Genetic Algorithm Optimization of Experiment Design for Targeted Uncertainty Reduction

A Thesis Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Alexander Amedeo DePillis

May 2024

ACKNOWLEDGEMENTS

I would like to acknowledge Dr. Sobes, Dr. Bogetic, Dr. Hines and my committee member Dr. Hayward, without whom this thesis would not have been possible.

I must also give thanks to my parents, brother and friends. Joseph Crapse especially has been a persistent inspiration. Additional thanks to Mori C. and P. Pippa for their music which kept me awake on the long drive across the country.

This material is based upon work supported in part by the Department of Energy National Nuclear Security Administration through the Nuclear Science and Security Consortium under Award Number DE-NA0003996. This report was prepared by the research group of Dr. Sobes under award 31310021M0041 from Assistance Agreements, Nuclear Regulatory Commission. The statements, findings, conclusions, and recommendations are those of the author(s) and do not necessarily reflect the view of the Assistance Agreements or the US Nuclear Regulatory Commission.

ABSTRACT

Nuclear cross sections are a set of parameters that capture probability information about various nuclear reactions. Nuclear cross section data must be experimentally measured, and this results in simulations with nuclear data-induced uncertainties on simulation outputs. This nuclear data-induced uncertainty on most parameters of interest can be reduced by adjusting the nuclear data based on the results from an experiment. Integral nuclear experiments are experiments where the results are related to many different cross sections. Nuclear data may be adjusted to have less uncertainty by adjusting them to match the results obtained from integral experiments. Different integral experiments will adjust cross section data differently and reduce their associated uncertainties differently.

Various nuclear experts use simulation systems to calculate parameters of interest for various nuclear systems. These simulated application systems can make use of adjusted nuclear to reduce the uncertainties on their parameters of interest. Thus, different integral experiments have different impacts on reducing the uncertainties on the parameters of interest for different application systems.

This thesis verifies a method for predicting the uncertainty reduction on a parameter from a given integral experiment. This will allow the designers of integral experiments to predict how valuable their experiments will be for various applications and parameters. The verification is performed by first simulating an experiment system, an application system, the different possible cross section values that really exist, an imperfect measurement of the experiment system, the process of adjusting the cross section values, and the application of the adjusted values to the simulation of the application system. The results of that procedure are then compared to the uncertainty reduction predicted by a recently developed formula called the Uncertainty Reduction Ratio (URR) formula. The advantage of the URR formula is that it can be applied to a wider range of systems and parameters than those that had to be assumed for the verification procedure.

After verification of the URR formula, this thesis goes on to use the formula to demonstrate the optimization of the design of an integral experiment system for the reduction of uncertainty on parameters of interest for application systems. The experimental system whose optimization is being demonstrated is the Flexible Neutron Source (FNS) at the University of Tennessee Knoxville. The nature of the FNS optimization meant that many existing optimization algorithms for integral experiment design would be either inappropriate or inefficient.

The FNS a multi-purpose sub-critical experiment system planned to be constructed in the basement of the newly constructed Zeanah Engineering Complex. The FNS is a highly modular system that can be configured in an extremely large number of ways by rearranging a number of plates of various

materials. The FNS is to be composed of 74 aluminum drawers (cassettes) arranged into three 5 by 5 layers (also referred to as zones), each of which is considered a distinct zone. Each cassette has room for twenty 0.5-inch material plates. These cassettes can be filled with plates made of a variety of materials depending on the application the FNS is being leveraged for and can be rearranged to perform different experiments. The number of combinations of plates in the FNS is large ($\sim 4^{470}$). A search algorithm needs to be used to determine a configuration that will be near optimal for adjusting cross section data for any given application.

A new edition of a general-purpose global optimization software package, Gnowee, is developed called Gnowee_multi. The new edition is improved in many ways including but not limited to: radical speed increases, parallelization, multi-objective capability, crash avoidance, crash recovery, and conversion to Python 3. The new edition is applied to the optimization of the FNS and sufficiently demonstrated its utility.

A demonstration optimization of the FNS was performed for an application reducing the uncertainty on the k-eigenvalue of a Sodium Fast Reactor (SFR). The optimization considered a limited design space, a limited set of unadjusted ENDF/B-VII.1 nuclear data, a very limited model of the measurement of the FNS, and calculated the sensitivities of the FNS (required for the URR formula) using the now out-of-date `PERT` MCNP 6.3 card (with associated limitations). Under those conditions, the final optimized designs suggested that the FNS could be used to reduce the uncertainty on the k-eigenvalue (A.K.A "the level of criticality") of a generic SFR by between 15% and 20%.

The future work section describes extensions and variations of the URR formula as well as the replacement of the `PERT` card.

TABLE OF CONTENTS

Chapter 1: Introduction 1

Chapter 2: Literature Review 7

2.1: Integral Experiment Optimization 7

2.2: Algorithmic Background 9

2.3: Previous FNS Work 11

Chapter 3: Changes to Gnowee 18

3.1 The Evolution and Mutation of the Gnowee Program 18

3.2 Conversion to Python 3 20

3.3 Implementing NSGA-II into Gnowee 20

3.4: Bad Vector Determination Step 24

3.5: Past Results System 26

3.6 Batch Submission System 27

3.7 Backwards Compatibility 30

Chapter 4: The Uncertainty Reduction Ratio 33

4.1: Background on the Uncertainty Reduction Ratio Formula 33

4.2: Verification of the Uncertainty Reduction Ratio Formula 34

Chapter 5: Optimizations 40

5.1: Multi-Objective MCFR Optimization 44

5.2: Single Objective SFR Optimization 51

5.3: A Point of Comparison 58

Chapter 6: Conclusions 59

6.1: Contributions 59

6.2: Future Work 59

References 62

Vita 65

LIST OF FIGURES

Figure 1.1: FNS Under Construction 3
Figure 1.2: Diagram of the FNS and Symmetry Patterns 4
Figure 1.3: CAD Model of the FNS. 4
Figure 1.4: The structure of a basic genetic algorithm. 5
Figure 2.1: Timeline of FNS Papers 12
Figure 2.2: Early Separated Design of the FNS 14
Figure 2.3: Early Rod-Based Design of the FNS 14
Figure 4.1: Calculation of Verification Data 36
Figure 4.2: Verification Results. 37
Figure 5.1: Diagram of FNS Design 41
Figure 5.2: MCNP model of MCFR Application 46
Figure 5.3: URR over the generations. 47
Figure 5.4: The Last Pareto Front 48
Figure 5.5: Plate Materials and Measurement Time. 49
Figure 5.6: Plate Materials and URR. 49
Figure 5.7: The URR of Selected Members of the Final Pareto Front if they had other time values. 50
Figure 5.8: URR over the generations. 55
Figure 5.9: The Optimal Design. 56
Figure 5.10: The Drivers of the Optimization. 56
Figure 5.11: Shared Uncertainty Term and the Number of Fuel Plates 57

LIST OF TABLES

Table 3.1: New Keyword Arguments Implemented in Gnowee_multi 32
Table 4.1: Bootstrapping Results 39
Table 5.1: Keyword Hyperparameters 47
Table 5.2: Isotopes and Reactions of Interest 53
Table 5.3: Keyword Hyperparameters Used for SFR Gnowee Optimization 55

Chapter 1: Introduction

This chapter[1] serves as an introduction to the moderately informed reader on the nature of the present thesis.

Nuclear cross sections are a set of parameters that act as proxies for the probabilities of nuclear reactions and are an essential basis for modeling any nuclear system. Fundamental nuclear properties for medium mass nuclei cannot be derived from first principles without accuracy reducing assumptions [5], [6] and thus nuclear cross sections must be experimentally measured and collected into databases. Most modern nuclear cross section libraries include uncertainties for their cross section data [7]. These nuclear data uncertainties result in uncertainties on the quantities of interest calculated by models and simulations of nuclear systems.

Historically, neutron cross section data were originally determined from differential cross section measurements [8]. Unfortunately, the errors on this cross section data for reactor related isotopes are often too large to predict the properties of reactors to within an acceptable accuracy and designers took to adjusting the cross section data based on the results of integral experiments that can have much better accuracy and less systematic error [8]. Integral experiments are nuclear experiments whose measured response depends on a range of different pieces of nuclear data [9]. Many integral experiments have been performed over the years and can be used in complementary fashions with each other (and the differential measurements they are used to fine tune the results of), but the set used to adjust data for a given application must be selected carefully to avoid introducing systemic error. When an experiment is designed to reduce uncertainty on a parameter of a specific application it may result in greater uncertainty reduction than a "generic" integral experiment.

Newer reactor designs and detector systems have data induced uncertainties on their models that need to be reduced for their designs to proceed [10]. For example, some reactor models have more than 100% uncertainty in calculated feedback coefficients, resulting in an unknown direction of feedback effects [11].

One proposed integral experiment system that could help address this need for new integral experiments is the Flexible[2] Neutron Source (FNS) experiment system at the University of Tennessee in the newly built Zeanah Engineering Complex [11]. The FNS is a highly modular system that can be

[1] Portions of this chapter are from previous work [1], [2],[3],[4], by Alexander DePillis, Vladimir Sobes, J. Wesley Hines, and Sandra Bogetic

[2] In earlier works the system was named the Fast Neutron Source (FNS), but it has since been renamed

configured in an extremely large number of ways by rearranging a number of plates of various materials [1]. Figure 1.1 shows the FNS under construction. The FNS is to be composed of 74 aluminum drawers (cassettes) arranged into three 5 by 5 layers (also referred to as zones), each of which is considered a distinct zone (the middle position in Zone 3, Figure 1.2, is empty to allow for a deuterium-deuterium (DD) neutron generator). Half a cassette is present in the middle of Zone 1 to allow space for various experiments. For the purposes of the present thesis, the experiment volume is assumed to be used for an integral flux (ϕ) measurement.

Each cassette has room for twenty 0.5-inch material plates (pictured in Figure 1.3). These cassettes can be filled with plates made of a variety of materials depending on the application the FNS is being leveraged for and can be rearranged to perform different experiments.

This means that it can be customized to help reduce uncertainty on a specific application's parameter[3]. The number of combinations of plates in the FNS is large ($\sim 4^{470}$). A search algorithm needs to be used to determine a configuration that will be near optimal for adjusting cross section data for any given application. Optimal in this context means that configuration of the FNS that will provide the greatest reduction in the cross section-induced uncertainty of a given application reactor resulting from a measurement performed on the FNS. Thus, there is an optimization problem that can be defined.

Genetic algorithms are one method of solving optimization problems. A genetic algorithm solves an optimization problem by mimicking the process of evolution and survival of the fittest. Every combination of the variables being optimized over can be thought of as a genome for an individual. In evolutionary theory, fitness is the tendency of an individual to pass on its genes. When using a genetic algorithm to solve an optimization problem, the value being optimized for (which the user specifies) is the fitness of the corresponding solution/individual. Evolution will then work to maximize the fitness and thus solve the optimization problem. The user may also specify the size of the population that is allowed in each generation and the particular methods by which new individuals are created. The structure of a basic genetic algorithm is shown in Figure 1.4.

Genetic algorithms have several key advantages for dealing with a problem like the optimization of the FNS. Firstly, they do not require the derivative of the fitness with respect to the variables being optimized over. Swappable plates do not constitute a variable that the fitness could have its derivative taken with respect to. Second, genetic algorithms work with a wide range of fitness functions. Third,

[3] In the present thesis, the parameter of interest for application systems is assumed to be k_{eff}, but the methods and logic of the present work could be applied to many other parameters as well

Figure 1.1: FNS Under Construction

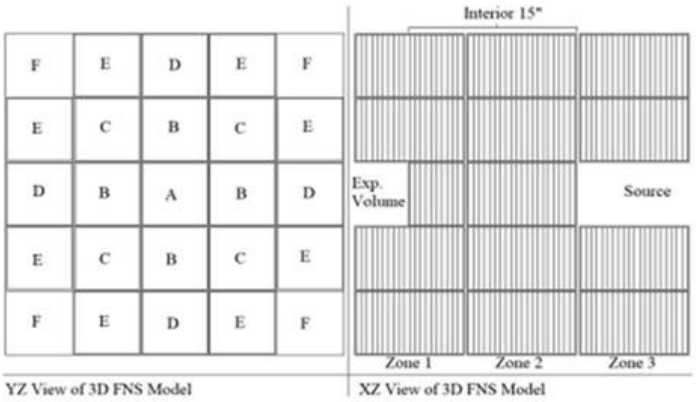


Figure 1.2: Diagram of the FNS and Symmetry Patterns (reprinted from [12].) The symmetry pattern holds within each zone and helps reduce the design space. Letters specify the locations that hold identical sets of plates for a given zone.

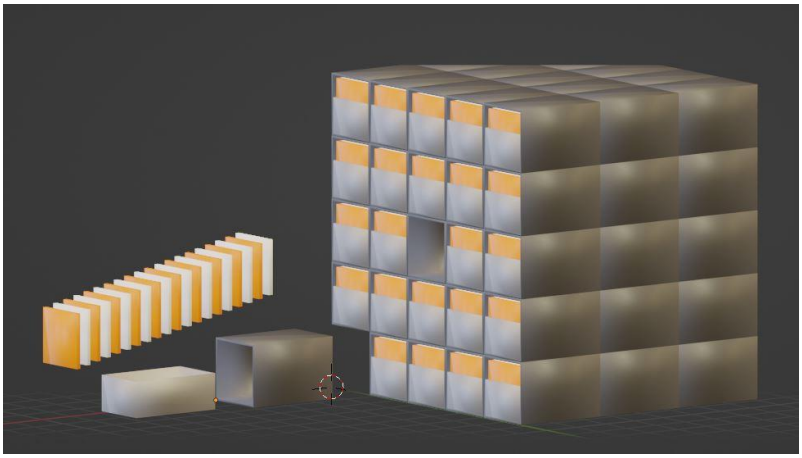


Figure 1.3: CAD Model of the FNS. Colors used for visual effect. Each zone contains a 5 by 5 set of drawers, each drawer contains the space for 20 plates (with the exception of the middle drawer in the first and last zones). [13]

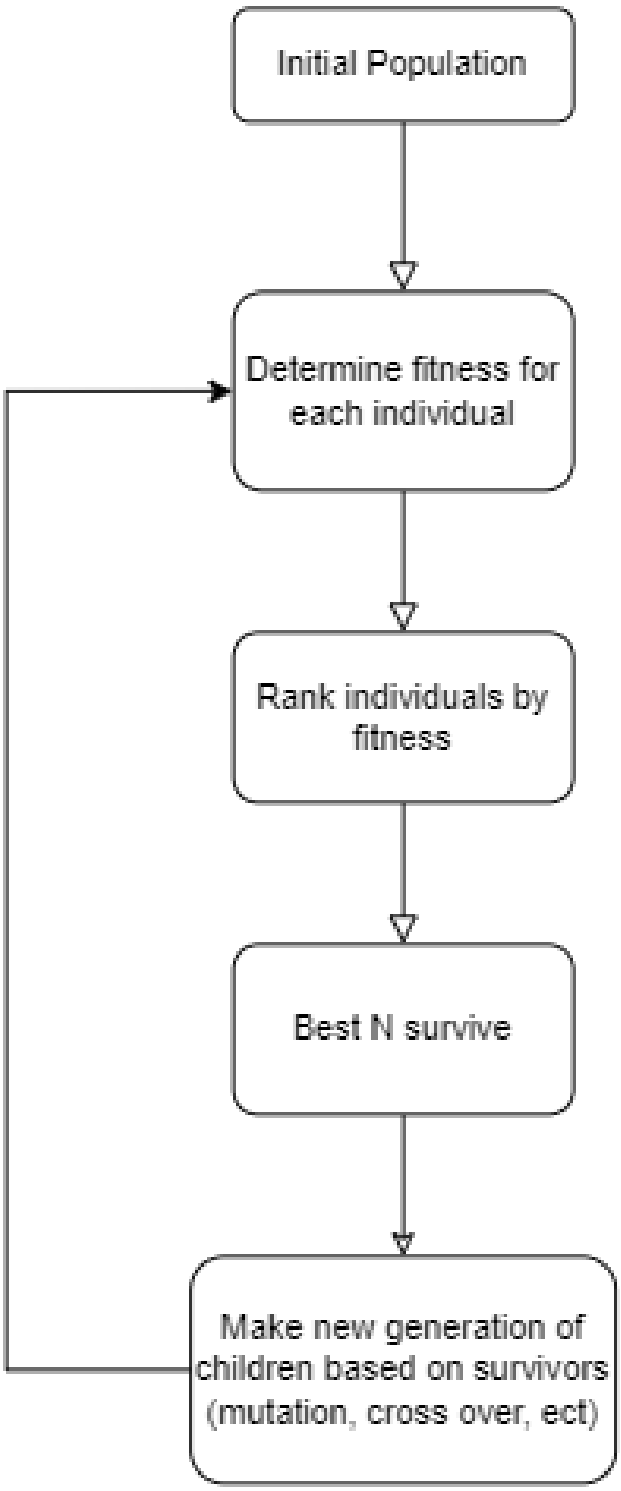


Figure 1.4: The structure of a basic genetic algorithm. The size of a generation (N) and the exact mechanics of generating new solution (AKA "children") vary from genetic algorithm to genetic algorithm.

genetic algorithms generalize easily to both discrete and continuous variables. Fourth, the individuals within a single generation can be evaluated in parallel.

The genetic algorithm used in this thesis is called Gnowee [14] and it is discussed in detail in chapters two and three. The basic premise of Gnowee is that it uses a variety of methods for creating children called "search heuristics" or "metaheuristics". The Gnowee user defines a python function that will compute fitness if given a particular solution[4] and tells the program the basic parameters of the optimization problem; Gnowee then attempts to find the solution with the minimum fitness.

The fitness function used in conjunction with Gnowee, for the present thesis, is a formula that calculates the cross section induced uncertainty on an application's parameter of interest after the cross sections have been adjusted based on an FNS experiment relative to its prior cross section induced uncertainty on the parameter. The formula to calculate this is discussed in chapter four and the verification of the formula is a major contribution of the present thesis along with the improvement of the Gnowee algorithm.

[4] A particular solution may sometimes be referred to as being: an individual, a feature vector (since its particular values are stored as a vector), a parent (if its fitness has already been evaluated and it has been put into the elite population), or a child (if it is a recently generated solution that has yet to be evaluated)

Chapter 2: Literature Review

In this chapter[5], papers that use genetic algorithms to optimize nuclear systems, papers that optimize integral experiment designs, papers that advance the FNS, and papers that provide background on the various algorithms used for these purposes are reviewed.

2.1: Integral Experiment Optimization

This section is meant to review some of the modern work that optimizes integral nuclear experiments.

"Sensitivity Studies, Gap Analysis, and Benchmark Experiment Optimization for Reactor Physics and Criticality Safety Applications" [15]

ARCHIMEDES (Application Relevant Critical/Subcritical HEU/Pu-based Integral Measurements for Enhancing Data and Evaluating Sensitivities) is an approach to designing new criticality experiments that have similar k_{eff} cross-section sensitivities to those of an application system. Indeed, most attempts at optimizing integral experiments for given applications use a similarity metric like this. This means that ARCHIMEDES is based on a sensitivity matching approach, rather than seeking to reduce uncertainty on the application system's parameter of interest as is done in this thesis. The paper's sensitivity metric of choice is c_k which is equivalent to $\frac{(S^T MG)^2}{(S^T MS)(G^T MG)}$ in this work. The term represents the relative amount of uncertainty due to the cross section covariance matrix that a given experiment and application have in common. The ARCHIMEDES approach outlined in this paper can be broken down into three steps:

1. Using simulations to calculate cross section sensitivities for the application and ICSBEP/IRPHhP benchmark models.
2. Performing a gap analysis by calculating c_k for the criticality benchmarks and creating a heat map.
3. Performing an optimization of an experiment with the goal being to design a new critical experiment that has a higher c_k value with the application than any previous critical benchmark experiment.

The paper takes the view that in order to be worth performing an experiment must have a better similarity coefficient than the previous best critical benchmark and this is the implicit reason for the gap analysis. In theory you could use all the critical benchmarks in a series of Bayesian updates to create a better-

[5] Portions of this chapter are from previous work [1], [2],[3],[4], by Alexander DePillis, Vladimir Sobes, J. Wesley Hines, and Sandra Bogetic

informed covariance matrix and then perform the optimization of the experiment with a covariance matrix that implicitly already takes into account the pre-existing benchmarks, removing the need for a gap analysis. The paper leaves the experiment optimization process for future work to describe in detail but notes that the team performed a Bayesian optimization on c_k using a Gaussian Process. The Gaussian process is a type of Bayesian optimization and is typically applied to optimization problems with continuous variables, making it generally inappropriate for the FNS problem.

"EUCLID: A New Approach to Constrain Nuclear Data via Optimized Validation Experiments using Machine Learning"[16]

Experiments Underpinned by Computational Learning for Improvements in Nuclear Data (EUCLID) at Los Alamos National Laboratory was a Laboratory Directed Research and Development project whose goal was to reduce compensating errors in nuclear data. Unlike ARCHIMEDES, EUCLID depends on expert judgement to identify places in nuclear data libraries that show potential for improvement. EUCLID can be broken down into the following parts:

1. Using MCNP 6.2 with ENDF/B-VIII.0 to simulate existing validation experiments.
2. Using machine learning to augment expert judgement in finding areas within the nuclear data library that may have compensating errors.
3. Designing new experiments with machine learning to target the identified areas of the nuclear data library.
4. Performing the experiments and then adjusting the libraries to reduce compensating errors "while accounting for physics constraints".

Part of the work done by the EUCLID team included calculating and simulating the sensitivities of eight different types of responses for more than one thousand previous benchmark experiments since fully constraining nuclear data from k_{eff} measurements is impossible. When it comes to experiment optimization, this team chose reducing overall uncertainty as the way to create an experiment that will be best for reducing compensating errors. This was operationalized in the form of optimizing on the log-determinant of the posterior covariance matrix, called D-optimality. The generalized linear least squares (GLLS) was used to predict the posterior covariance matrix. The stated advantage of D-optimality over c_k is that it will better allow for the combination of multiple different experimentally measured responses. This is a very interesting feature of the approach taken in this paper and it suggests that one way to improve the URR formula would be to extend it to incorporate multiple measurements of the experiment systems. The optimization method chosen for this paper was another Gaussian Process.

"Designing Critical Experiments using Gaussian Process Optimization"[17]

At first glance this appears like just another integral experiment optimization using c_k but it is actually quite interesting since it implements a discrete version of an optimization algorithm (Gaussian Process (GP) Optimization) normally only applied to continuous variables. The optimization utilizes a gaussian process model that acts as a surrogate for c_k. It chooses to run full MCNP evaluations of the designs it expects to yield the greatest reduction in the uncertainty of the GP model. Unfortunately, GP models suffer from a "curse of dimensionality", as stated in a review of the matter [18]:

Specifically, the number of design points required to keep the same quality of approximation grows exponentially with the number of variables and the volume concentrates on the boundary of the search space. - "A Survey on High-dimensional Gaussian Process Modeling with Application to Bayesian Optimization,"

This makes it inappropriate for the FNS design problem that is central to this thesis.

2.2: Algorithmic Background

Certain papers and algorithms are foundational to the work presented in this thesis and they are reviewed here.

"Gnowee: A Hybrid Metaheuristic Optimization Algorithm for Constrained, Black Box, Combinatorial Mixed-Integer Engineering Design Problems" [14]

This article is the original paper behind the Gnowee algorithm. From the beginning, Gnowee was designed with nuclear applications in mind, and this affected the design choices involved in its creation. The article notes that expensive objective function evaluations like Monte Carlo simulation make finding globally optimal designs via parametric studies or intuition nearly impossible. Gnowee is designed to work with constraints (a useful feature for engineering applications), black box scenarios (i.e. meant to work regardless of the objective function), as well as problems with combinatorial or mixed-integer characteristics. This makes Gnowee stand out among optimization algorithms as being applicable to an incredibly vast array of possible optimization problems. The paper walks through as list of heuristic search strategies and how each applies to the major operators within the Gnowee algorithm.

Finally the paper attempts to demonstrate the superior performance characteristics of the Gnowee algorithm but notes that:

Gnowee was designed for rapid convergence of nearly globally optimum solutions for complex engineering problems with mixed-integer and combinatorial design vectors and high-cost, noisy,

discontinuous, black box objective function evaluations. However, no coherent set of benchmarks exist for this problem space, nor are there completely comparable codes against which to test.

Lacking a good standard set of benchmark problems for optimization algorithms in this area, the authors come with their own to use for comparing Gnowee against other algorithms. Gnowee was compared against one set of algorithms for the continuous benchmark problems, another set for mixed-integer problems, and a third set for combinatorial problems. Gnowee generally compared favorably in each match-up based on a figure of merit that incorporated the average fitness obtained, the average number of function evaluations it took each algorithm, and the standard deviation of the fitness solution distribution.

The work of the present thesis builds upon the Gnowee algorithm and uses it for exactly the type of problem it was designed to solve. The authors of the original paper even say that improved parallelization was an area for future work and it is a suggestion that this present thesis follows through upon.

"Improvements, Validation, and Applications of a Metaheuristic Optimization Method for Neutron Spectra Tailoring at the National Ignition Facility" [19]

Like the present thesis, Dr. Bogetic's thesis is centered around the Gnowee optimization algorithm. The thesis coupled Gnowee with MCNP on the Berkeley cluster through a program called COEUS. One of the improvements COEUS used was to implement parallel processing on the Berkely cluster for the Gnowee algorithm. This has an equivalent in one of the improvements this thesis documents in its batch submission system. Unlike the work presented in this thesis, COEUS implements additional strategies to speed up computation like the utilization of ADVANTG. This was not done for this present thesis because this work wanted to avoid being excessively problem specific and because ADVANTG is built for earlier editions of MCNP [20].

The National Ignition Facility at LLNL has a strong neutron source, which needs to be tuned for various applications such as cross section experiments, Boron-Neutron Capture Therapy (BNCT), and more. In order to make the source more accessible for more users and use cases, there was a need for the ability to automatically design and customize the nature of the neutron source through Energy Tuning Assemblies (ETAs). The thesis utilizes Gnowee and COEUS (a software package that helps couple Gnowee to MCNP) to optimize ETAs. The goal of the thesis is to further the development, validation, and application of Gnowee/COEUS to ETAs to create neutron spectra for various applications. The work at NIF consisted of: the modification and expansion of Gnowee/COEUS, experimental verification and validation of the modelling and simulation work done in the thesis, and the design of ETAs for specific applications. COEUS is integrated with Gnowee in this work and not confined to the user-supplied

objective function. COEUS 2.0 was developed by Dr. Bogetic and is different from the work performed in this thesis because among other things it:

1. Is MCNP specific
2. Works only for anticipated types of MCNP geometries
3. Will evaluate feature vectors that have already been run

"A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II" [21]

This paper introduced the nondominated sorting genetic algorithm II (NSGA-II) to the world. - NSGA-II is a fast and elitist multi-objective genetic algorithm designed to optimize high-cost objective functions effectively. It addresses the computational complexity, non-elitism, and sharing parameter specification issues found in earlier versions of multi-objective evolutionary algorithms (MOEAs). NSGA-II introduces a fast nondominated sorting approach, an elitist strategy, and a crowding distance mechanism to ensure a diverse set of solutions without requiring predefined parameters for diversity maintenance. Through simulations, NSGA-II demonstrated superior performance in spreading solutions and converging near the true Pareto-optimal front compared to other elitist MOEAs. Additionally, NSGA-II's capability in handling constraints within multi-objective optimization problems is explored, showing significant improvements over other algorithms in terms of performance and solution diversity. It is important to note that NSGA-II is mostly a method of constructing a Pareto front and breaking ties about what members should be included in a surviving population and not really about the specifics of how new individuals for evaluation are generated. This means that it can easily be imbedded or implemented inside other algorithms.

2.3: Previous FNS Work

This section presents an exhaustive list of publications documenting work done on the design of the FNS. The figure below provides a timeline for the papers. The design under consideration, the parameters of the FNS, and the name of the FNS change over the course of time and this section (with Figure 2.1 below) is meant to provide the reader with clarity on the nature of the FNS and its relationship to the optimization problem formulations that make up the basis of the present thesis.

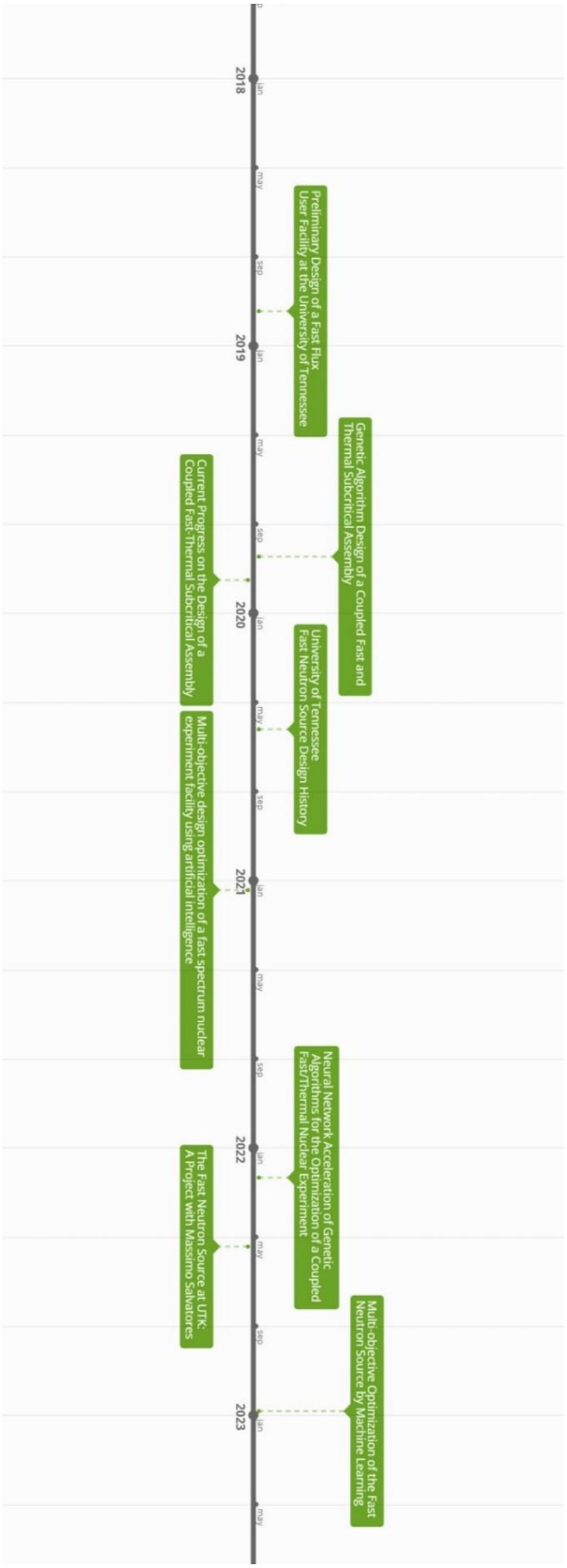


Figure 2.1: Timeline of FNS Papers

"Preliminary Design of a Fast Flux User Facility at the University of Tennessee" [22]

Although the idea of the FNS in some form dates back to at least 2016, the first publication on the topic was published in 2018. At the time of the publication Dr. Vladimir Sobes was a researcher at Oak Ridge National Laboratory but was still collaborating on the design of the FNS. In this early conception of the FNS a sheet of 9.75% enriched uranium separated a designated fast section of the FNS from the thermal section of the FNS (see Figure 2.2). The sheet separating the fast and thermal sections was intended to reduce the thermal spectrum leaking into the fast section and to provide additional fast neutrons to both sections. The early conception did not have twenty plates for each cassette, rather cassettes in thermal section contained enriched uranium rods laid length-wise suspended in high density polyethylene. In the fast section, there would also be uranium rods laid lengthwise but they had different dimensions and were combined in checkerboard pattern with either lead, sodium, or salt (as shown in Figure 2.3).

"Genetic Algorithm Design of a Coupled Fast and Thermal Subcritical Assembly" [23]

This is an earlier article on the FNS and lays out some basics. It talks about the need to reduce cross section uncertainties to support fast reactor design, why a coupled fast-thermal design approach was chosen, and the potential utility of a genetic algorithm for nuclear optimization for a system like the FNS. The design that was laid out in this paper is not the one that this thesis studies and there are many differences such as the cadmium-uranium shutdown plates, the use of 6 by 6 patterns of materials within cassettes (see Figure 2.3 above), the use of lead, the optimization was over four continuous variables rather than 150 integer variables, the three objectives were combined into a single objective, bespoke crossover and mutation operators were used. The paper did conclude that there was not a definitive benefit to having cadmium and uranium separators between zones 2 and 3. The paper also concluded that a genetic algorithm could be used to effectively search the FNS design-space, and that future work would include matching the representativity of the flux spectrum to particular applications and allowing the algorithm to select the location of various materials within the FNS.

"Current Progress on the Design of a Coupled Fast-Thermal Subcritical Assembly"[24]

This is the first paper to introduce the use of twenty plates of swapable plates in each cassette. It also introduced the use of the symmetry patterns which are differentiated for each zone. The symmetry pattern is slightly different from the one used on the present thesis however. The genetic algorithm used on the paper was not Gnowee and was a bespoke creation specifically for the FNS. Like the present thesis, the paper also utilized a k-eigenvalue pre-calculation step in addition to a source driven calculation. The fitness function for the paper is not specified explicitly but is stated to be a scalar combination of four

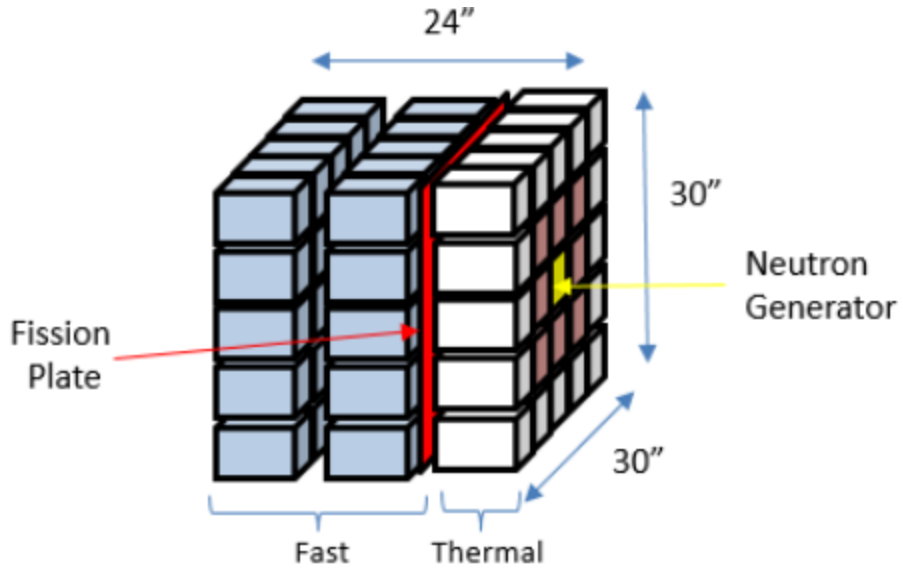


Figure 2.2: Early Separated Design of the FNS(reprinted from [22])

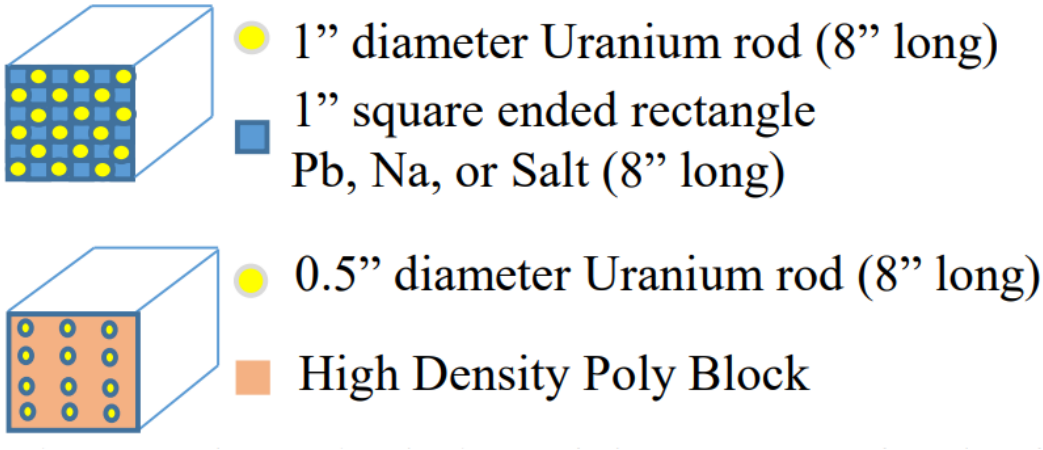


Figure 2.3: Early Rod-Based Design of the FNS(reprinted from [22])

objectives (rather than a true multi-objective optimization): a term based on the fuel mass, a term based on the ratio of fast flux to other flux, a term based on the fast flux per source neutron, and a term based on the distance from the ideal k value of 0.95. The relative weighting of these terms is left unspecified. The paper concludes that there may be a trade-off between the ratio of fast flux to non-fast flux and the amount of fast flux per source neutron.

"University of Tennessee Fast Neutron Source Design History" [25]

This paper contains no new design of the FNS or analysis of the FNS but contains important information on the design rationale for various features of the design changes over time.

"Multi-objective design optimization of a fast spectrum nuclear experiment facility using artificial intelligence" [12]

This paper does apply NSGA-II to the FNS (much like this thesis), and it uses the same set of plate materials. NSGA-II is not nested within Gnowee, but rather a different genetic algorithm not specified in enough detail to recreate it. The URR formula is derived in this paper, but rather than use it directly, the paper uses it to justify its own two objectives. The paper states that: "It is assumed that an FNS configuration with an RF value that approaches 1.0 would share nuclear data uncertainty with the target reactor design that it is emulating." RF here stands for Representativity Factor and it can be calculated with the formula:

$$RF = \frac{\phi_{App}^{T}\phi_{Exp}}{\sqrt{\left(\phi_{App}^{T}\phi_{App}\right)\left(\phi_{Exp}^{T}\phi_{Exp}\right)}} \tag{2.1}$$

Where ϕ_{Exp} and ϕ_{App} are column vectors representing the neutron energy spectra in the experiment and application respectively. Many papers that analyze the value of various integral experiments use some variation of this kind of representativity factor.

The two objectives are the replication of a specified neutron spectrum and the maximization of neutron flux. It introduces the symmetry pattern simplification used in the optimizations used in this thesis but for the particular optimization used in the paper it was focused on a center 15 inches and assumes the plates there to be infinitely reflected in two dimensions.

"Neural Network Acceleration of Genetic Algorithms for the Optimization of a Coupled Fast/Thermal Nuclear Experiment" [26]

Dr. Pevey's paper uses the NSGA-II genetic algorithm to optimize the FNS for three objective functions: maximizing the flux per source particle, maximizing the representativity of the flux spectra, and maximizing the change in the k value of the FNS when placing a target material in the experimental volume. The objectives were used as imperfect stand-ins for the true objective which was to "produce configurations which minimize the uncertainty on a target reactor concept propagated from nuclear data." A version of the NSGA-II algorithm was also run that used a convolutional neural network as a surrogate transport solver in place of MCNP. The surrogate model was shown to be a useful acceleration method.

The present thesis uses a perfect quantitative version of the objective, making it an advancement of Dr. John Pevey's work. Additionally, the NSGA-II is used in a different manner in the present work as it is inserted into Gnowee.

"The Fast Neutron Source at UTK: A Project with Massimo Salvatores" [27]

This article, dedicated to the memory of Massimo Salvatores, provides an overview of the history of the development of the idea of FNS at UTK with particular attention paid to the contributions of Massimo Salvatores. As befitting, the original moniker of the FNS, the system was originally conceptualized as being of special use to the development of fast spectrum advanced reactor designs by helping reduce their modeling uncertainties as an integral experiment. This paper also presents the URR formula again and lays out future avenues of research.

"Multi-objective Optimization of the Fast Neutron Source by Machine Learning" [28]

This thesis by John Lockwood Pevey claims the following four things as original contributions:

1. "Develop[ment] of a genetic algorithm for the multiobjective optimization of nuclear experiments"
2. "Develop[ment] methodologies for the use of feature-extracting neural networks to be used as a surrogate models for neutronic calculations."
3. "Develop[ment] methodologies for the use of gradient descent with directly calculated sensitivities (dk-eff/k-eff/dΣ/dΣ) for nuclear systems."
4. "Implement these methods (#1, #2, #3) into a Fast Neutron Source optimization (#1) as a proof of concept for their applicability to nuclear design."

The claimed contributions in his thesis largely tie back to his papers discussed above and going over them again would be redundant. Dr. Pevey also performs a literature survey of the application of genetic algorithms to nuclear engineering systems consisting of eleven papers (over 9 pages), which this thesis

seeks to avoid repeating, but which readers are directed to if they desire additional coverage of the state of the literature.

Chapter 3: Changes to Gnowee

This chapter[6] is meant to walk the reader through the changes that were made to the Gnowee program, what the rational for the changes were, their implementation, and relate remarks that may be helpful to the user in understanding the program and its changes.

3.1 The Evolution and Mutation of the Gnowee Program

Before discussing the changes made to the Gnowee program for the present thesis, some background on the history of the Gnowee program is due. Figure 3.1 shows the family tree of iterations of the program and where current and future work lies in relation to previous work on the program. The essential elements and differences between the various editions are discussed below.

Gnowee/COEUS 1.0

Despite the name, the original version of Gnowee that was created was Gnowee/COEUS 1.0. It had several limitations and among them are: it only works for the anticipated objective functions, anticipated types of MCNP geometries, it would rerun MCNP geometries that had already been run, and it would only work on computational clusters very similar to that of the University of California Berkeley.

Gnowee/COEUS 2.0

The second edition of Gnowee was developed by Dr. Sandra Bogetic at the same time as Gnowee 1.0 was being developed. Gnowee/COEUS 2.0 shared most of the limitations present in Gnowee/COEUS 2.0 but featured more flexibility in the MCNP geometries the user could pick from, more freedom on the type of objective function the user could use, and the use of ADVANTG to dynamically speed up MCNP jobs throughout the optimization. The Automated Variance Reduction Generator (ADVANTG) is a program that employs a variety of techniques to optimize an MCNP simulation to improve the statistical quality of results for a given amount of computation [20].

Gnowee 1.0

Gnowee 1.0 was meant as a bare implementation of the algorithm for purposes of scientific study of its characteristics. In that context, things like redundant evaluations are acceptable. Gnowee 1.0 was

[6] Portions of this chapter are from previous work [1], [2],[3],[4], by Alexander DePillis, Vladimir Sobes, J. Wesley Hines, and Sandra Bogetic

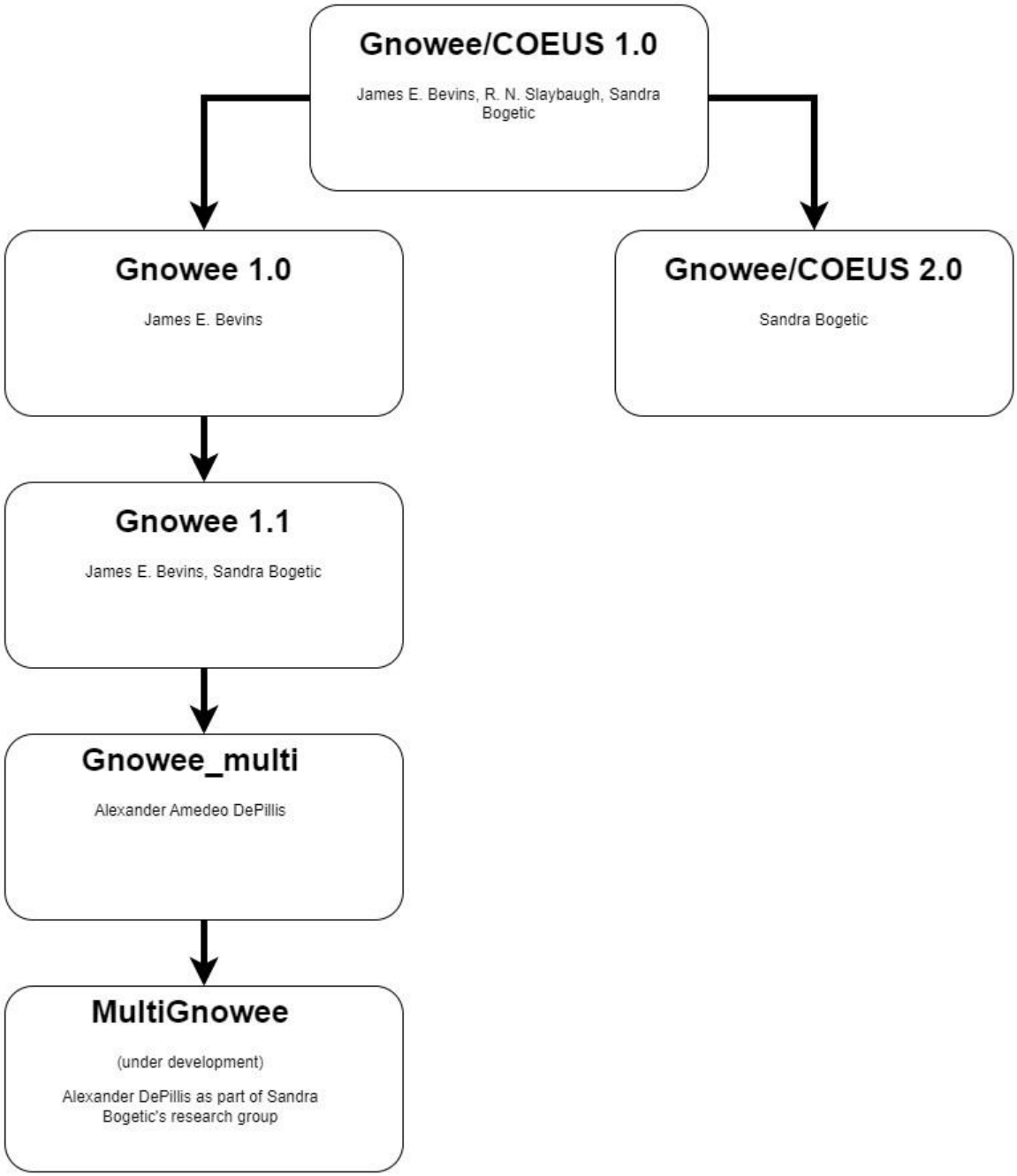


Figure 3.1: The family tree of Gnowee

admirably made open source and publicly available on github [29] alongside the paper that widely announced it [14].

Gnowee 1.1

Gnowee 1.1 is a slightly updated version of the code available for the Gnowee 1.0 release hosted on the exact same Github repository[29]. Nothing changed in the main structure of the Gnowee program, but additional plots, plotting tools, and results were added and tweaked to be consistent with James Bevins' dissertation [30].

Gnowee_multi

This is the edition of Gnowee created in this thesis and this chapter covers how it has been changed from Gnowee 1.1.

MultiGnowee

A planned future edition of Gnowee that is under development, full set of improvements is yet to be determined.

3.2 Conversion to Python 3

Previous editions of Gnowee, were written in Python 2.x, which is no longer a supported program since 2020 [31]. Unfortunately, Python 3 is not backwards compatible with Python 2, due to differences in syntax, so a conversation had to be performed. The conversion was done with the help of the `lib2to3` module [32] which automated the conversion of the files. The conversion will help the Gnowee program remain relevant in the future on modern systems and enable it to use modern programming libraries.

3.3 Implementing NSGA-II into Gnowee

An elitist genetic algorithm creates a child population of feature vectors from a parent generation of feature vectors, evaluates all the solutions, and then lets the best N (where N is a user specified population size) solutions continue into the next generation regardless of whether they are parents or children. In the multi-objective context however, it is not so clear how to select which feature vectors are "best" and should thus continue to the next generation. Consider two possible feature vectors $\vec{x}_A$ and $\vec{x}_B$ that are evaluated across two different objective functions and have the associated fitnesses $f_{A,1}, f_{A,2}, and\ f_{B,1}, f_{B,2}$ respectively. If $f_{A,1}$ is better than $f_{B,1}$, but $f_{B,2}$is better than $f_{A,2}$, is A or B more worthy of continuing to the next generation?

Domination is an important concept in multi-objective optimization. If the fitnesses of feature vector A are better then or equal to the corresponding fitnesses of feature vector B, then A is said to dominate B. If there is no feature vector that dominates A, than A is said to be undominated. The set of all feature vectors that are undominated is called the Pareto front. The set of all feature vectors that are only dominated by members of the Pareto front can be thought of as a second front. The set of all feature vectors only dominated by members of the first two fronts can be thought of as the third front and so on. Thus one can assign every feature vector ever evaluated to a front. An illustrative example of individuals being sorted into fronts is shown in Figure 3.2.

The nondominated sorting genetic algorithm II (NSGA-II) is so called because it sorts feature vectors based on their degree of domination. First NSGA-II sorts all the available feature vectors from the parents and children into fronts. Then, to fill the N population slots available for the next generation it starts by including all the members from the first front, then moves on to the second front and so on. However, it still needs a mechanism to decide what to include when it reaches a front that it can only partially include in the N slots available. This mechanism is the crowding distance sorting operation. The relationships of these procedures are shown in Figure 3.3 and pseudo-code for the algorithm is presented in Figure 3.4.

The crowding distance sorting operation functions as a kind of tie-breaker within a partially included front and breaks ties with the goal of preserving diversity. This operation proceeds as follows for a front to calculate the crowding distances ($I[i]_{distance}$) for each of the *l* individuals (*i*):

For each *I*, set $I[i]_{distance} = 0$

For each objective *m*:

I = sort(*I*,*m*)

$I[1]_{distance} = I[l]_{distance} = \infty$

For *i* =2 to (*l*-1):

$$I[i]_{distance} += \frac{I[i+1].m - I[i-1].m}{f_m^{max} - f_m^{min}}$$

The original paper says that "f_m^{max} and f_m^{min} are the maximum and minimum values of the *m*th objective function." This is vague and could mean the maximum and minimum: theoretical possible values that objective function m could take, values that have ever been generated over the course of the algorithm, or of the values in the current front for which the crowding distance is being calculated. Most implementations of the algorithm found online seem to use the third interpretation and so that is what is used in this work.

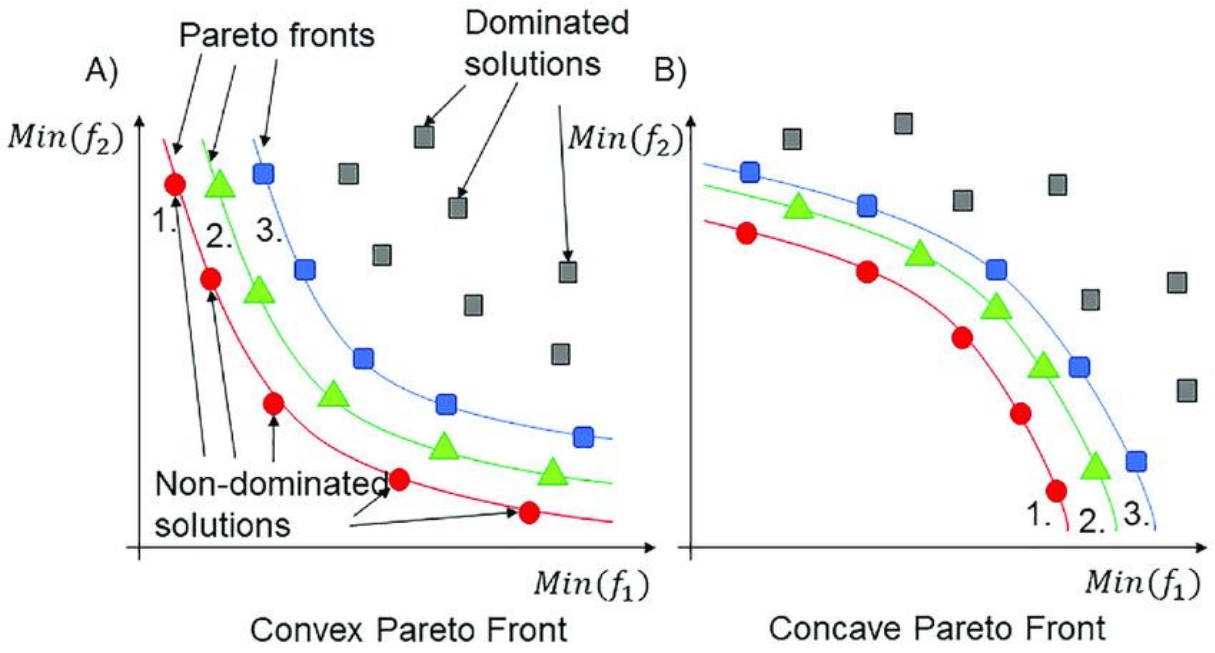


Figure 3.2: Illustrative examples of individual solutions sorted into fronts (reprinted from [33]).

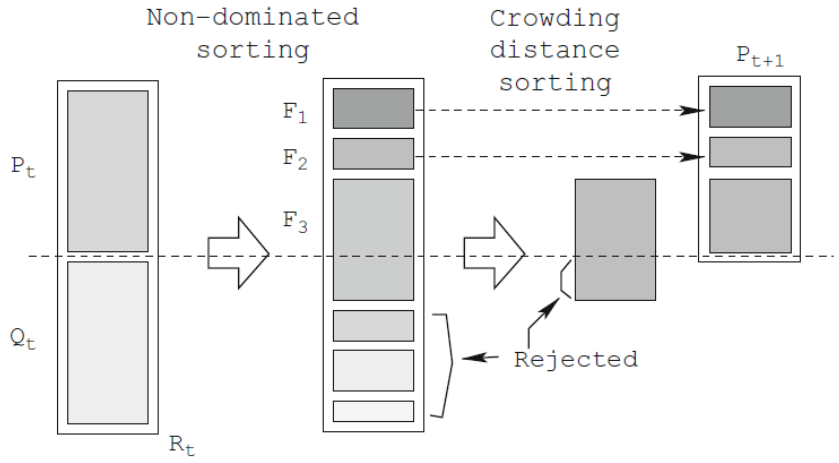


Figure 3.3: The Sorting Mechanism in NSGA-II (reprinted from [21])

$R_t = P_t \cup Q_t$	Combine parent and offspring population
$F = fast - non - dominated - sort(R_t)$	$F = (F_1, F_2, I)$, all nondominated fronts of R_t
$P_{t+1} = \emptyset \; and \; i = 1$	
$until \; \lvert P_{t+1} \rvert + \lvert F_i \rvert \leq N$:	Until the parent population is filled
Crowding-distance-assignment(F_i)	Calculate crowding-distance in F_i
$P_{i+1} = P_{i+1} \cup F_i$	Include *i*th nondominated front in the parent population
$i = i + 1$	Check the next front for inclusion
Sort($F_i \prec_n$)	Sort in descending order using $\prec_n$
$P_{t+1} = P_{t+1} \cup F_i[1: (N - \lvert P_{t+1} \rvert)]$	Choose the first $(N - \lvert P_{t+1} \rvert)$ elements of F_i
$Q_{t+1} = make - new - pop(P_{t+1})$	Use selection, crossover, and mutation to create a new population Q_{t+1}
$t = t + 1$	Increment the generation counter

Figure 3.4: NSGA-II Commented Pseudo-Code (reprinted from [26])

Importantly, the main features of the NSGA-II algorithm (the sorting by front and crowding distance) do not make demands of the method of creating new feature vectors. As such, the important parts of the algorithm can be implemented with nearly any set of search heuristics (methods of creating new feature vectors for evaluation). Gnowee, on the other hand, is composed of various methods of generating new feature vectors (which it calls children before their incorporation into the population of parents) then using the `population_update` function to evaluate the fitness of the children and determine which ones will be included in the population of parents (see Figure 3.5 below).

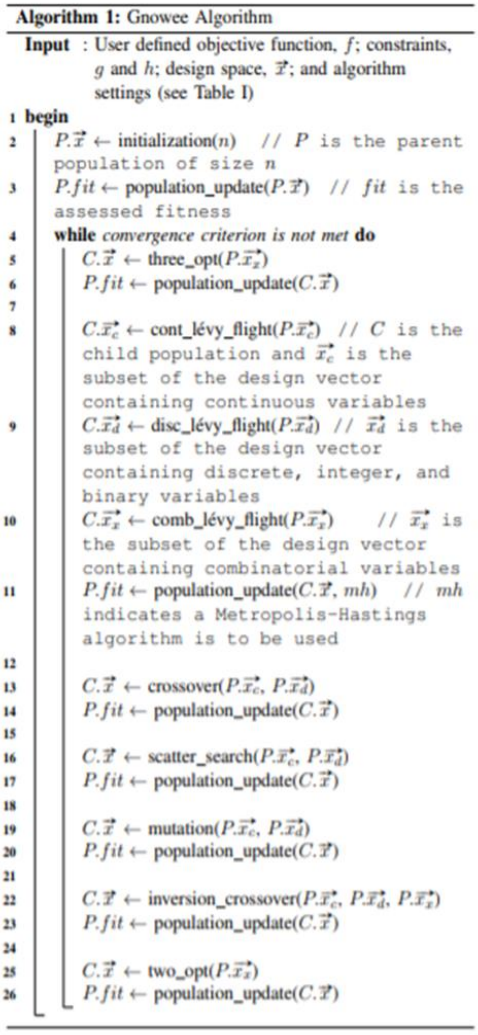
Thus the approach to integrating the two algorithms was to put the main features of NSGA-II inside `population_update` as the multi-objective alternative to the normal procedures of that function. The resulting function was called `population_update_multi` and it would become one of many files and functions to be named with the "`_multi`" suffix to indicate a compatibility with multiple objective functions.

One of the goals that was pursued in the creation of the new version of the Gnowee program was to maintain backwards compatibility with Gnowee 1.1 where possible. As such, one of the first things `population_update_multi` does is detect the number of objective functions the program is being run with. If there is exactly 1 objective function, it runs the older version of `population_update` from Gnowee 1.1, if there is more than one objective function, it runs the sorting procedures from NSGA-II. The NSGA-II procedures utilize a new file called `NSGAmethods.py` which helps interface with an externally supplied code [34] in a folder called `nsga2` that implements the actual sorting.

3.4: Bad Vector Determination Step

Early on in the work for this thesis, it was realized that when randomly generating configurations of fissile material, there was a chance that a configuration would be critical or super-critical. When such a system is run in a Monte Carlo source-driven simulation, it is likely to cause error or never complete[7]. Thus, there was a need for a system that would determine if a feature vector would result in a near critical system before it was submitted to the normal objective function that would normally just perform a source-driven Monte Carlo simulation.

[7] In the case of MCNP, it may result in the creation of increasingly large files to store data used for the simulation. In performing optimization runs, some configurations of the FNS were witnessed to be causing the creation of hundreds of Gb of files without end (before this researcher manually stopped the simulations). If left unchecked this could have caused the crash of the departmental computing cluster, and given that the optimization runs can be expected to go for weeks researchers cannot be expected to be on hand to stop this kind of problem.



Algorithm 1: Gnowee Algorithm

Input : User defined objective function, f; constraints, g and h; design space, $\vec{x}$; and algorithm settings (see Table I)

begin
- $P.\vec{x} \leftarrow$ initialization(n) // P is the parent population of size n
- $P.fit \leftarrow$ population_update($P.\vec{x}$) // fit is the assessed fitness
- **while** *convergence criterion is not met* **do**
  - $C.\vec{x} \leftarrow$ three_opt($P.\vec{x}_x$)
  - $P.fit \leftarrow$ population_update($C.\vec{x}$)
  - $C.\vec{x}_c \leftarrow$ cont_lévy_flight($P.\vec{x}_c$) // C is the child population and $\vec{x}_c$ is the subset of the design vector containing continuous variables
  - $C.\vec{x}_d \leftarrow$ disc_lévy_flight($P.\vec{x}_d$) // $\vec{x}_d$ is the subset of the design vector containing discrete, integer, and binary variables
  - $C.\vec{x}_x \leftarrow$ comb_lévy_flight($P.\vec{x}_x$) // $\vec{x}_x$ is the subset of the design vector containing combinatorial variables
  - $P.fit \leftarrow$ population_update($C.\vec{x}$, mh) // mh indicates a Metropolis-Hastings algorithm is to be used
  - $C.\vec{x} \leftarrow$ crossover($P.\vec{x}_c$, $P.\vec{x}_d$)
  - $P.fit \leftarrow$ population_update($C.\vec{x}$)
  - $C.\vec{x} \leftarrow$ scatter_search($P.\vec{x}_c$, $P.\vec{x}_d$)
  - $P.fit \leftarrow$ population_update($C.\vec{x}$)
  - $C.\vec{x} \leftarrow$ mutation($P.\vec{x}_c$, $P.\vec{x}_d$)
  - $P.fit \leftarrow$ population_update($C.\vec{x}$)
  - $C.\vec{x} \leftarrow$ inversion_crossover($P.\vec{x}_c$, $P.\vec{x}_d$, $P.\vec{x}_x$)
  - $P.fit \leftarrow$ population_update($C.\vec{x}$)
  - $C.\vec{x} \leftarrow$ two_opt($P.\vec{x}_x$)
  - $P.fit \leftarrow$ population_update($C.\vec{x}$)

Figure 3.5: Gnowee Algorithm Pseudo-Code(reprinted from [14])

This was the problem-specific inspiration for a new generic feature in the Gnowee program. This feature is called the "Bad Vector Determination Step" and it provides the user with the ability to perform a pre-evaluation of feature vectors before they are submitted to the normal evaluation program. The functionality is turned off by default and the user must use a keyword (`BadVectorDetermination = True`) to turn it on when they initiate Gnowee_multi. If they choose to use it they must provide a file named "**UserBadVectorDetermination.p**y" in the same folder as the rest of the program. This python file must define a function called "`getVectorDetermination`" which accepts a list of feature vectors and the function must return a list of 1s and 0s corresponding to the list of feature vectors. A 0 tells the modified Gnowee that the corresponding feature vector is not acceptable and should not be given to the full objective function. Feature vectors that are determined to be unacceptable are still recorded in the past results system described below, but they are recorded as having a fitness equal to the penalty the user specifies. If the user forgot to change the penalty value from its default value of 0.0 (which is functionally no penalty at all), the fitness of the unacceptable feature vectors is recorded in the past results system as 1.0. If the user does not chose to enable the Bad Vector Determination Step, all feature vectors are assumed to be acceptable for submission to the normal objective function and Gnowee's normal constraints system is assumed to be able to handle the various constraints the user wishes to implement after the objective function is evaluated.

3.5: Past Results System

In some cases (such as the MCNP simulations used in parts of this thesis), the objective function can be very computationally expensive to evaluate and calls to the full objective function should be minimized where possible. The default `population_update` from **Gnowee 1.1 calls the objective function for each child feature vector to be evaluated twice**. This means the objective function is being called on feature vectors that have already been evaluated at least 50% of the time. Additionally, under some circumstances, the function will call the objective function on previously evaluated "parent" feature vectors. There is also the possibility that a child vector is generated by the search heuristics multiple times over the course of the Gnowee optimization. These facts motivated the creation of a system that would ensure that the full objective function was only called upon to perform truly novel evaluations of feature vectors.

The resulting system is called the Past Results System. It is partially implemented in the `population_update_multi` function and in the file `pastResults.py` (and its associated functions). The basic idea of the system is that every feature vector that would be evaluated is first checked against a library of all the feature vectors that have been previously evaluated; and if it was

previously evaluated, the corresponding fitness value(s) are simply pulled from the library instead of calling the objective function(s). The library of results is implemented as a file (named with a keyword option) containing a `resultsLibray` type object that is saved and loaded periodically with the help of the pickle module. In theory the system could receive a minor speed up by omitting the use of an external file. In practice however, the Past Results System has the additional purpose of providing crash recovery services. The hardware systems that Gnowee_multi has been run on so far have proven temperamental at times and optimization runs have crashed wasting weeks worth of computational time, saving to an external file allows a subsequent optimization run to skip evaluations that were performed on a previous run (thus preserving the benefits of the computational time previously expended). This is especially powerful because Gnowee_multi will produce the same "random" mutations and crossover selections as it did on the previous run if all the fitnesses match up, thus allowing a re-run to effectively pick up where the previous run crashed. To help with this re-running, the Past Results System will attempt to detect if a named results pickle file already exists (and thus avoid overwriting it). All-in-all the Past Results System should *reduce the number of objective function calls (and thus runtime[8]) by at least 50%, making it a significant improvement over the previous code.*

3.6 Batch Submission System

In Gnowee 1.1, the feature vectors undergoing evaluation are evaluated one at a time, but structurally this need not be the case. Evaluations are only performed within the `population_update` function and the order which they are evaluated within that function does not affect what child feature vectors the search heuristics will produce in the future. Nothing about the *conceptual* structure of Gnowee depends on feature vectors being evaluated one at a time. Furthermore, Gnowee thus far has been used by users with access to computational clusters that can perform computations in parallel.

As a result, there was a desire to create a system within Gnowee that would allow for the simultaneous evaluation of many feature vectors and this was to be called the Batch Submission System. Unfortunately, the program code to evaluate expensive objective functions in parallel will inevitably be specific to the objective function(s) the user wishes to evaluate and the computational system the user is running Gnowee on. This meant that unlike the features of `Gnowee_multi` described up until now, the Batch Submission System's code requires modifications to suit the particular circumstances of the user. For this reason the main file that implements the system is called `BatchSubmit_beta.py`, recognizing that it will need to be changed in the future. The current system implemented in

[8] If objective function evaluations are performed in series (as they were in Gnowee 1.1, or still are without the batch submission system), runtime should scale roughly linearly with the number of objective function calls

`BatchSubmit_beta.py` is designed to assist in running MCNP on the UTK Nuclear Engineering department cluster and to work with the types of optimization problems that are the focus of this thesis.

With those caveats, the system was designed to be easy to modify and mold to a particular user's needs if their circumstance satisfies a couple of assumptions that make it similar to the optimization problems discussed in this thesis. These assumptions are that:

- The user's objective function(s) depend on results in files that are the output of case-by-case inputs to a piece of processing software (like MCNP)
- The inputs to the processing software change depending on the contents of the feature vector
- The input to the processing software can be generated by substituting particular values into specific locations in a template file (which will be specified at the start of Gnowee_multi)
- The substitutions can be determined based on the feature vector
- The 'qsub' command is how processing jobs are submitted to the computational cluster

To help ensure backwards compatibility, the Batch Submission System is disabled by default and must be enabled through the use of the BatchSubmit keyword. If the system is turned on, then *the user-supplied objective functions must be written to accept both a feature vector argument and a filename argument.*

The keyword enables an alternative set of commands inside `population_update_multi` and triggers a variety of mechanisms. The main bit is that it runs the `Batch_Submit` function in the file `BatchSubmit_beta.py`. The function is given as inputs: a list of feature vectors to be evaluated and which have been screened by the Past Results System (and optionally the Bad Vector Determination Step), the name of the aforementioned template file, and a prefix. The file, `BatchSubmit_beta.py` contains several global variables whose values are preserved between different calls to the `Batch_Submit` function. Among these variables are a set of counters for keeping track of the number of feature vectors evaluated so far (for the purposes of assigning ID numbers), and a string storing a template of a `.sh` script for submitting an MCNP job to the UTK NE computational cluster (and for generating a file with the same name as the input file but with "`_done.dat`" at the end of its name). If the user is not using the UTK Nuclear Engineering Department computational cluster they should modify this template script to be able to submit a job of their desired type to the cluster they are running their program on.

Next, `Batch_Submit`, proceeds to use a function called `translate_feature_vector` as it loops through feature vectors one at a time. `translate_feature_vector` is meant to be

programmed by the Gnowee_multi end user in a file called `translate_feature_vector.py` and not merely modified. The following are the requirements for this file the user must program:

- It must be a python 3 file named `translate_feature_vector.py`
- It must have a function named `translate_feature_vector` to enable the import statement and function calls inside the Batch Submission System
- The function must accept a single Gnowee feature vector as its only argument
- The function must return two lists: a list of variable names, and a list of variable values (in that return order)
- The function is meant to assist the Batch Submission System with the creation of a realized input file from the template file and the feature vector Gnowee_multi generated
- The list of variable names should be a list of strings, each of which should match the text in the template file the user wishes to replace when generating the input file that corresponds to the feature vector
- The list of variable values should also be a list of strings and should be of the same length as the list of variable names since each variable value should correspond to the string in the template file which it will replace

Next `Batch_Submit` creates the input file using the a simple find-and-replace procedure on the template file with the substitutions the user supplied. The input file name is a combination of the user supplied prefix argument, the number of evaluations performed so far (from the global variables), and "`.inp`". It also generates a series of `.sh` script files named in a similar fashion. The script template also tells the Batch Submission system to expect the output files to be named the same as the input files but with "`.inpo`" instead of "`.inp`". After looping through the creations of the script files and input files, the `Batch_Submit` function executes all of the `.sh` scripts with the help of the '`qsub`' command. If '`qsub`' is not used on the user's system, the user is advised to modify line 193 of `BatchSubmit_beta.py`.

Afterwards, the `Batch_Submit` proceeds to wait on the completion of all the processing of the input files. It does so with the help of the `wait_on_jobs` function in the `BatchSubmit_beta.py` file. This program waits on the completion of the processing of the input files by periodically checking the directory for the presence of files containing "`_done.dat`" and the appropriate ID numbers in their file names. This assumes that these files are generated in the location where the program is being run. If this is not the case, the user is advised to consider modifying line 63 of `BatchSubmit_beta.py`. After all the jobs have completed the Batch Submission System attempts to remove all files in the

execution directory with the strings "`inp.sh`", "`inps`", and "`inp_done.dat`". The final part of the `Batch_Submit` returns a list of the names of the output files.

The remainder of the system is executed within `population_update_multi` which receives the list of output file names. It takes these names and gives them one-at-a-time to the user supplied objective function along with their associated feature vectors. Note that this means that utilizing the Batch Submission System requires the user-supplied objective function to accept two arguments when called: the objective function and an output file name (in that order). After getting the fitness values, entries are added to the Past Results System and files in the directory containing "`.inp`" are deleted (thus deleting the output files) and the Batch Submission System has concluded.

It should be noted that parts of the Batch Submission System for associated with waiting on the completion of jobs reuses code written by John L. Pevey with his permission.

3.7 Backwards Compatibility

Throughout the development of Gnowee_multi, backwards compatibility with Gnowee 1.1 was made a top priority. Gnowee_multi should be able to run any optimization problem Gnowee 1.1 could and get the same final answer. An attempt was also made to ensure that the same scripts that initiated runs of Gnowee 1.1 could initiate runs of Gnowee_multi with minimal modification. Ensuring this would be the case while still improving the speed and suite features of the program required making compromises (such as the strange structure of the Batch Submission System) and giving the user more control via keywords.

Throughout the proceeding sections on the new features and systems of Gnowee_multi, references have been made to various keyword arguments whose use will be elaborated on in Table 3.1. These keyword arguments are specified in the same line the user uses to specify the other characteristics of the optimization (like the maximum number of generations). A description of each is presented in the table below.

The notable exceptions that prevent total backwards compatibility with scripts designed to submit Gnowee 1.1 optimization processes are:

- The switch to python 3
- The different name of the program
- Objective functions being supplied to Gnowee as an entry in a list
- The limitation on the number of individuals evaluated in the initialization step

To improve the efficiency of Gnowee in higher dimensional optimization problems, the number of individuals evaluated in the initialization step was capped at ten times the size of the population the user specifies. This was not mentioned previously due to the brief nature of the change.

Table 3.1: New Keyword Arguments Implemented in Gnowee_multi

Keyword Argument	Allowed Values	Description and Purpose	Default Value if unspecified
`BatchSubmit`	True/False	Specifies whether the Batch Submission System should be enabled	False
`templateFile`	String including file type	The name of the template file to be used in the Batch Submission System	“batch_test.txt”
`prefix`	String	A string prefix to be attached to input and output file names generated by the Batch Submission System	“MCNP_File_Num”
`FileName`	String not including file type	The text name to be used for the creation or re-use of a Past Results System library file	"DefaultResultLibName"
`BadVectorDetermination`	True/False	Specifies whether the Bad Vector Determination Step should be taken	False

Chapter 4: The Uncertainty Reduction Ratio

This chapter[9] contains content that was originally written as paper that is currently under review in the Journal of Nuclear Engineering and Design, a preprint of which is available online[2]. The purpose of this chapter is to present and verify a formula for predicting the uncertainty reduction ratio that can be expected from a given experiment for a given application.

4.1: Background on the Uncertainty Reduction Ratio Formula

The Uncertainty Reduction Ratio (URR) formula (Equation 4.1) was first presented in a 2022 paper and predicts the ratio of data-induced uncertainty on an application's parameter of interest after an integral experiment is performed and the cross section data is adjusted (δ_a') to the application's data-induced uncertainty on the parameter of interest before the integral experiment was performed (δ_a) [11].

$$\frac{\delta_a'}{\delta_a} = \sqrt{1 - \frac{\frac{(S^T MG)^2}{(S^T MS)(G^T MG)}}{1 + \frac{V}{G^T MG}}} \tag{4.1}$$

Where S is a column vector of the sensitivities of the application's criticality to the cross sections, G is a column vector of the sensitivities of the integral experiment's measurement (in this case flux ϕ in the experiment volume) to the cross sections, M is a covariance matrix of the cross section data representing prior knowledge of cross sections and correlations between uncertainties, V is the variance of the measurement of the integral experiment[10]. The derivation is based on Bayesian updating and relies on the following assumptions:

1. The priors of the nuclear data are normally distributed
2. The likelihood distributions involved are normal distributions
3. That the application response changes linearly with respect to the nuclear data
4. That all nuclear data is being adjusted
5. That a single measurement of the experiment system is being made before adjustment[11]

[9] Portions of this chapter are from previous work [1], [2],[3],[4], by Alexander DePillis, Vladimir Sobes, J. Wesley Hines, and Sandra Bogetic

[10] In chapter 5 V is calculated as being based on a perfect detector with Poisson statistics, but this is not the only method of assigning a value to V, the experimentalist should be consulted to determine a proper model of V that incorporates how it will change for different variations of the experiment system

[11] A variation of the URR formula exists that does not rely on this assumption.

Of these assumptions, the most limiting ones are the third and fourth. The third assumption means that Equation 4.1 will only apply when relatively minor adjustments are being made to the nuclear data. The fourth assumption is nearly impossible to satisfy for all but the most simple systems since some pieces of nuclear data are very difficult to get sensitivities for.

Since the formula is predicting the average reduction in uncertainty on the parameter of interest due to the adjustment of various pieces of nuclear data, one should also know how the corresponding adjustment itself is calculated. Equations 4.2 and 4.2 (from [35]) give these adjustments.

$$M' = (M^{-1} + G^T V^{-1} G)^{-1} \tag{4.2}$$

$$P' = P + M' G^T V^{-1} (D - T) \tag{4.3}$$

In the above equations, P is a column vector containing the prior (prior to the measurement of the experiment system) best estimates of the nuclear data, M is the prior covariance matrix that corresponds to P (the same M as in Equation 4.1), D is the scalar value of the result from the measurement performed on the experiment system, T is the calculated theoretical value for the measurement result based on the prior estimates of the nuclear data, V is the variance on the measured value D (same as in Equation 4.1), G is a column vector of the sensitivities of the integral experiment's measurement to the cross sections, P' is a column vector containing the adjusted values of the nuclear data, and M' is its associated covariance matrix.

4.2: Verification of the Uncertainty Reduction Ratio Formula

The URR formula was verified with a multi-step process using the TSURFER, TSUNAMI-3D, CSAS6, and SAMPLER modules of SCALE for a demonstration problem. To verify the URR formula, an alternative formula for the relative average improvement that can be expected from an integral experiment:

$$Average\ Improvement = \sqrt{\frac{\sum_{i=1}^{1000}\left(k_{app,new\ knowledge,i} - k_{app,real,i}\right)^2}{\sum_{i=1}^{1000}\left(k_{app,current\ knowledge} - k_{app,real,i}\right)^2}} \tag{4.4}$$

Where $k_{app,new\ knowledge,i}$ is the i[th] value of the k-eigenvalue calculated for the application using cross sections that were updated based on the integral experiment, $k_{app,real,i}$ is the true value of the k-eigenvalue for the application as calculated with the true cross sections as defined by the i[th] perturbed cross section library, and $k_{app,current\ knowledge}$ is the criticality of the application as calculated with

cross sections that were not updated. Each of these variables is calculated in a step in the process outlined below in Figure 4.1.

Figure 4.1 shows part of the process involved in calculating the average improvement. SCALE and its SAMPLER module work with a set of 1000 perturbed nuclear data libraries. The key aspect of these libraries is that they are perturbed versions of the standard ENDF/B-VII.1 nuclear data library and that the way they are perturbed is statistically consistent with the covariance matrices for ENDF/B-VII.1. This allows us to justify using them as a data set to represent the different possible theoretical "true" cross sections. Given a template geometry, SAMPLER can generate 1000 SCALE input decks for the CSAS6 module, each linked to a different perturbed data library. Thus, SAMPLER was used to generate 1000 input decks for an application system and 1000 input decks for an integral experiment system, and the k-eigenvalue (and an associated uncertainty) was calculated for all of them. To account for the inability of the experimentalist to measure the true k-eigenvalue in their integral experiment, the criticality values for the experiment system ($k_{exp,real,i}$) are adjusted. The measured amount of criticality ($k_{exp,measured,i}$) is drawn from a normal distribution centered at $k_{exp,real,i}$ whose variance is set to some pre-defined level representing an arbitrary amount of experimental uncertainty. The TSUNAMI-3D module of SCALE is used to calculate the sensitivities of the application and integral experiment systems to the underlying nuclear data as well as criticality values for the unperturbed ENDF/B-VII.1 library (thus representing the assumed current state of knowledge). The TSURFER module of SCALE then takes the sensitivity information, current knowledge of the application system ($k_{app,current\ knowledge}$), experimentally measured criticality of the experiment system ($k_{exp,measured,i}$), and uses them to calculate a new k-eigenvalue for the application ($k_{app,new\ knowledge,i}$) that reflects the knowledge gained from the experiment.

For the demonstration problem a pure ^{235}U cylinder 8 cm tall and 8 cm in diameter served as the application and a pure ^{235}U sphere with a radius of 8.7407 cm served as the integral experiment. Average improvement was calculated using the method outlined in Figure 4.1 and Equation 4.4 above for 100, 200, 500 and 1000 pcm uncertainty on the measurement of the k-eigenvalue of the experiment system. URR values were also calculated using MCNP models[12] for the same variance values and the results of both sets of calculations are shown in Figure 4.2 below. Good agreement between the calculations for Equation 4.1 and Equation 4.4 was found with a percentage difference ranging between -3.26% and 1.6%.

[12] These calculations were done using `KSEN` rather than `PERT` so they were unaffected by the bug described in Chapter 5

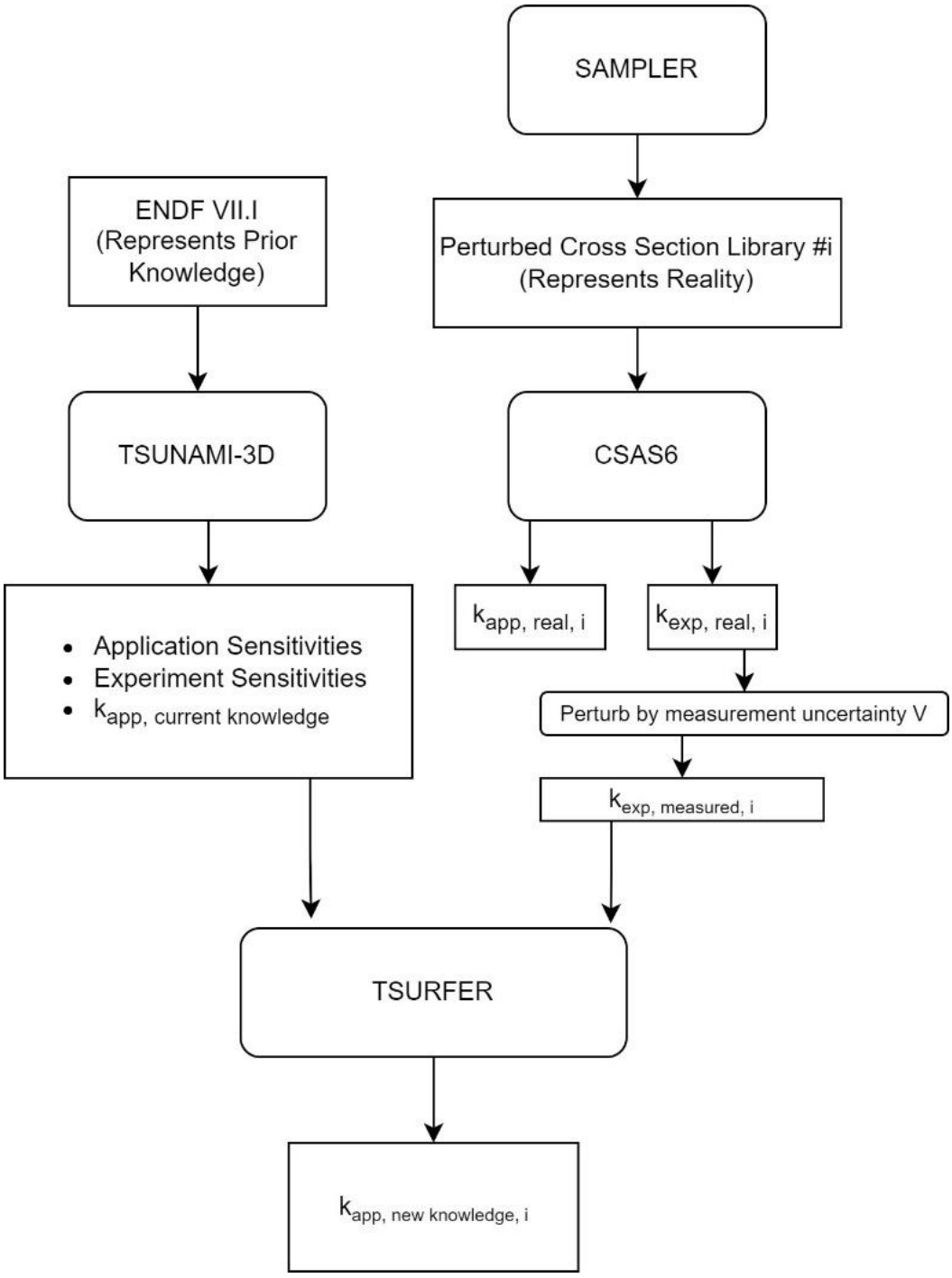


Figure 4.1: Calculation of Verification Data

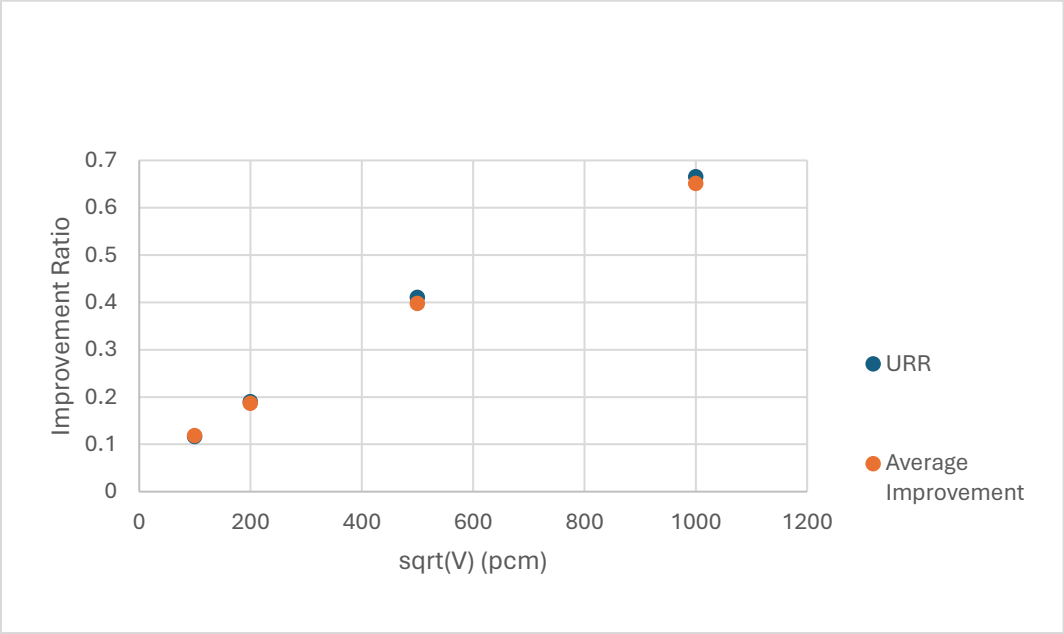


Figure 4.2: Verification Results. URR and Average Improvement vs Measurement Uncertainty.

To verify that these differences were not significant, a bootstrapping procedure was used for the Average Improvement. The bootstrapping procedure used 1000 resamples of 500 values each and the results are given in Table 4.1. Using bootstrapping, 1000 values of the average improvement were calculated (from samples of size 500 drawn with replacement), the average of these 1000 samples was then calculated as well as the standard deviation for the 1000 values of the average improvement. The results suggest that the averages and do not have a statistically large difference from the URR values (they are all within one standard deviation).

Table 4.1: Bootstrapping Results

√V (pcm)	Average Improvement	URR	Bootstrapped Average	Bootstrapped Standard Deviation
100	0.118393979	0.11651560	0.118730609911	0.005033315
200	0.186986376	0.18993562	0.186879519705	0.008259778
500	0.397657802	0.41084146	0.398380460823	0.015918281
1000	0.651551377	0.66542118	0.652852165530	0.022751420

Chapter 5: Optimizations

There were two main optimizations performed for the FNS using Gnowee_multi and which will be covered in this chapter[13]: one multi-objective optimizing based on the ^{35}Cl (n,p) cross section for a Molten Chloride Fast Reactor (MCFR), and one single-objective using a variety of cross sections to optimize for a Sodium Fast Reactor (SFR). As such this chapter is split into two main parts to correspond to each optimization.

The basics of the optimization problems were laid out in the introduction. For both optimizations, a set of simplifications were made to the problem to ease the optimizations. These simplifications were first introduced by Dr. John Pevey in his 2019 article [24].

In order to reduce the size of the design space, a symmetry pattern was applied as seen in Figure 5.1. Each cassette in the same letter-denoted-location in the same zone was given the same pattern of internal plates. Not all plate locations were optimized over. Location F cassettes in zones 1, 2 and 3 were all filled with carbon steel. Locations E and D in zones 1 and 2 are filled with tuning plates, in zone 3 locations E and D are filled with polyethylene plates. One plate location in the middle of zone 2 was specified to always contain a uranium plate to guarantee that there would always be fissile material where the neutrons are initialized in MCNP[14]. After accounting for the symmetry conditions there are 149 effective independently specifiable plates whose materials are optimized over leading to features vectors with a length of 150 (the extra one accounts for the measurement time variable). These constraints also guarantee the presence of 240 steel plates, 480 tuning plates, and 240 polyethylene plates regardless of the feature vector of an individual. The MCFR optimization had the option of salt plates to serve a tuning function, the SFR optimization used sodium metal plates for tuning.

Both optimizations use the URR formula verified in the previous chapter and the Gnowee_multi algorithm outlined in Chapter 3.

$$\frac{\delta_a'}{\delta_a} = \sqrt{1 - \frac{\frac{(S^T MG)^2}{(S^T MS)(G^T MG)}}{1+\frac{V}{G^T MG}}} \tag{5.1}$$

[13] Portions of this chapter are from previous work [1], [2],[3],[4], by Alexander DePillis, Vladimir Sobes, J. Wesley Hines, and Sandra Bogetic

[14] In source-driven calculations, neutrons are started in the source location, but during criticality calculations (such as those used in the bad vector determination step) MCNP requires that some neutron start in a location with fissile material, hence a need for a location guaranteed to have fissile material

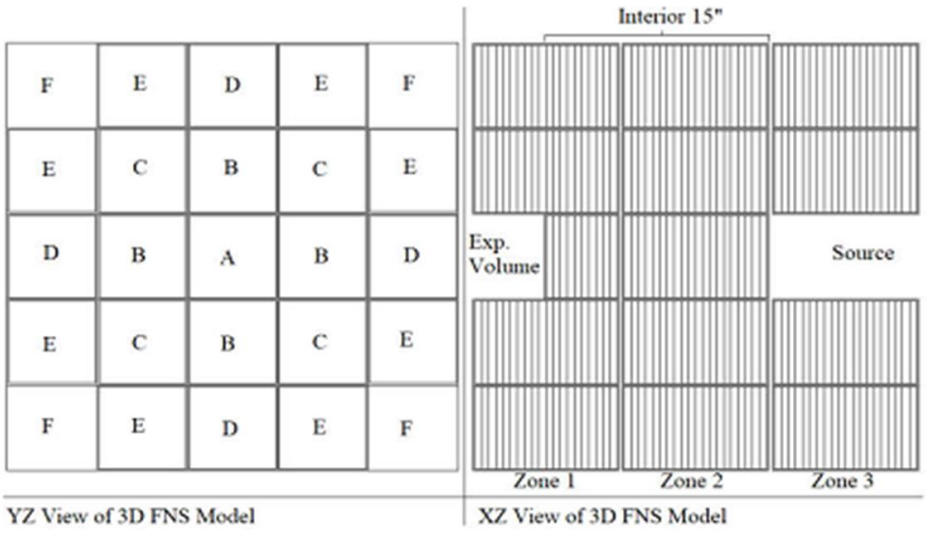


Figure 5.1: Diagram of FNS Design (reprinted from [12])

But the definitions of V[15], S, M, and G differ. ENDF/B-VII.1 was used because it was the most recent dataset for which covariance information was available in the SCALE edition available when research began.

The URR formula is not constrained to a single cross section and applying it to account for multiple pieces of nuclear data merely requires the right formulation for the vectors and matrices. For example, if there are three pieces of nuclear data for reactions A, B, and C, then the vectors and matrices would be formulated the following three equations:

$$\vec{S} = \begin{bmatrix} \frac{\partial k_{App}}{\partial \Sigma_A} \\ \frac{\partial k_{App}}{\partial \Sigma_B} \\ \frac{\partial k_{App}}{\partial \Sigma_C} \end{bmatrix} \tag{5.2}$$

$$M = \begin{bmatrix} \delta_{\Sigma_A}^2 & Cov(\Sigma_A, \Sigma_B) & Cov(\Sigma_A, \Sigma_C) \\ Cov(\Sigma_B, \Sigma_A) & \delta_{\Sigma_B}^2 & Cov(\Sigma_B, \Sigma_C) \\ Cov(\Sigma_C, \Sigma_A) & Cov(\Sigma_C, \Sigma_B) & \delta_{\Sigma_C}^2 \end{bmatrix} \tag{5.3}$$

$$\vec{G} = \begin{bmatrix} \frac{\partial \phi_{Exp}}{\partial \Sigma_A} \\ \frac{\partial \phi_{Exp}}{\partial \Sigma_B} \\ \frac{\partial \phi_{Exp}}{\partial \Sigma_C} \end{bmatrix} \tag{5.4}$$

where δ_i^2 denotes the square of the standard deviation of the variable i and:

$$Cov(X,Y) = \frac{\sum_{i=1}^{N}(x_i - \bar{x})\,(y_i - \bar{y})}{N} = Cov(Y,X) \tag{5.5}$$

If things are in relative terms the respective equations become:

[15] Both optimizations model V as based on the assumption that the experiment is performed with a perfect neutron detector with Poisson statistics. This is because a real measurement methodology has yet to be determined for this type of experiment being performed on the FNS. Once one is determined, future work will be able to more accurately model its uncertainty and potential entanglements between the measurement system and the cross sections being adjusted.

$$\vec{S} = \begin{bmatrix} \frac{\Sigma_{A,0}}{k_{App,0}} \frac{\partial k_{App}}{\partial \Sigma_A} \\ \frac{\Sigma_{B,0}}{k_{App,0}} \frac{\partial k_{App}}{\partial \Sigma_B} \\ \frac{\Sigma_{C,0}}{k_{App,0}} \frac{\partial k_{App}}{\partial \Sigma_C} \end{bmatrix} \tag{5.6}$$

$$M = \begin{bmatrix} \frac{\delta^2_{\Sigma_A}}{\Sigma^2_{A,0}} & \frac{Cov(\Sigma_A, \Sigma_B)}{\Sigma_{A,0}\Sigma_{B,0}} & \frac{Cov(\Sigma_A, \Sigma_C)}{\Sigma_{A,0}\Sigma_{C,0}} \\ \frac{Cov(\Sigma_B, \Sigma_A)}{\Sigma_{B,0}\Sigma_{A,0}} & \frac{\delta^2_{\Sigma_B}}{\Sigma^2_{B,0}} & \frac{Cov(\Sigma_B, \Sigma_C)}{\Sigma_{B,0}\Sigma_{C,0}} \\ \frac{Cov(\Sigma_C, \Sigma_A)}{\Sigma_{C,0}\Sigma_{A,0}} & \frac{Cov(\Sigma_C, \Sigma_B)}{\Sigma_{C,0}\Sigma_{B,0}} & \frac{\delta^2_{\Sigma_C}}{\Sigma^2_{C,0}} \end{bmatrix} \tag{5.7}$$

$$\vec{G} = \begin{bmatrix} \frac{\Sigma_{A,0}}{\phi_{Exp,0}} \frac{\partial \phi_{Exp}}{\partial \Sigma_A} \\ \frac{\Sigma_{B,0}}{\phi_{Exp,0}} \frac{\partial \phi_{Exp}}{\partial \Sigma_B} \\ \frac{\Sigma_{C,0}}{\phi_{Exp,0}} \frac{\partial \phi_{Exp}}{\partial \Sigma_C} \end{bmatrix} \tag{5.8}$$

where a 0 subscript indicates the mean or default value of the associated random variable and $\delta^2_?$ is the variance of the associated random variable. The relative formulation keeps all the terms unitless. The documentation that describes the `KSEN` card for the MCNP 6.1 Manual defines the sensitivity coefficient in this relative fashion [36] and is still referenced in the same way for MCNP 6.3. The data for the covariance matrices in SCALE is also of the relative type. The `PERT` card in MCNP can be used to calculate sensitivities in source-driven calculations [37],[38], [39]. Unfortunately, the MCNP `PERT` just gives an alternative flux tally based on a change in a cells composition (density, ect) for a given reaction and thus needs to be processed into a relative sensitivity using the formulas below.

$$G_i(relative) = \frac{\Sigma_{i,0}}{\phi_{i,0}} \frac{\partial \phi}{\partial \Sigma_i} \tag{5.9}$$

$$G_i(relative) \approx \frac{\frac{\Delta flux\ count}{count_0}}{\frac{\Delta \Sigma}{\Sigma_{i,0}}} = \frac{\frac{\Delta flux\ count}{count_0}}{0.01} \tag{5.10}$$

Alternative densities and compositions were based on a 1% perturbation to stay within a linear regime.

In order to detect individuals that would cause errors if they were run (such as supercritical configurations in source driven calculations), the Bad Vector Determination Step option (see Chapter 3) was used in both optimizations. In the case of these optimizations, the step was used to run criticality calculations of various configurations to ensure that they were (k_{eff} had to be 4 standard deviations below 0.96 because the planned initial NRC license is targeting a k_{eff} under 0.96 and serval thousand evaluations were anticipated) before submitting them for source driven calculations.

5.1: Multi-Objective MCFR Optimization

In the multi-objective optimization only the ^{35}Cl (n,p) cross section (from ENDF/B-VII.1) was considered. The maximum groups available in the data in SCALE was 252, so M is the 252 group covariance matrix for ^{35}Cl (n,p), S is a vector of sensitivities of the k-eigenvalue of the MCFR with respect to each energy group in the ^{35}Cl (n,p) cross section, G is a vector of sensitivities of a FNS flux tally with respect to perturbations in the ^{35}Cl (n,p) cross section in each energy group, V is an approximation of measurement variance and is modeled to be inversely proportional to the measurement time based on assuming poisson statistics (see Equation 5.11).

$$V = \frac{1}{c_0 * 4 * 10^9 * t} \tag{5.11}$$

In Equation 5.11, c_0 is the anticipated number of counts detected in the experimental volume per source neutron and, t is the measurement time in seconds. The measurement time was simulated as an integer value between 1 and 3600 and served as a second objective. A perfectly efficient detector is assumed. For the calculation of the S vector a simplistic model of an MCFR was constructed in MCNP and it is shown in Figure 5.2 below.

Table 5.1 gives the keyword hyperparameters for the Gnowee optimization run. `population` is the number of individuals allowed in each generation, `maxGens` is the maximum number of generations Gnowee is allowed to search, `maxFevals` is the maximum number of objective function evaluations Gnowee is allowed to perform, `fracLevy` controls the probability that an individual has the Lavy flight search heuristic applied to it, `fracMutation` controls the probability of mutations within the mutation search heuristic, `convtol` is the minimum amount of change between generations with improvement that must be observed for the search to be considered unconverged, `stallLimit` is the maximum number of evaluations to be performed without improvement before the search is terminated for stalling.

Figure 5.3 shows the rate of progress in the optimization run. Specifically, it shows the best URR fitness in a generation in each generation in which there is an improvement in the measure. Points in which there is no improvement are excluded because the best URR individual will remain the same. Figure 5.4 shows the final pareto front and demonstrates a non-linear trade-off between reducing data uncertainty and performing the measurement quickly.

An important feature of the final pareto front is that all individuals have the minimum possible number of sodium chloride plates (480). This is likely because the 480 sodium chloride plates provide sufficient sensitivity to the chloride cross section. This would need to be verified through future work.

Several important trends may be seen in Figures 5.5 and 5.6. Figures 5.5 and 5.6 demonstrate that additional polyethylene plates decrease the measurement time but increase the post experiment uncertainty on the flux. The figures also show that the addition of void plates increase the measurement time but reduce the post experiment uncertainty. On the whole additional uranium plates seem to decrease the measurement time and post experiment uncertainty, but the relationship is noisy. The noise in the effect of additional uranium plates may suggest that their effect is highly position- and configuration-dependent.

Additional analysis of the members of the final Pareto Front shows that the members do not have unique performance-with-additional-time characteristics past about 1000 seconds (see Figure 5.7). This suggests that if the time value were fixed at one thousand seconds, a single-objective optimization would arrive at the same or better results. Thus, for the optimization in Section 5.2 there is a single objective function with a time value of 1000.

After this work was performed, it came to the attention of the present researcher that the edition of MCNP that had been used for the optimization (6.2) had a bug related specifically to the use of the Pert card for energy-dependent sensitivities [38], [40]. *The bug caused particles to be scored in the wrong energy bins, thus the original verification process, which had been based on comparing a perturbation across all energies to the results from a perturbation in density, did not detect this bug for this work before it had been performed. The bug had been patched in MCNP 6.2.2, but this version was not installed on the computational cluster at the time the work had been done. Thus, the specific results from this work cannot be trusted.*

Worrying that the single objective assumption was now in doubt for the second optimization, an additional analysis was performed to see if the justification for a single objective optimization still held. As this was performed at a later date, it focused on the newer optimization problem which has a variety of reactions under consideration and was aimed towards an SFR rather than an MCFR. A generation's worth

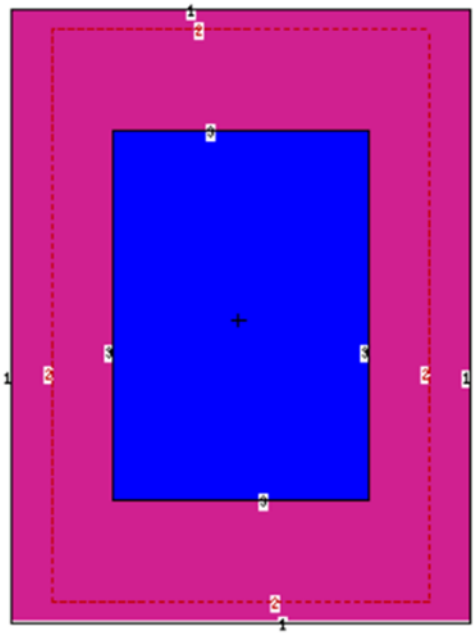


Figure 5.2: MCNP model of MCFR Application

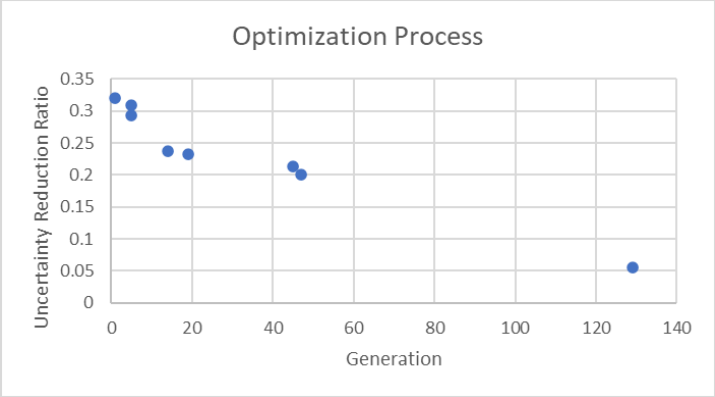


Figure 5.3: URR over the generations. The best URR of any individual in a generation for each generation in which there was an improvement in the best URR performing individual.

Table 5.1: Keyword Hyperparameters

Keyword Hyperparameter	Value
`population`	20
`maxGens`	400
`maxFevals`	10000
`fracLevy`	0.2
`fracMutation`	0.1
`convTol`	1E-6
`stallLimit`	2000

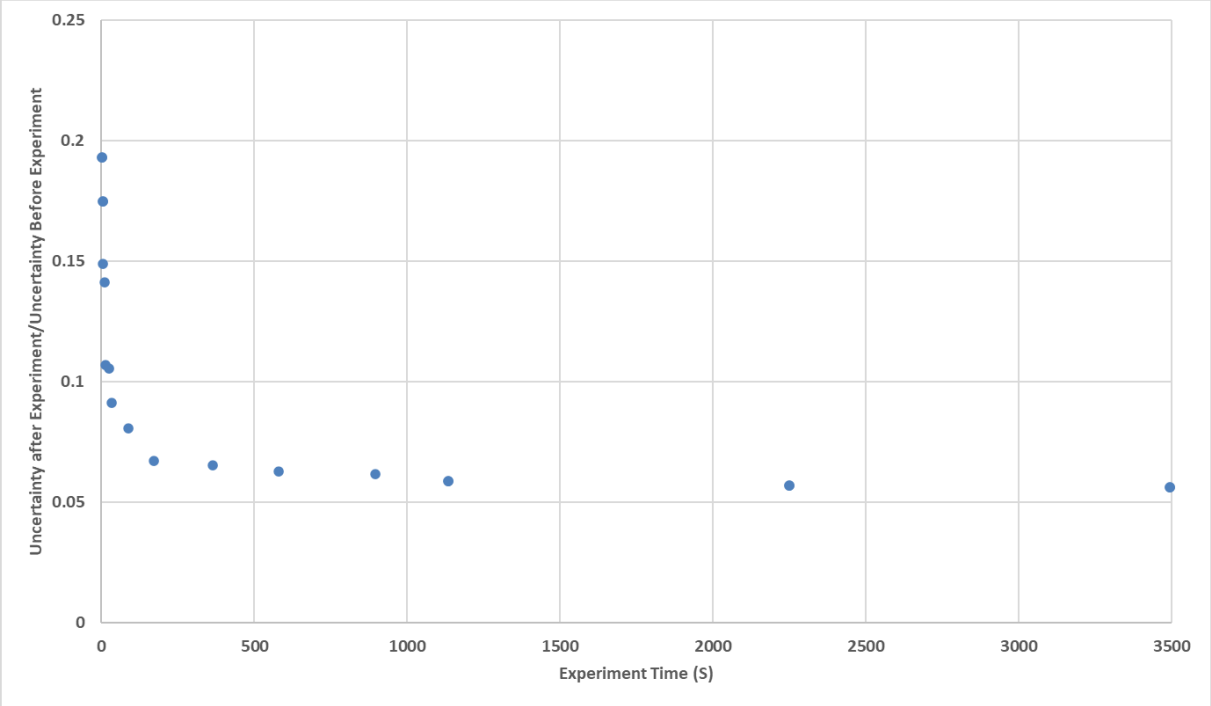


Figure 5.4: The Last Pareto Front

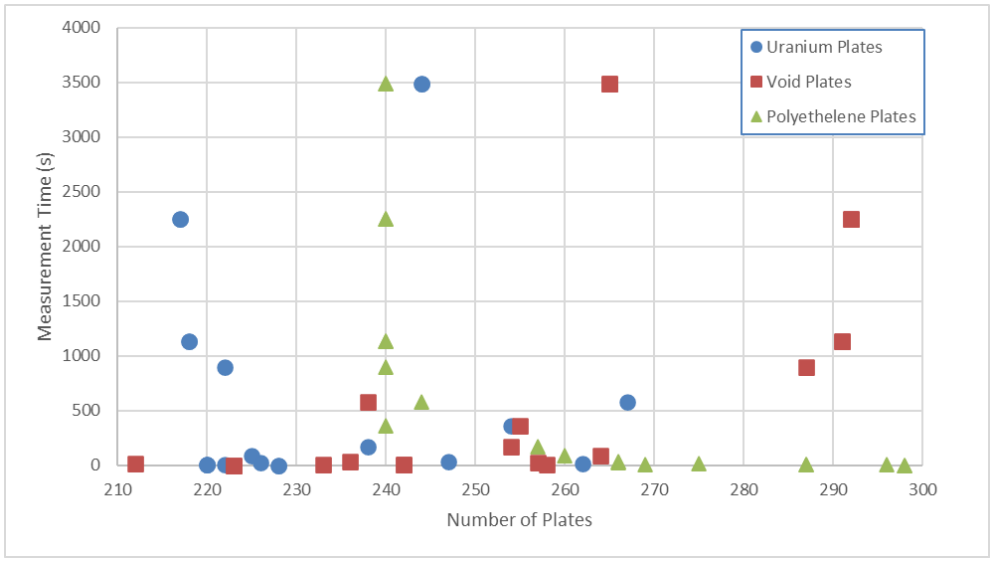


Figure 5.5: Plate Materials and Measurement Time. Effects of the number of various plate material types on measurement time in the final pareto front.

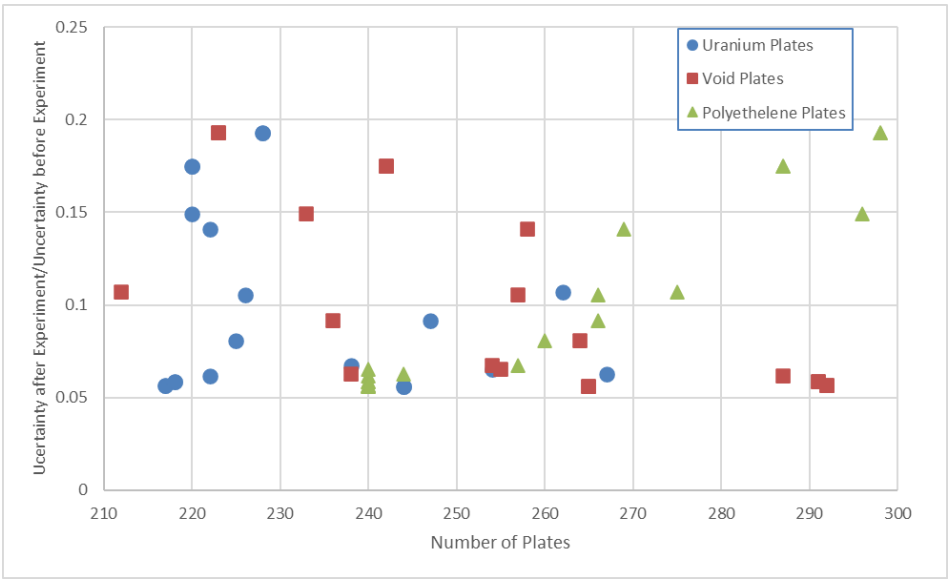


Figure 5.6: Plate Materials and URR. Effects of the number of various plate material types on post experiment uncertainty reduction in the final pareto front.

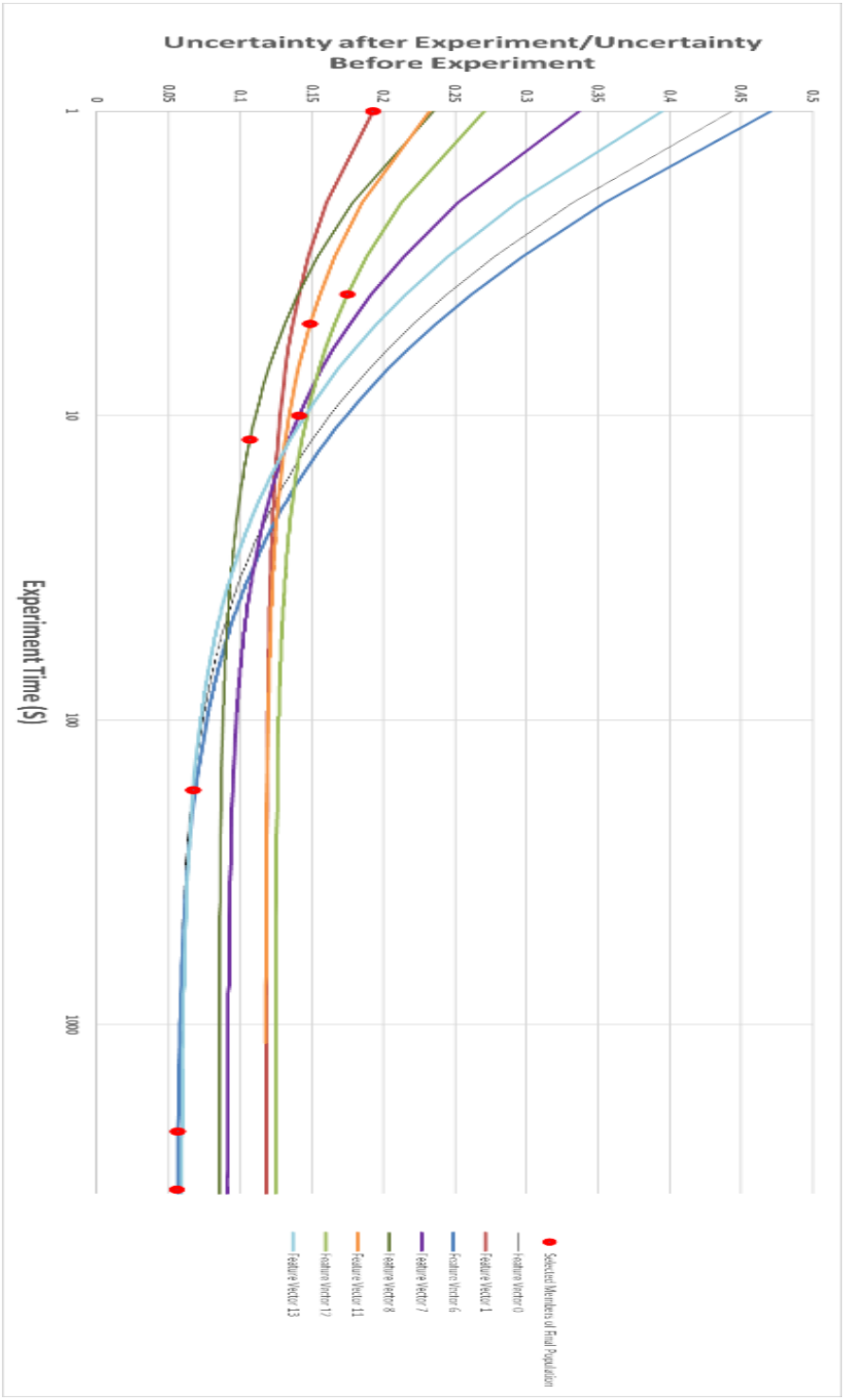


Figure 5.7: The URR of Selected Members of the Final Pareto Front if they had other time values. Results show that the trend with respect to time stops being feature vector-specific past about 1000 seconds. Thus after 1000 seconds the URR rank holds steady.

(20) of feature vectors was generated completely randomly, picking the value for each gene (including time) from a random uniform distribution. The resulting 20 feature vectors were fed to the Bad Vector Determination System, which eliminated 7 of them. The remaining 13 were run and evaluated using the newer optimizations objective function to see if it was still reasonable to simply evaluate all feature vectors at 1000 seconds. The results show that the time drop off effect is still present but is muted.

5.2: Single Objective SFR Optimization

The other optimization was performed for a generic SFR. Accounting for the lessons learned in the previous optimization the time variable was fixed at one thousand seconds and a single objective formulation was adopted.

The optimization also differs from the previous one in the reactions it considers. Unfortunately, the `PERT` card imposes several limitations on the optimization process and reaction selection. The first of which is that each `PERT` card can only determine the sensitivity contributions of a single isotope's reaction in a single material over a single energy range, this means that it cannot account for the sensitivities of isotopes that appear in more then one material. Due to this limitation, the structural material of the MCNP FNS model was changed from carbon steel to pure iron to avoid carbon appearing in multiple materials.

The second limitation is that each `PERT` card increases runtime by between 10% and 20%, this means that the number of `PERT` cards must be kept to a minimum; consequentially sensitivities are calculated for just 56 energy groups and limited subset of all nuclear reactions are considered during the optimization. Enough neutron histories must be run to converge the results of each of the `PERT` cards[16], which further increases run time. Lastly, MCNP only allows a maximum of 1000 unique `PERT` card identifiers, which further forces restrictions on the reactions considered.

The selection of isotopes and reactions to be included in the S and G vectors considered several criteria. Firstly, the isotope had to appear within the plates of the FNS as the sensitivity of the FNS to the isotope had to change significantly from feature vector to feature vector. Second, the isotopes (with the exception of hydrogen) had to also appear within the SFR model. Third, the isotope and reaction had to be present within the 56-group ENDFVII.I covariance data present in SCALE 6.2.3 as well as TSUNAMI-3D sensitivity output data for the SFR. Fourth, `PERT` had to be able to compute the sensitivity to the

[16] The uncertainty on the sensitivities from these cards is not propagated onto the URR equation as the Gnowee algorithm would have no use for this type of information. There exist other types of algorithms that could make use of such information, but they are beyond the scope of the current work.

reaction[17]. To further reduce the number of PERT cards, carbon was dropped from consideration as the SFR was least sensitive to it compared to all the other remaining isotopes. The (n,2n) reactions were dropped from consideration as they were the reactions left that the SFR was least sensitive to. ^{23}Na(n,n') was not considered due to it being a small fraction of the SFR's uncertainties with respect to k_{eff} (as determined by SMS calculations and TSUNAMI) and due to the peculiarities[18] of its treatment by the MCNP PERT card. Lastly, ^{1}H reactions were added based on the judgement that they would be the primary source of moderation in the FNS (and thus potentially affect the GMG term). The series of considerations is laid out in Table 5.2. The following nuclear cross section data was included for the optimization: ^{235}U(fission), ^{235}U(n,γ), ^{235}U (elastic), ^{235}U(n,n'), ^{238}U(fission), ^{238}U (n,γ), ^{238}U (elastic), ^{238}U(n,n'), ^{23}Na (n, γ), ^{23}Na (n,p), ^{23}Na (n,α), ^{23}Na(elastic), ^{1}H(n,γ), and ^{1}H(elastic).

The choice of this particular subset was verified for use with this optimization. The value of the URR was calculated for a randomized feature vector, a feature vector halfway through the optimization process and the final optimized feature vector. First the URR was calculated with just the reactions considered in the optimization, next URR was calculated with all reactions in the final column of Table 5.2 (with the exception of ^{23}Na(n,n')) by splitting the reactions across multiple MCNP files. The comparative rankings of the feature vectors were found to be the same in both URR calculations so it was concluded that the reactions that were left out would not change the overall outcome of the optimization process.

The sensitivities of the application SFR to the same set of reactions were calculated using a TSUNAMI-3D model of a generic SFR. The data for M was from the data found in the *scale.rev08.56groupcov7.1* covariance library file included with distributions of SCALE 6.2.3.

The Gnowee optimization used the keyword hyperparameters given in Table 5.3 and the optimization variables were set as 149 integers between 1 and 4 (representing the 4 different plate material options).

The results show that Gnowee can be used to optimize the FNS. Figure 5.8 shows the rate of progress in the optimization run. Specifically, it shows the best URR fitness in each generation in which there is an improvement in the measure. Points in which there is no improvement are excluded because the best URR will remain the same. As one would expect of the optimization process, the more generations pass, the less frequently improvements are made.

[17] This ruled out non-cross section nuclear data such as Chi and nubar

[18] If one attempts to calculate the sensitivity with the pert card in a straightforward manner, the sensitivity returned will be zero. There was an idea to pursue a calculation based on a summation over various reaction channels but this would have meant that it would not be exactly comparable to the ways in which other sensitivities were calculated and the accuracy/reliability of the method could not be vouched for so the reaction was dropped from consideration.

Table 5.2: Isotopes and Reactions of Interest

Isotopes Present in FNS Plates	Isotopes Present in Both Plates and SFR	SFR SMS Due to Isotope and Rank	Reactions Present in SFR Sensitivity Data and ENDFVII.1 Covariance Data
^{1}H	-	-	(n,γ), elastic
C	C	2.159×10^{-12} (4^{th})	(n,γ), (n,p), (n,d), (n,α), elastic, (n,n')
^{232}U	-	-	-
^{234}U	-	-	-
^{235}U	^{235}U	7.316×10^{-4}(1^{st})	Fission, (n,γ), elastic, (n,n'), (n,2n)
^{236}U	-	-	-
^{238}U	^{238}U	1.441×10^{-5} (2^{nd})	Fission, (n,γ), elastic, (n,n'), (n,2n)
^{23}Na	^{23}Na	8.335×10^{-8} (3^{rd})	(n,γ), (n,p), (n,α), elastic, (n,n'), (n,2n)

The optimization concluded when it reached its preset maximum number of requests for evaluations. The best performing individual (shown in Figure 5.9[19]) in the run was found on the 2003th evaluation, this was allowed to happen after the 2000th because the convergence criteria are only checked at the end of a generation loop. Visual analysis of the results suggests that further improvements in fitness would have been marginal. The optimization took approximately 197 hours while making maximum use of a set of processors on a computational cluster[20].

The final design has reduced the number of sodium plates almost to the minimum allowed and uses a combination of mostly fuel plates with empty space along with the occasional polyethylene plate. Examining the best performers in generations with improvement in Figure 5.10 helps explain this result. Figure 5.10 compares $\frac{(S^T MG)^2}{(S^T MS)(G^T MG)}$ to $\frac{V}{G^T MG}$ to determine what phenomena drive the optimization. $\frac{(S^T MG)^2}{(S^T MS)(G^T MG)}$ is conceptually the relative amount of uncertainty due to the cross section covariance matrix M that the experiment and application have in common with each other. $\frac{V}{G^T MG}$ is a term relating to the amount of uncertainty in the experiment's measurement. It is clear that the experiment uncertainty term does not improve much compared to the improvements made in the shared uncertainty term, and thus it is an increasing alignment between the experiment and application that drives the optimization. The magnitude of the experiment's measurement uncertainty is likely unrealistically low because Equation 5.11 is modelling a perfect detector whose uncertainty can be made arbitrarily small with enough measurement time. This is a gross oversimplification that (as the conclusion states) future work should improve upon. Figure 5.11 shows that overall designs evaluated there is a strong correlation between the shared uncertainty term and the number of fuel plates a design has.

The reason for the strong relationship between the shared uncertainty term and the number of fuel plates is that the prior uncertainty on the application ($S^T MS$) is due almost entirely (~98%) to reactions on the ^{235}U and ^{238}U isotopes which are only present in the fuel plates. Neutrons that are absorbed in sodium or polyethylene are neutrons that might have otherwise been absorbed in uranium (and thus increased

[19] Note that the empty spaces in Figure 5.9 will in reality contain air, the neutronic influence of which is neglected in the present work.

[20] Unfortunately the program had to be run on a departmental cluster while given low priority in a queueing system which assigned it to whatever CPUs and cores became available. This means that the particular number of cores it was run on, given access to, or waiting for, was not consistent. An auxiliary system was used to record when each MCNP file finished its computation, but without information on the exact cores the program was running on and level of parallelization at a given time, the information is not useful aside from making it evident that the optimization was time-constrained.

Table 5.3: Keyword Hyperparameters Used for SFR Gnowee Optimization

Keyword Hyperparameter	Value
`population`	20
`maxGens`	200
`maxFevals`	2000
`fracLevy`	0.2
`fracMutation`	0.1
`convTol`	1E-6
`stallLimit`	2000

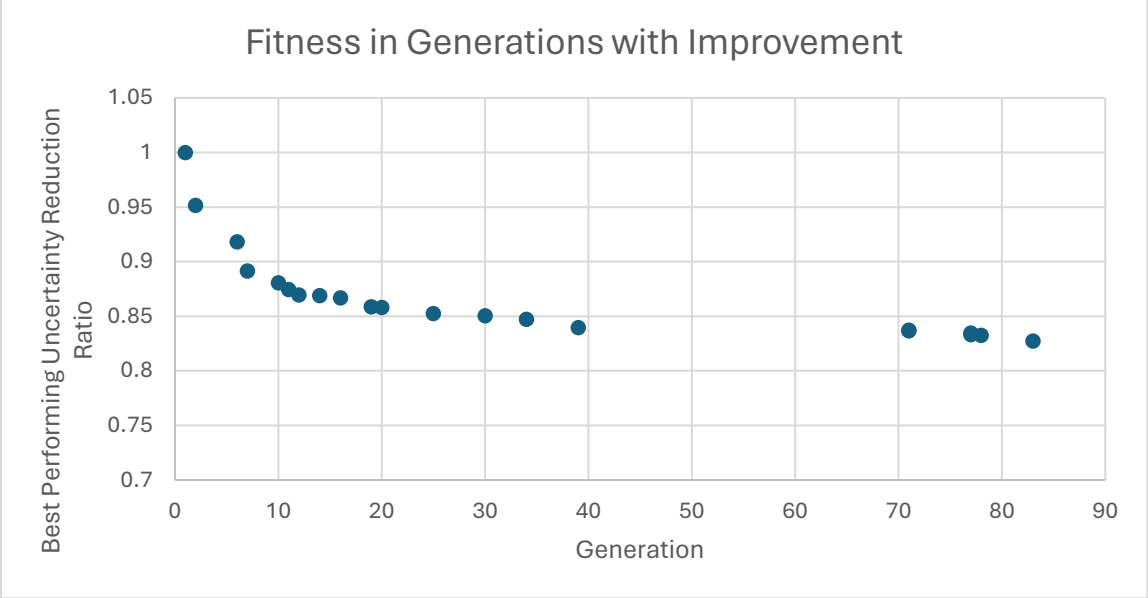


Figure 5.8: URR over the generations. The best URR of any individual in a generation for each generation in which there was an improvement in the best URR performing individual.

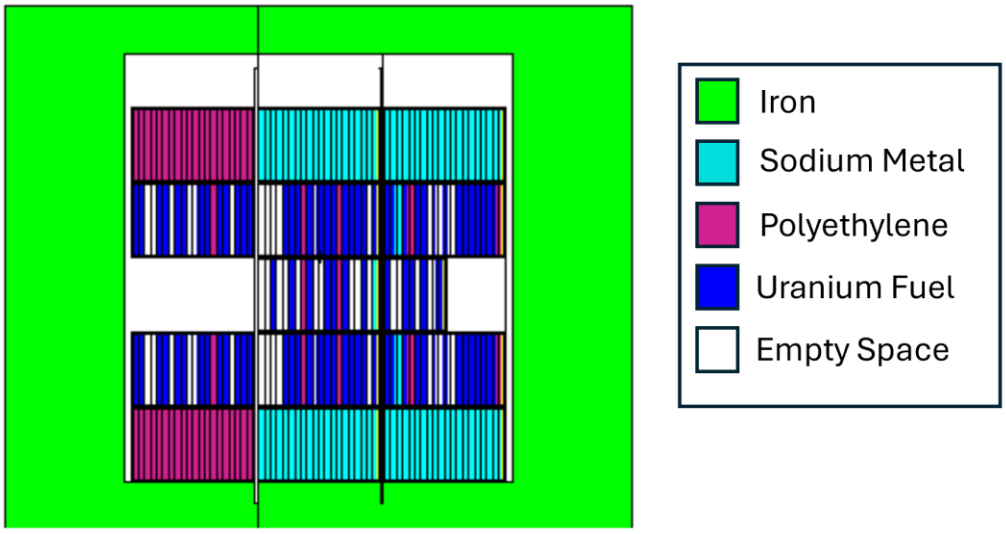


Figure 5.9: The Optimal Design. XZ View of the MCNP Model of the Final Best Performing Individual. Note that some regions are specified by symmetry constraints and assumptions.

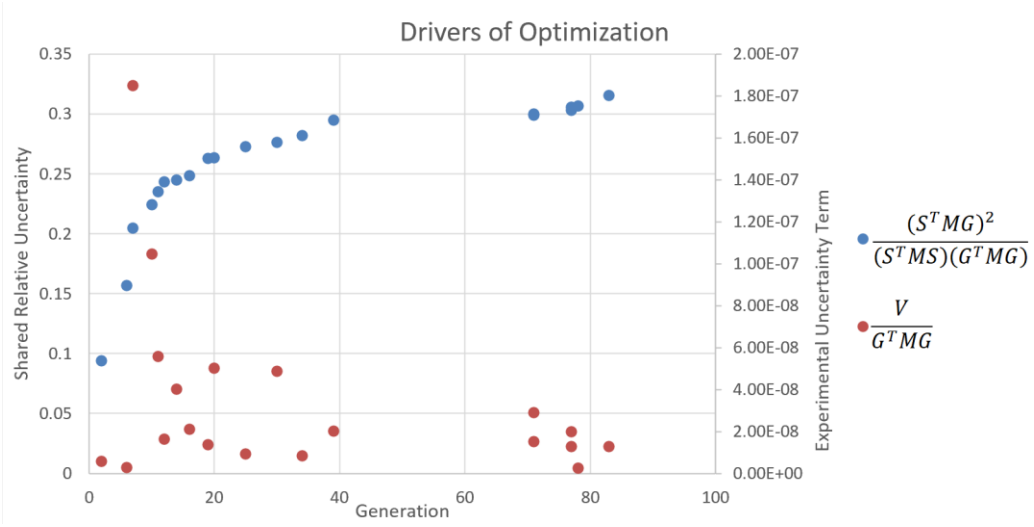


Figure 5.10: The Drivers of the Optimization. It is clear that the numerator term is the primary part of the URR equation where improvement is made.

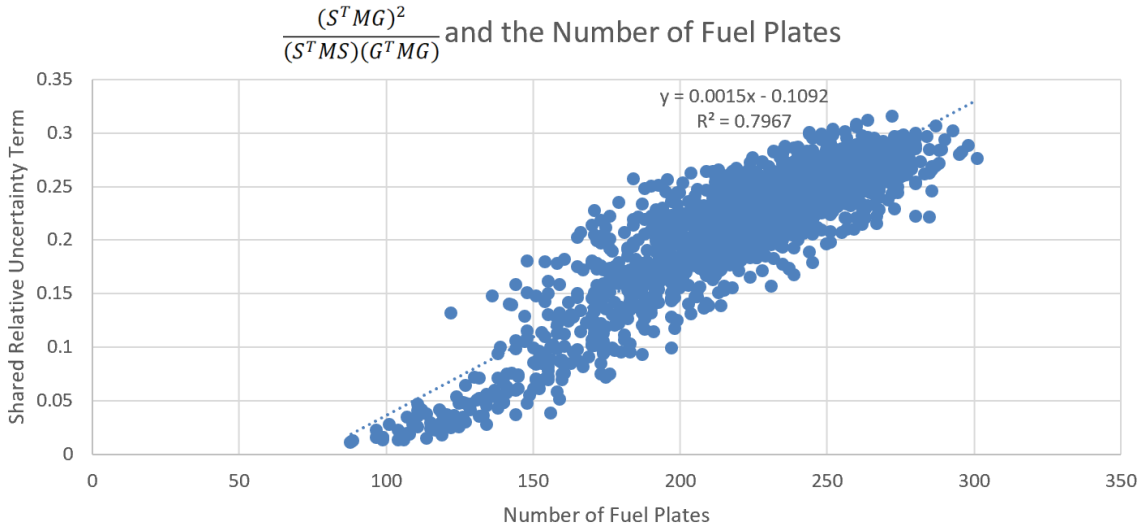


Figure 5.11: Shared Uncertainty Term and the Number of Fuel Plates

sensitivity to uranium). The reason the optimized design was not all fuel plates was because of the criticality constraint requiring designs to have a value of k well below 0.96.

5.3: A Point of Comparison

The URR formula is a new objective function whose application is an important part of the present thesis. It would be pointless for the present thesis however, if an optimization using an objective function that is a proxy for the URR could outperform an optimization using the URR as an objective function in terms of minimizing the URR. Thus there is a need for a rough comparison to show that the present methods are an improvement over those used in the past.

One readily available point of comparison is the set of solutions generated for the paper "Neural Network Acceleration of Genetic Algorithms for the Optimization of a Coupled Fast/Thermal Nuclear Experiment" [26]. That paper used representativity (the RF given by Equation 2.1) as one of its main objective functions. The best representativity from the convolutional neural network-surrogate NSGA-II optimization was found in Table 4 of the paper. This was cross referenced with an Excel file provided by Dr. Pevey to find the file name of the associated MCNP input deck, the file was then located on the University of Tennessee Knoxville Department of Nuclear Engineering computational cluster, hand edited to match the densities and constraints of the newer models of the FNS, and then analyzed with MCNP to find a corresponding URR value of 0.98. This value compares disfavorably with the best performing individual in the preceding (section 5.2) optimization which has a URR value of 0.83.

The inverse calculation was also performed. The representativity for the best performing individual from section 5.2 was 0.940 which was a "worse" value for the representativity than the best performing individual in Dr. Pevey's paper (0.995). This suggests that the representativity metric is an imperfect proxy and can in fact lead an optimization astray. However it should be kept in mind that this is not an apples to apples comparison due to differences in geometry and materials.

Chapter 6: Conclusions

This chapter[21] contains the main conclusions of the author and the prospects for future work.

6.1: Contributions

There are several major contributions made by this thesis work. The capabilities and usability of Gnowee have been improved by adding parallelization and other improvements. A procedure for optimizing integral experiments for any particular specified application has been developed, verified and applied to multiple cross sections and reactions. The method for predicting the uncertainty reduction on a parameter of interest from an integral experiment has been verified. Thus, integral experiment designers have two new tools at their disposal: an improved Gnowee, and a formula for predicting the value of their experiment to various applications.

6.2: Future Work

There are shortcomings in the present work and future work should be able to address them.

The first major thing to expand upon is the model of the experimental uncertainty V (V in Equation 5.11). The current model is naïve with respect to the effect of measurement time and nature of particle detection. One way to improve it would be based on more explicitly modelling the detector in MCNP. Another would be to better capture the effect of systematic biases.

A second area for future work is the FNS optimization process. Future work could allow more freedom in the selection of plate materials by altering the symmetry conditions (shown in Figure 5.1) and including the outer ring of cassette positions in the optimization. The location of the detector could also be made subject to optimization. Alternative objective functions could be used, such as cost, alongside the URR in multi-objective optimizations.

A third area for future work concerns the `PERT` card. This card imposed severe limitations on the present work, but future versions of MCNP will have an alternative called the `FSEN` card which will not be restricted to isotopes that only appear in a single material [41],[42].

Another area for future work is a multi-measurement extension of the URR formula. If there are two measurements taken of the experiment system the equivalent of Equations 5.4 would be:

[21] Portions of this chapter are from previous work [1], [2],[3],[4], by Alexander DePillis, Vladimir Sobes, J. Wesley Hines, and Sandra Bogetic

$$G = \begin{bmatrix} \frac{\partial \phi_{Exp,1}}{\partial \Sigma_A} & \frac{\partial \phi_{Exp,2}}{\partial \Sigma_A} \\ \frac{\partial \phi_{Exp,1}}{\partial \Sigma_B} & \frac{\partial \phi_{Exp,2}}{\partial \Sigma_B} \\ \frac{\partial \phi_{Exp,1}}{\partial \Sigma_C} & \frac{\partial \phi_{Exp,2}}{\partial \Sigma_C} \end{bmatrix} \tag{6.1}$$

Where $\phi_{Exp,1}$ and $\phi_{Exp,2}$ are the two different measurements of the experiment system. V (the uncertainty associated with the experimental measurement) would change from a scalar to a matrix given by (in the two measurement case):

$$V = \begin{bmatrix} Var(\phi_{Exp,1}) & Cov(\phi_{Exp,1}, \phi_{Exp,2}) \\ Cov(\phi_{Exp,1}, \phi_{Exp,2}) & Var(\phi_{Exp,2}) \end{bmatrix} \tag{6.2}$$

Where Var() and Cov() mean the variance or covariance (respectively) of the associated variables. Because of the changed dimensionality of G and V, the derivation of the URR formula has to be slightly altered and this results the form below for the URR formula,

$$\frac{\delta_a'}{\delta_a} = \sqrt{1 - \frac{S^T MG(G^T MG + V)^{-1} G^T MS}{S^T MS}} \tag{6.3}$$

Given that an extension can be made for multiple experimental measurements, one might also ask for an extension that will handle multiple parameters of interest (AKA more then just k_{eff}). If there are multiple parameters in the application system that one wishes to update simultaneously, then S (the vector of sensitivities of the application's parameter of interest to the nuclear data) must become a matrix of the form (if there are two parameters of interest):

$$S = \begin{bmatrix} \frac{\partial parameter_1}{\partial \Sigma_A} & \frac{\partial parameter_2}{\partial \Sigma_A} \\ \frac{\partial parameter_1}{\partial \Sigma_B} & \frac{\partial parameter_2}{\partial \Sigma_B} \\ \frac{\partial parameter_1}{\partial \Sigma_C} & \frac{\partial parameter_2}{\partial \Sigma_C} \end{bmatrix} \tag{6.4}$$

If there are multiple parameters one is trying to reduce the uncertainty on, a simple scalar ratio can no longer be constructed to represent the ratio of **the** uncertainty on the application after over **the** uncertainty on the application before the measurement. The uncertainty on the application before the measurement can still be represented as $S^T MS$, but $S^T MS$ is no longer a scalar so it can't serve as a denominator. Instead, it takes the form:

$$\delta_a^2 = S^T MS = \begin{bmatrix} Var(parameter_1) & Cov(parameter_1, parameter_2) \\ Cov(parameter_1, parameter_2) & Var(parameter_2) \end{bmatrix} \quad (6.5)$$

Similarly, the uncertainty after the experimental measurement for the parameters of the application system is represented by:

$$\delta_a'^2 = S^T[M']S = \begin{bmatrix} Var(parameter_1') & Cov(parameter_1', parameter_2') \\ Cov(parameter_1', parameter_2') & Var(parameter_2') \end{bmatrix} \quad (6.5)$$

Where M' is the updated nuclear data covariance matrix $parameter_1'$ and, $parameter_2'$ are the updated parameters of interest for the application system. Because these terms are now matrices, the URR formula is forced to take the form:

$$\delta_a'^2 = \delta_a^2 - S^T MG(G^T MG + V)^{-1} G^T MS \quad (6.6)$$

Lastly, future work could improve on Gnowee_multi. In fact there is already another edition of the program on the way provisionally named MultiGnowee. This new edition will have a variety of new features and improvements.

References

[1] A. DePillis, V. Sobes, S. Bogetic, and J. W. Hines, “Genetic Algorithm Optimization of Experiment Design for Targeted Uncertainty Reduction on the Cl-35 (n,p) Cross Section,” in *Transactions of the American Nuclear Society*, Phoenix, AZ: American Nuclear Society, Nov. 2022, pp. 700–703. doi: doi.org/10.13182/T127-39894.

[2] A. A. DePillis, S. Bogetic, and V. Sobes, “Prediction of and Genetic Algorithm Optimization on Data Induced Uncertainty Reduction with the Use of an Integral Experiment.” Rochester, NY, Mar. 05, 2024. doi: 10.2139/ssrn.4748215.

[3] A. A. DePillis, “Autonomous Design of Nuclear Measurement Systems,” presented at the University Program Review 2022, University of Michigan, Ann Arbor, MI 48109, Jun. 08, 2022.

[4] A. A. DePillis, “AI-Based Optimization of Nuclear Measurement Systems,” presented at the University Program Review 2023, University of California, Berkeley, CA 94750, Jun. 06, 2023.

[5] S. Quaglioni and P. Navrátil, “Nuclear Resonances, Scattering, and Reactions from First Principles: Progress and Prospects,” *Nuclear Physics News*, vol. 30, no. 4, pp. 12–16, Oct. 2020, doi: 10.1080/10619127.2020.1752089.

[6] S. Binder, J. Langhammer, A. Calci, and R. Roth, “*Ab initio* path to heavy nuclei,” *Physics Letters B*, vol. 736, pp. 119–123, Sep. 2014, doi: 10.1016/j.physletb.2014.07.010.

[7] Talou, Patrick, “Evaluating nuclear data and their uncertainties,” *EPJ Nuclear Sci. Technol.*, vol. 4, p. 29, 2018, doi: 10.1051/epjn/2018032.

[8] H. KUROI and H. MITANI, “Adjustment to Cross Section Data to fit Integral Experiments by Least Squares Method,” *Journal of Nuclear Science and Technology*, vol. 12, no. 11, pp. 663–680, Nov. 1975, doi: 10.1080/18811248.1975.9733172.

[9] I. Michaud, M. Grosskopf, J. Hutchinson, and S. Vander Wiel, “Expert-in-the-loop design of integral nuclear data experiments,” *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 17, no. 2, p. e11677, 2024, doi: https://doi.org/10.1002/sam.11677.

[10] N. W. Touran and J. Yang, “Sensitivities and Uncertainties Due to Nuclear Data in a Traveling Wave Reactor,” presented at the PHYSOR, Sun Valley, Idaho: American Nuclear Society, May 2016. [Online]. Available: https://www.researchgate.net/publication/376233001_Sensitivities_and_Uncertainties_due_to_Nuclear_Data_in_a_Traveling_Wave_Reactor

[11] V. Sobes, J. Pevey, A. Depillis, O. Chvala, S. Bogetic, and W. Hines, “The fast neutron source at UTK: a project with Massimo Salvatores,” United States: ANS - American Nuclear Society, 2022. doi: doi.org/10.13182/PHYSOR22-37654.

[12] J. L. Pevey, C. Salyer, O. Chvala, V. Sobes, and J. W. Hines, “Multi-objective design optimization of a fast spectrum nuclear experiment facility using artificial intelligence,” *Annals of Nuclear Energy*, vol. 162, p. 108476, Nov. 2021, doi: 10.1016/j.anucene.2021.108476.

[13] Brezinski Bernadette, *FNSBoxOut.JPG*. 2023.

[14] J. E. Bevins and R. N. Slaybaugh, “Gnowee: A Hybrid Metaheuristic Optimization Algorithm for Constrained, Black Box, Combinatorial Mixed-Integer Design,” *Nuclear Technology*, vol. 205, no. 4, pp. 542–562, Apr. 2019, doi: 10.1080/00295450.2018.1496692.

[15] J. D. Hutchinson *et al.*, “Sensitivity Studies, Gap Analysis, and Benchmark Experiment Optimization for Reactor Physics and Criticality Safety Applications,” *Transactions of the American Nuclear Society*, vol. 123, no. 1, Art. no. LA-UR-20-29405; LA-UR-20-24750, Nov. 2020, doi: 10.13182/t123-32959.

[16] Hutchinson, J. *et al.*, “EUCLID: A New Approach to Constrain Nuclear Data via Optimized Validation Experiments using Machine Learning,” *EPJ Web of Conf.*, vol. 284, p. 15006, 2023, doi: 10.1051/epjconf/202328415006.

[17] I. Michaud *et al.*, “Designing Critical Experiments Using Gaussian Process Optimization,” *Transactions of the American Nuclear Society*, vol. 121, no. 1, Nov. 2019, doi: 10.13182/t31031.
[18] M. Binois and N. Wycoff, “A Survey on High-dimensional Gaussian Process Modeling with Application to Bayesian Optimization,” *ACM Trans. Evol. Learn. Optim.*, vol. 2, no. 2. Association for Computing Machinery, p. Article 8, 2022.
[19] S. Bogetic, “Improvements, Validation, and Applications of a Metaheuristic Optimization Method for Neutron Spectra Tailoring at the National Ignition Facility.,” United States, 2020. doi: 10.2172/1631912.
[20] “ADVANTG 3.2.1, AutomateD VAriaNce reducTion Generator without ORNL-TN.” Accessed: Mar. 19, 2024. [Online]. Available: https://www.oecd-nea.org/tools/abstract/detail/ccc-0854/
[21] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr. 2002, doi: 10.1109/4235.996017.
[22] W. Hines, J. Pevey, and V. Sobes, “Preliminary Design of a Fast Flux User Facility at the University of Tennessee,” in *Transactions of the American Nuclear Society*, Orlando, Florida: Oak Ridge National Lab. (ORNL), Oak Ridge, TN (United States), Nov. 2018, pp. 1261–1264. Accessed: Mar. 20, 2024. [Online]. Available: https://www.osti.gov/biblio/1486919
[23] J. Pevey, O. Chvála, S. Davis, V. Sobes, and J. W. Hines, “Genetic Algorithm Design of a Coupled Fast and Thermal Subcritical Assembly,” *Nuclear Technology*, vol. 206, no. 4, pp. 609–619, Apr. 2020, doi: 10.1080/00295450.2019.1664198.
[24] J. Pevey, O. Chvala, S. Davis, V. Sobes, and J. Hines, “Current Progress on the Design of a Coupled Fast-Thermal Subcritical Assembly,” in *Transactions of the American Nuclear Society*, Washington, D.C.: American Nuclear Society, Jan. 2019, pp. 1536–1538. doi: 10.13182/T30927.
[25] O. Chvala, J. Hines, J. Pevey, and V. Sobes, “University of Tennessee Fast Neutron Source Design History,” in *Transactions of the American Nuclear Society*, Online Virtual Meeting: American Nuclear Society, Jun. 2020, pp. 760–763. doi: 10.13182/T32430.
[26] J. Pevey, V. Sobes, and Wes. J. Hines, “Neural Network Acceleration of Genetic Algorithms for the Optimization of a Coupled Fast/Thermal Nuclear Experiment,” *Frontiers in Energy Research*, vol. 10, 2022, doi: 10.3389/fenrg.2022.874194.
[27] V. Sobes, J. Pevey, A. Depillis, O. Chvala, S. Bogetic, and W. Hines, “The Fast Neutron Source at UTK: A Project with Massimo Salvatores,” presented at the International Conference on Physics of Reactors 2022 (PHYSOR 2022), Pittsburgh: ANS - American Nuclear Society, Jan. 2022, pp. 1068–1074. doi: 10.13182/PHYSOR22-37654.
[28] J. Pevey, “Multi-objective Optimization of the Fast Neutron Source by Machine Learning,” PhD Dissertation, University of Tennessee, Knoxville, 2022. [Online]. Available: https://trace.tennessee.edu/utk_graddiss/7734
[29] J. E. Bevins and R. N. Slaybaugh, “Gnowee,” Github. Accessed: Mar. 13, 2024. [Online]. Available: https://github.com/SlaybaughLab/Gnowee?tab=readme-ov-file#readme
[30] J. Bevins, “Targeted Modification of Neutron Energy Spectra for National Security Applications,” UC Berkeley, 2017. Accessed: Mar. 13, 2024. [Online]. Available: https://escholarship.org/uc/item/487109jp
[31] “Sunsetting Python 2,” Python.org. Accessed: Mar. 03, 2024. [Online]. Available: https://www.python.org/doc/sunset-python-2/
[32] “2to3 — Automated Python 2 to 3 code translation,” Python documentation. Accessed: Mar. 03, 2024. [Online]. Available: https://docs.python.org/3/library/2to3.html
[33] J. Abonyi, Á. Ipkovich, G. Dörgő, and K. Héberger, “Matrix factorization-based multi-objective ranking-What makes a good university?,” *PloS one*, vol. 18, p. e0284078, Apr. 2023, doi: 10.1371/journal.pone.0284078.
[34] Pham Ngo Gia Bao, Tram Loi Quan, Quan Thanh Tho, and Akhil Garg, “NSGA-II.” Jan. 29, 2024. Accessed: Feb. 07, 2024. [Online]. Available: https://github.com/baopng/NSGA-II
[35] V. Sobes, “FNS_Design_in_Four_Parts.pdf.” Dec. 15, 2021.

[36] B. C. Kiedrowski, “MCNP6.1 k-Eigenvalue Sensitivity Capability: A User’s Guide,” in *Proceedings of 2012 ANS Winter Meeting*, San Diego: American Nuclear Society, Mar. 2013. doi: 10.2172/1072231.
[37] J. R. Lamproe, T. E. Cutler, M. Y. Hua, J. D. Hutchinson, and S. A. Pozzi, “Validation of Nuclear Data Sensitivity Calculations by the MCNP PERT Card [Poster],” Mar. 2021, doi: 10.2172/1906290.
[38] J. R. Lamproe *et al.*, “A Verification of Flux Sensitivity Estimates Using the MCNP Tally Perturbation Tool,” in *Transactions of the American Nuclear Society*, Washington, D.C.: ANS - American Nuclear Society, Dec. 2021. doi: 10.13182/T125-36921.
[39] J. R. Lamproe *et al.*, “Preliminary verification of the MCNP perturbation and fixed-source tally sensitivity tools,” *Annals of Nuclear Energy*, vol. 194, p. 110040, Dec. 2023, doi: 10.1016/j.anucene.2023.110040.
[40] J. A. Favorite and D. K. Parsons, “(U) Energy-Dependent Perturbations of Scattering Cross Sections Using the PERT Card in MCNP6.2,” Los Alamos National Laboratory (LANL), Los Alamos, NM (United States), LA-UR-20-24082, Jun. 2020. doi: 10.2172/1632669.
[41] M. E. Rising and A. R. Clark, “Development of a New Fixed-source Sensitivity Tally Capability in the MCNP® Code [Slides],” in *Report Number: LA-UR-22-27323*, Research Org.: Los Alamos National Lab. (LANL), Los Alamos, NM (United States), Jul. 2022. doi: 10.2172/1898128.
[42] M. E. Rising and A. R. Clark, “Development of a New Fixed-source Sensitivity Tally Capability in the MCNP® code,” *Nuclear Data*, Jul. 2022, [Online]. Available: https://www.osti.gov/biblio/1878090

Vita

Alexander Amedeo DePillis was born on May 25th 1996 to loving parents Anne Walton and Mario DePillis Jr. He grew up in Amherst, Massachusetts and graduated in 2014. He attended Beloit College and Rensselaer Polytechnic Institute, graduating from both simultaneously with two Bachelor's degrees one a B.A. in Physics (with a minor in Chemistry) and the other a B.S. in Nuclear Engineering.