown effect*, changes in time step duration can impact the aggregate throughput of supply-constrained recycle loops by affecting minimum bounds on idling material quantities.

Cyclus' flexibility for accommodating different modeling choices uniquely enables many interesting comparisons. The value of realism in fuel cycle modeling is greatly be enhanced by an associated understanding of how outcomes are affected by modeling fidelity. This

understanding can be enhanced through exercises like Experiment 2. Those in the field of fuel cycle analysis in general should be cognizant of how and to what degree modeling choices may be affecting results and conclusions.

Experiment 3 further builds on the theme of Experiment 1 enabling bigger questions to be asked in fuel cycle analysis. This is made possible by Cyclus' ability to be deployed on cluster type environments in conjunction with unique methods developed for evaluating deployment strategies under uncertainty. Rather than assuming an exact, particular future (which is very unrealistic), strategies can be evaluated under a diverse set of possible futures. Questions like "How would a potential permanent moratorium on deploying technology X affect what we build now?" and "How can we hedge against likely socio-political changes in the balance between economics and environmental concerns?" can begin to be approachable.

Evaluating hedging effectiveness of particular strategies under uncertainty requires more computational resources than a typical optimization run. In order to evaluate a single potential hedging strategy, optimal responses must be identified across the spectrum of uncertainty resolutions. This requires several optimization runs. Outcome distributions can then be generated that provide a good high-level mechanism for evaluating the risk-reward trade-off of different strategies. Going beyond evaluating hedging strategies, automatically generating good hedging strategies is much more difficult. The simple approximation techniques used for hedging optimization here, while moderately effective, would need to be extended and improved in order to become more generally useful. Despite challenges, however, the hedging evaluation and optimization methodologies are promising tools for better understanding how the effects of uncertainty can be measured and mitigated.

6.1 Future Work

Managing problem dimensionality will be an important component of future fuel cycle optimization work. In the experiments done here, variables were designed to represent capacity levels and reactor fractions. For annual deployments over 100 years with several facilities, the number of variables increases quickly into the 100s such that finding good deployment schedules for particular objectives (e.g. fastest transition, cheapest transition, etc.) can quickly become intractable. There is potential for the development of techniques to reduce this dimensionality. One such mechanism would be to further modify variables to represent points on power and facility-ratio curves rather than being explicit values for every time step. Interpolations schemes could then be used to determine deployments between the times represented by the variables. Another complementary way to reduce dimensionality would be to replace part of the decision space with heuristics (likely intra-simulation) that use domain knowledge and current simulation state to execute approximations to optimal behavior on shorter time scales internally between extra-simulation optimizer-specified variables. This might take the form of look-ahead type capability where internally, the simulation peeks ahead a few time steps at the future, rewinds back to the original time step, and then makes some adjustments according to observations about the future.

Hedging strategy optimization is impeded by the computational complexity of dealing with post-disruption behavioral change - resulting in a nested optimization problem. The simple approximation method used here for avoiding this problem leaves much room for improvement. More sophisticated techniques could be developed that attempt to convert abstract strategies for dealing with the disruption into concrete deployment responses. Understanding how such approximation techniques perform under a variety of different objectives and disruptions would also be valuable.

The disruption and hedging methodology could be extended to handle multiple uncertain events. For example, a large-capacity separations facility may potentially go offline independent of a potential time varying objective function perturbation due to shifts in

social and political winds. Handling multiple uncertainties simultaneously would involve multi-dimensional PDFs. Measuring hedging effectiveness would then require developing a multi-dimensional approximation of $S^*(x_1, x_2, ...)$ where each x_i represents a possible resolution of an uncertainty dimension. Generating an approximation to S^* could involve using response surface techniques such as Kriging.

Ultimately, the primary utility of this research is in providing a foundation for better understanding nuclear energy as it functions in the real world. The scenario details, objective functions, and disruption used for these experiments lack a level of detail or realism often associated with fuel cycle analysis work. Less naive physics with respect to handling material burnup calculations could increase the integrity of results (i.e. more sophisticated reactor models). Adding an economic component to the objective function such as facility construction and operational costs would provide interesting feedback in recycle vs. waste disposal trade-offs. Experiment 3 methods could be very useful to policy-focused work with a more reality-based disruption such as an objective function perturbation representing a shift in preferences regarding economic-environmental trade-offs. The use of these optimization and hedging methods on a more realistic problem would provide valuable feedback for the improvement the methods in addition to contributing to a better understanding of their utility for fuel cycle analysis generally.

REFERENCES

[1] Adams, B.M., L.E. Bauman, W.J. Bohnhoff, K.R. Dalbey, M.S. Ebeida, J.P. Eddy, M.S. Eldred, P.D. Hough, K.T. Hu, J.D. Jakeman, L.P. Swiler, and D.M. Vigil. 2013. DAKOTA, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 5.4 user's manual.

[2] Audet, Charles, and J. E. Dennis. 2006. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization* 17(1):188–217.

[3] Borshchev, Andrei, and Alexei Filippov. 2004. From system dynamics and discrete event to practical agent based modeling: Reasons, techniques, tools. Oxford, England.

[4] Brinton, S., S. Passerini, and M. Kazimi. 2013. Nuclear fuel cycle analysis and optimization with the code for advanced fuel cycles assessment (CAFCA). *Transactions of the American Nuclear Society* 108:131–3. INSPEC:14053434.

[5] Carlsen, Robert W., Matthew J. Gidden, and Paul P.H. Wilson. 2014. Deployment Optimization with the CYCLUS Fuel Cycle Simulator. In *Transactions of the American Nuclear Society*, vol. 111, 241–244. Anaheim, CA. DOI link for code, methods, etc: http://dx.doi.org/10.6084/m9.figshare.1086284.

[6] Chen, Stephen, James Montgomery, and Antonio BolufÃ©-RÃ¶hler. 2015. Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution. *Applied Intelligence* 42(3):514–526.

[7] Clerc, M., and J. Kennedy. 2002. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation* 6(1):58–73.

[8] Coello, C. a. C. 2002. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering* 191(11-12):1245–1287. WOS:000173304000009.

[9] Eastham, Sebastian D., David J. Coates, and Geoffrey T. Parks. 2012. A novel method for rapid comparative quantitative analysis of nuclear fuel cycles. *Annals of Nuclear Energy* 42:80–88. WOS:000302670300011.

[10] Eberhart, Russ C., and James Kennedy. 1995. A new optimizer using particle swarm theory. *Proceedings of the sixth international symposium on micro machine and human science* 1:39âŁ“43.

[11] Eddy, John, and Kemper Lewis. 2001. Effective generation of pareto sets using genetic programming. In *2001 ASME design engineering technical conference and computers and information in engineering conference, september 9, 2001 - september 12, 2001*, vol. 2 of *Proceedings of the ASME Design Engineering Technical Conference*, 783–791. American Society of Mechanical Engineers.

[12] Epstein, Joshua M. 2006. *Generative social science: Studies in agent-based computational modeling*. Princeton University Press.

[13] Flicker, Margaret. 2014. Evaluation criteria for analyses of nuclear fuel cycles. vol. 111, 233–236. Anaheim, CA.

[14] Forrester, Alexander I. J., and Andy J. Keane. 2009. Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences* 45(1âŁ“3):50–79.

[15] Forrester, Jay Wright. 1961. *Industrial dynamics*, vol. 2. MIT press Cambridge, MA.

[16] Gidden, Matthew. 2014. An Agent-Based Modeling Framework and Application for the Generic Nuclear Fuel Cycle.

[17] Gidden, Matthew, R. Carlsen, A. Opotowsky, O. Rakhimov, A. Scopatz, and P. Wilson. 2014. Agent-based dynamic resource exchange in cyclus. In *Proceedings of PHYSOR*. Kyoto, Japan.

[18] Hays, Ross, and Paul Turinsky. 2014. Stochastic optimization for nuclear facility deployment scenarios using vision. *Nuclear Technology* 186(1):76–89. WOS:000334977700006.

[19] Jacobson, J., A. Yacout, G. Matthern, S. Piet, D. Shropshire, R. Jeffers, and T Schweitzer. 2010. VERIFIABLE FUEL CYCLE SIMULATION MODEL (VISION): A TOOL FOR ANALYZING NUCLEAR FUEL CYCLE FUTURES. *Nuclear Technology* 172:157–178.

[20] Kennedy, James, James F. Kennedy, and Russell C. Eberhart. 2001. *Swarm intelligence*. Morgan Kaufmann.

[21] Kunsch, P. L., and J. Teghem Jr. 1987. Nuclear fuel cycle optimization using multi-objective stochastic linear programming. *European Journal of Operational Research* 31(2): 240–249.

[22] Laidler, J. J., J. E. Battles, W. E. Miller, J. P. Ackerman, and E. L. Carls. 1997. Development of pyroprocessing technology. *Progress in Nuclear Energy* 31(1âŁ“2):131–140.

[23] Lewis, R. M., and V. Torczon. 2000. Pattern search methods for linearly constrained minimization. *Siam Journal on Optimization* 10(3):917–941. WOS:000087452000015.

[24] Noraini, Mohd Razali, and John Geraghty. 2011. Genetic algorithm performance with different selection strategies in solving TSP. Imperial College, London.

[25] North, Michael J. 2014. A theoretical formalism for analyzing agent-based models. *Complex Adaptive Systems Modeling* 2(1):3.

[26] OECD. 2010. Projected cost of generating electricity, 2010 edition.

[27] Paich, Mark. 2004. Modeling general motors and the north american automobile market.

[28] Park, Byung Heung, Fanxing Gao, Eun-ha Kwon, and Won Il Ko. 2011. Comparative study of different nuclear fuel cycle options: Quantitative analysis on material flow. *Energy Policy* 39(11):6916–6924. WOS:000298120200022.

[29] Passerini, Stefano, Mujid S. Kazimi, and Eugene Shwageraus. 2012. Sensitivity analysis and optimization of the nuclear fuel cycle. In *International congress on advances in nuclear power plants 2012, ICAPP 2012, june 24, 2012 - june 28, 2012*, vol. 4 of *International Congress on Advances in Nuclear Power Plants 2012, ICAPP 2012*, 2512–2519. American Nuclear Society.

[30] Piet, Steven J., Brent W. Dixon, Jacob J. Jacobson, Gretchen E. Matthern, and David E. Shropshire. 2011. Dynamic simulations of advanced fuel cycles. *Nuclear Technology* 173(3):227–238.

[31] Pulido, Gregorio Toscano, and Carlos A. Coello Coello. 2004. A constraint-handling mechanism for particle swarm optimization. In *Evolutionary computation, 2004. CEC2004. congress on*, vol. 2, 1396âŁ“1403. IEEE.

[32] R0oland. TwoPointCrossover.

[33] on Radiological Protection, International Commission. 1993. *Age-dependent doses to members of the public from intake of radionuclides: Compilation of ingestion and inhalation dose coefficients*. Age-dependent doses to members of the public from intake of radionuclides: Compilation of ingestion and inhalation dose coefficients no. 72, Pergamon.

[34] Reyes-Sierra, Margarita, and CA Coello Coello. 2006. Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *International journal of computational intelligence research* 2(3):287âŁ“308.

[35] Rios, Luis Miguel, and Nikolaos V. Sahinidis. 2013. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization* 56(3):1247–1293.

[36] Schweitzer, Tyler M., Paul J. Turinsky, and Jacob J. Jacobson. 2008. Analysis of inventories and lead times for building separation facilities with the verifiable fuel cycle simulation model (VISION). vol. 99 of *Transactions of the American Nuclear Society*, 191–192. American Nuclear Society.

[37] Shropshire, D. E., K. A. Williams, W. B. Boore, J. D. Smith, B. W. Dixon, M. Dunzik-Gougar, R. D. Adams, D. Gombert, and E. Schneider. 2007. Advanced fuel cycle cost basis. *Idaho National Laboratory, INL/EXT-07-12107*.

[38] Vaz, A. I. F., and L. N. Vicente. 2009. PSwarm: a hybrid solver for linearly constrained global derivative-free optimization. *Optimization Methods & Software* 24(4-5):669–685. WOS:000268801200010.

[39] Vaz, A. Ismael F., and LuÃ-s N. Vicente. 2007. A particle swarm pattern search method for bound constrained global optimization. *Journal of Global Optimization* 39(2):197–219. WOS:000249261400003.

[40] Whitley, D. 1994. A genetic algorithm tutorial. *Statistics and Computing* 4(2):65–85. WOS:A1994NQ60200004.

[41] Wilson, Paul P.H. Comparing nuclear fuel cycle options. Tech. Rep.