UNIVERSIDAD NACIONAL AUTÓMOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA

ENERGÍA – SISTEMAS ENERGÉTICOS

FUEL LOADING PATTERN OPTIMIZATION IN THE ALLEGRO GAS-COOLED FAST REACTOR THROUGH METAHEURISTICS

TESIS

QUE PARA OPTAR POR EL GRADO DE:

DOCTOR EN INGENIERÍA

PRESENTA:

YROBEL LIMA REINALDO

TUTOR PRINCIPAL:

DR. JUAN LUIS FRANÇOIS LACOUTURE

FACULTAD DE INGENIERÍA - UNAM

COMITÉ TUTOR:

DR. JOSÉ ALEJANDRO CASTILLO MÉNDEZ, INSTITUTO NACIONAL DE INVESTIGACIONES NUCLEARES

DR. JUAN JOSÉ ORTIZ SERVIN, INSTITUTO NACIONAL DE INVESTIGACIONES NUCLEARES

DRA. CECILIA MARTÍN DEL CAMPO MÁRQUEZ, FACULTAD DE INGENIERÍA – UNAM

DR. ROBERTO CARLOS LÓPEZ SOLIS, INSTITUTO NACIONAL DE INVESTIGACIONES NUCLEARES

CIUDAD DE MÉXICO, JUNIO DE 2023

JURADO ASIGNADO:

Presidente:	Dra. Martín del Campo Márquez Cecilia
Secretario:	Dr. Castillo Méndez José Alejandro
1er. Vocal:	Dr. François Lacouture Juan Luis
2do. Vocal:	Dr. Ortiz Servin Juan José
3er. Vocal:	Dr. López Solis Roberto Carlos

Lugar donde se realizó la tesis: FACULTAD DE INGENIERÍA - UNAM

TUTOR DE TESIS:

DR. JUAN LUIS FRANÇOIS LACOUTURE

[illegible]

FIRMA

Dedication

A mi abuela y mi abuelo…

Mi abuelo, siempre en mi memoria.

Acknowledgements

My heartfelt gratitude to my beloved wife for her unconditional support, motivation, encouragement, and companionship throughout all these years. This achievement would not have been possible without her.

I wish to extend my deepest gratitude to my thesis director, Juan Luis. This research would not have been possible without his full support, expertise, and advice. I will be eternally grateful for his commitment, time, kindness, understanding and for being a constant source of strength and inspiration throughout all these years.

To my family and friends. My wife, my grandparents, my siblings, my nieces and nephews, my parents, my uncles, my cousins, and my wife's parents. Thank you for your support, encouragement, and motivation throughout all these years. This achievement goes for you.

My deepest gratitude to Dr. José Alejandro for his full support, expertise, time, collaboration, and helpful advice during the implementation of the tabu search optimization technique. It was a great learning experience working with him.

I want to express my deepest gratitude to all members of the thesis committee, Dr. José Alejandro, Dr. Juan José, Dr. Cecilia, and Dr. Roberto Carlos for their supervision and advice throughout the research. Their recommendations and feedback on each report have been invaluable in improving and completing this dissertation.

To the National Council of Science and Technology (CONACYT) for providing the economic support to conduct this research.

My deepest gratitude to the National Autonomous University of Mexico (UNAM), especially to the School of Engineering, for the opportunity to learn and grow in this prestigious institution. The exceptional academic experience during these years has contributed significantly to my personal and professional development.

To Mexico.

Abstract

Metaheuristic optimization techniques have been used extensively in the fuel reload design of light water and heavy water reactors. However, to date, there are no studies conducted on the implementation of these methods to fuel loading pattern optimization in advanced fourth-generation gas-cooled fast reactors. The main reason for the lack of studies on this topic could be that it is a developing concept, and therefore research has focused on neutronics and materials science.

This doctoral dissertation focuses on the development of a computer code based on metaheuristics techniques to optimize the fuel loading pattern of the ALLEGRO fast reactor. The early stage covers the modeling of the reactor core. Firstly, a three-dimensional heterogeneous model is set up using the Monte Carlo Serpent reactor physics code. Since there are no available studies assuming advanced ceramic fuel configuration, this model in Serpent is adopted as a benchmark. Subsequently, the deterministic ERANOS reactor physics code is used as an alternative to reduce the computational cost of core calculations. It is well known that in optimization calculations, a significant number of evaluations of a given objective function are required, so reducing the execution time of the simulation code is crucial to find an optimal solution in a reasonable computation time. Several case studies are investigated using different calculation options and energy groups structure. Among the core parameters calculated at the beginning of the cycle are the effective neutron multiplication factor, the neutron spectrum, the neutron flux and power distributions, the Doppler constant, the effect of helium density on reactivity and the effective delayed neutron fraction. The evolution of the main fuel isotopes and the k-eff value, over the operating time, are also analyzed. A fuel cycle study is carried out in ERANOS by using several built-in modules for in-core fuel management. The methodology applied to determine the core equilibrium conditions is presented and the results for the proposed reloading and reshuffling scheme are discussed.

In the next stage, the genetic algorithm and tabu search techniques are applied to the fuel loading pattern optimization in the ALLEGRO reactor core. The objective is to maximize the k-eff value at the end of the cycle by satisfying the constraints on the power peaking factor over the cycle, the excess reactivity at the beginning of the cycle, and the linear heat generation rate. The basic methodology used to program the interface between the ERANOS code and the optimizer to compute the objective function is presented. The practical implementation of each technique is discussed. Among the highlights of the programmed optimization code based on genetic algorithms is the incorporation of the partially mapped crossover and order crossover operators, commonly used for permutation-based problems. Other code improvements that reduce running time are also presented. An improved version of the tabu search algorithm is also successfully implemented. This technique was selected to overcome some of the inherent shortcomings of the previous method. By applying the concepts of variable tabu list and aspiration criterion, cycling was avoided, resulting in search diversification. The technique proves to be a powerful tool compared to genetic algorithms, where it is common to get trapped in local optima.

Resumen

Las técnicas metaheurísticas de optimización han sido ampliamente utilizadas en el diseño de recargas de combustible en reactores de agua ligera y reactores de agua pesada. Sin embargo, hasta la fecha, no existen estudios sobre la aplicación de estos métodos en la optimización de patrones de recarga en reactores rápidos enfriados por gas de cuarta generación. La principal razón de la falta de estudios sobre este tema podría ser que se trata de un concepto en desarrollo y, por tanto, la investigación se ha centrado en la neutrónica y la ciencia de los materiales.

Esta tesis doctoral se enfoca en el desarrollo de un código basado en técnicas metaheurísticas para optimizar la recarga de combustible en el reactor rápido ALLEGRO. La etapa inicial de la investigación abarca el modelado del núcleo del reactor. Primero, se establece un modelo tridimensional heterogéneo usando el código de simulación Monte Carlo Serpent. Debido a que no existen estudios disponibles donde se considere la configuración del núcleo con combustible cerámico avanzado, este modelo es tomado como referencia. Posteriormente, el código determinista ERANOS es usado como alternativa para reducir el costo computacional de la simulación del núcleo del reactor. Como bien se conoce, en los cálculos de optimización se requieren un elevado número de evaluaciones de una función objetivo dada, por lo que reducir el tiempo de ejecución por parte del código de simulación es crucial para encontrar una solución óptima en un tiempo de cómputo razonable. Se experimenta estableciendo diferentes opciones de cálculo y estructura de grupos de energía. Entre los parámetros calculados a inicio de la vida del reactor se encuentran el factor efectivo de multiplicación de neutrones, el espectro de neutrones, las distribuciones de flujo neutrónico y de potencia, la constante Doppler, el efecto de la densidad del helio sobre la reactividad y la fracción efectiva de neutrones diferidos. También se analiza la evolución de la concentración de los principales isótopos de combustible en función del tiempo de operación. Se lleva a cabo un estudio del ciclo de combustible usando varios módulos incorporados en ERANOS para la gestión del combustible dentro del núcleo. Se presenta la metodología implementada para determinar las condiciones de equilibrio del reactor y se discuten los resultados aplicando el esquema de recarga y reacomodo propuesto.

En la segunda etapa, se aplican las técnicas metaheurísticas, algoritmo genético y búsqueda tabú, a la optimización del patrón de recarga de combustible del reactor ALLEGRO. El objetivo es maximizar el valor del factor efectivo de multiplicación de neutrones al final del ciclo satisfaciendo las restricciones operativas y de seguridad, como el factor de pico de potencia, el exceso de reactividad al inicio del ciclo y la tasa de generación de calor lineal. Se presenta la metodología usada para programar la interfaz entre el código de simulación ERANOS y el código de optimización, usada para el cálculo de la función objetivo. Se cubre en detalle la implementación práctica de cada técnica. Entre los aspectos más destacados del código programado, basado en algoritmos genéticos, se encuentra la incorporación de los operadores de cruce basado en correspondencia parcial y de cruce basado en el orden. También se presentan otras mejoras introducidas al código para reducir el tiempo de ejecución total. Finalmente, se implementó satisfactoriamente una versión de la técnica de búsqueda tabú. Esta técnica se seleccionó para solventar algunas de las deficiencias inherentes del método previo. Aplicando los conceptos de lista tabú variable y criterio de aspiración se evita el ciclado, lo que resulta en una diversificación de la búsqueda. La técnica demuestra ser una herramienta poderosa comparada con los algoritmos genéticos, en los que es habitual quedar atrapado en regiones de óptimos locales.

Table of Contents

List of Tables

List of Figures

Chapter 1

1. Introduction

This chapter presents an overview of the fourth-generation gas-cooled fast reactor, focusing on the ALLEGRO experimental reactor selected for the present study. It also briefly introduces the topic of in-core fuel management. Finally, it discusses the purpose of the research and outlines the main activities undertaken to achieve the objectives.

1.1. The Gas-cooled Fast Reactor

The gas-cooled fast reactor (GFR) is a promising advanced nuclear energy system selected and supported worldwide by the Generation-IV International Forum (GIF) for further research and development. GIF was established in 2000 as an international collaboration to research, develop and demonstrate the feasibility and performance of Generation-IV (Gen-IV) nuclear reactors. The Gen-IV goals were originally defined in 2002 and comprise four broad areas: sustainability, safety and reliability, economic competitiveness, and proliferation resistance and physical protection. The following six technologies, including the GFR, were identified as the most promising to meet these goals (U.S. DOE, 2002):

1. Sodium-cooled Fast Reactor (SFR);
2. Very High Temperature Reactor (VHTR);
3. Gas-cooled Fast Reactor (GFR);
4. Molten Salt Reactor (MSR);
5. Lead-cooled Fast Reactor (LFR);
6. SuperCritical Water-cooled Reactor (SCWR).

The latest GIF reports and roadmaps provide detailed information on the research and development efforts, the current technology status, and the timeline for the viability, performance, and demonstration phases for each Gen-IV system.

The GFR system is a high-temperature helium-cooled fast-spectrum reactor with a closed fuel cycle (GIF, 2018). It combines the advantages of fast spectrum systems for the long-term sustainability of uranium resources and waste minimization (through fuel multiple reprocessing and fission of long-lived actinides) with those of high-temperature systems (high thermal cycle efficiency and industrial use of the generated heat). This type of innovative nuclear system has several attractive features: the helium coolant is a single-phase coolant that is chemically inert, which does not dissociate or become activated, is transparent and while the coolant void coefficient is still positive, it is small and dominated by the Doppler feedback. The high outlet temperature places onerous demands on the capability of the fuel to operate continuously with the high-power density necessary for good neutron economics. The introduction, testing, and qualification of a suitable fuel and cladding material that can endure the harsh conditions of high temperatures and doses over long periods, while maintaining safe and stable reactor operation, is a crucial stage in the GFR technology development. (GIF, 2021).

Several GFR concepts have been under development considering different fuel forms such as coated fuel particles, silicon carbide blocks with dispersed microparticle fuel inside, silicon carbide plates with fuel pellets arranged in honeycomb structure, and the current design of hexagonal fuel assemblies composed of cylindrical rods of fuel pellets, arranged in a hexagonal array and surrounded with a hexagonal SiC wrapper (Perkó et al., 2015). The favored fuel material is the advanced carbide (ceramic) fuel, while for the pin clad it is silicon carbide fiber reinforced silicon carbide (SiC-fib/SiC). The reference concept for the GFR currently corresponds to the large-scale reactor, with 2400 MWth of thermal power (GFR2400), proposed by the French CEA (French Alternative Energies and Atomic Energy Commission) (Perkó, 2012; Perkó et al., 2015; Stainsby et al., 2011).

The European 75 MWth ALLEGRO gas-cooled fast reactor is the proposed concept to test the basic features of the GFR2400 reference reactor (GIF, 2021). The objectives of the ALLEGRO project are to demonstrate the GFR feasibility and to qualify specific technologies such as fuel, helium-related technologies, and safety systems (e.g., decay heat removal system). It will also demonstrate that these features can be integrated successfully into a representative system. The reactor will operate with two different cores: the starting core will consist of already well qualified fuel assemblies, with uranium oxide (UOX) or mixed oxide (MOX) fuel in stainless steel claddings will serve as a driving core for six experimental fuel assemblies containing the advanced ceramic fuel. The second core will consist entirely of ceramic fuel, enabling ALLEGRO to operate at the high temperature (GIF, 2021).

The development of ALLEGRO is governed by a consortium named V4G4 Centre of Excellence, established in 2012 in Slovakia by its four founding members: EK from Hungary, NCBJ from Poland, UJV Rez from the Czech Republic, VUJE from Slovakia, and two associated members, CEA from France, and CVR from Czechia (GIF, 2021). If the project is successful, ALLEGRO would be the first demonstration of a GFR built to date.

1.1. In-core Nuclear Fuel Management

Nuclear fuel management has traditionally been divided into two categories: out-of-core and in-core fuel management. In turn, the out-of-core is divided into the activities and decisions associated with the front-end and the back-end of the fuel cycle. Related to the front-end stage, decisions are typically made during the initial design phase of the reactor core and are relatively fixed. The back-end comprises all activities that ensure the safe separation of spent fuel and radioactive waste from the environment. Two main strategies are considered in the back-end stage: *open cycle* and *closed cycle*. In the former, the spent fuel is stored long-term and then disposed of in a deep geological repository, and in the latter, the nuclear fuel is reprocessed (the useful fuel material is extracted and reused to manufacture new fuel) and only the waste is disposed of. Out-of-core fuel management involves, among others, decisions related to the fresh fuel fabrication and the partially burnt fuel to reinsert into the core for additional energy production. The questions to be addressed are: What to manufacture? and What to reinsert? (Jayalal et al., 2014; Turinsky, 2005). Concerning the fresh fuel to be loaded, it includes lattice design decisions such as pin-wise enrichment and the type and configuration of reactivity control materials. It also encompasses the axial placement of lattice designs and the number of each assembly for a specific reload cycle. Now, for the partially burnt fuel assemblies to be reinserted into the core, the decision-making can be associated to the selection of spent fuel assemblies (e.g., from eligible assemblies in the previous cycle and spent-fuel pool) and the determination of appropriate cycle lengths.

In-core fuel management involves the arrangement of fresh and partially burnt fuel assemblies (i.e., loading pattern) and reactivity control mechanisms (i.e., control rod pattern and burnable poison material placement) within the core to optimize reactor performance during the next operating cycle, while ensuring that operating constraints are always met. It will answer the question “Where to position?” In-core decisions are made both during the initial core loading and at regular refueling intervals and are therefore more frequent. Out-of-core and in-core problems are closely related. For example, out-of-core decisions, such as number of fresh fuel assemblies to load will serve for in-core optimization problem.

The objective of in-core fuel management is to minimize the cost of electrical energy generation subject to operational and safety constraints. These constraints include cycle energy requirements, discharge burnup limits, thermal limits (e.g., limitation on the radial power peaking factors and the linear heat generation rate), reactivity limits (e.g., hot excess and shutdown margin), radial power imbalance, etc. (Turinsky, 2005).

This research will focus on the loading pattern optimization task as part of in-core fuel management decision making problem. The following features make this problem quite difficult to solve (Turinsky, 2005):

1. NP-hard combinatorial problem (the loading pattern defines combination of individual fuel and loading position in the core).

2. Nonlinear objectives and constraints (multiples local optima in the solution space),
3. Lack of derivative information.
4. Multi-objective optimization problem.
5. Large decision space.
6. Numerous constraints due to safety concerns in the design process.

In addition, due to the nonlinear nature there are many local optima in the solution space which makes it easy to get trapped in local optima. The nonlinearity also implies that the objective value cannot be obtained by the superposition of the other objective values, which makes it necessary to perform core calculations for each candidate design. This makes it computationally demanding.

Given the characteristics summarized above, the loading pattern optimization is considered as a complex global multi-objective combinatorial optimization problem. It has taken decades of significant effort to develop automated computational capability to assist the reload core nuclear design engineer in making nuclear fuel management decisions. This development has ranged from heuristic rules to utilization of mathematical optimization approaches (Turinsky, 2005). A well-organized historical review of the loading pattern optimization is provided by (Nissan, 2019). Early techniques to address this optimization problem were based on manual methods, in which experts used their knowledge and experience to find optimal solutions. Expert systems based on heuristic rulesets proved useful when introduced to the in-core optimization decision-making problem. These methods were limited in scope and prone to errors and inaccuracies. Evolving complexity of the design features and constraints often invalidated expert rules based on past design experiences. Later approaches developed were gradient based (e.g., linear programming) or hill climbing like methods (e.g., direct search). These methods have inherent disadvantages that are difficult to overcome in practical applications. They tend to get trapped in local optima and cannot treat multi-modality which is essential in the loading pattern optimization design (Yamamoto, 1998).

In the 1970s and 1980s, the invention of metaheuristic techniques such as genetic algorithms (Holland, 1975), simulated annealing (Kirkpatrick et al., 1983), tabu search (Glover, 1986), etc., represented a significant milestone in the optimization research field. These techniques introduced innovative approaches to address optimization problems. In-core fuel management immediately benefited from the emergence of these global optimization methods, especially to solve fuel load pattern optimization problems. They have been improved over the years and several new methods and approaches have been developed. One approach that has been particularly successful is the development of hybrid methods, which combine several existing methods to create more powerful and efficient algorithms.

Metaheuristic techniques have been successfully applied to solve the loading pattern optimization problem, mainly in light water reactors (boiling water reactors and pressurized water reactors) and pressurized heavy water reactors (Canada Deuterium Uranium), technologies that are currently in operation (IAEA, 2023). Since research in this field has been extensive, the following are just a few examples of published studies categorized by technique:

1. Genetic Algorithms – (DeChaine & Feltus, 1995; Martín-del-Campo et al., 2009; Ortiz et al., 2007; Ortiz & Requena, 2004; So et al., 2021; Toshinsky et al., 1999).
2. Tabu Search – (Castillo et al., 2004, 2005, 2007; François et al., 2013; Hill & Parks, 2015; Jagawa et al., 2001).
3. Simulated Annealing – (Kropaczek & Turinsky, 1991; Park et al., 2009; Stevens et al., 1995; Tran et al., 2021).
4. Particle Swarm Optimization – (Domingos et al., 2006; Khoshahval et al., 2010; Meneses et al., 2009, 2010; Yadav & Gupta, 2011).
5. Ant Colony Optimization – (de Lima et al., 2008; M Dorigo et al., 1996; M Dorigo & Gambardella, 1997; Esquivel-Estrada et al., 2011).

A key component in optimization calculations is the core simulation code to evaluate the loading patterns and compute objectives and constraints (note that the code will be interfaced with the optimizer.). The reactor physics code must provide reliable calculations for complex lattice, assembly, and core designs. The optimization process is iterative, so many evaluations are required, particularly when metaheuristic techniques are used. Therefore, a trade-off between the level of modeling detail and computational cost is needed. Commonly, deterministic transport codes are selected for three-dimensional core calculations. In addition, the code must include appropriate modules or calculation options for in-core fuel management, i.e., integrated routines to perform reloading and reshuffling, fuel cycle simulations, and core follow-up.

1.2. Purpose and Evolution of the Research

The main objective of this research is to develop a computer code for the fuel loading pattern optimization of the ALLEGRO fast reactor using metaheuristic techniques. The work was conducted as part of the *Nuclear Reactors and Fuel Cycles* project and was motivated by several factors. Firstly, a literature review on in-core fuel management showed a lack of studies on fuel loading pattern optimization in advanced GFR systems since most of the literature focused on light-water reactors. Secondly, there was extensive experience in fast neutron systems simulation from a previous analysis of the GFR2400 reactor. Thirdly, the team had expertise (since the 1990s) on fuel loading pattern optimization in BWR nuclear reactors (Laguna Verde Nuclear Power Plant) with several published papers. The reactor selected for the study is the ALLEGRO experimental reactor, proposed as an initial step for the Gen-IV GFR development. Figure 1.1 shows the schedule of the main activities carried out per semester to achieve the objectives.

The first stage of the research was a literature review to define the design parameters of the ALLEGRO reactor core corresponding to the ceramic fuel configuration. Based on these specifications, a reference model was developed using the Monte Carlo Serpent code, followed by a second model using the deterministic ERANOS 2.0 code to reduce the calculation time. Some discrepancies were found in the deterministic model compared to the reference model, mainly because several isotopes were missing from the cross-section libraries. This problem was solved using the latest available version of ERANOS (version 2.3N). In the next phase, a fuel cycle analysis was performed using the ERANOS built-in modules for in-core fuel management. This included defining the procedures to implement the reloading and reshuffling scheme proposed for the equilibrium cycle calculation. Subsequently, the LP optimization problem to be solved by metaheuristics was formulated.

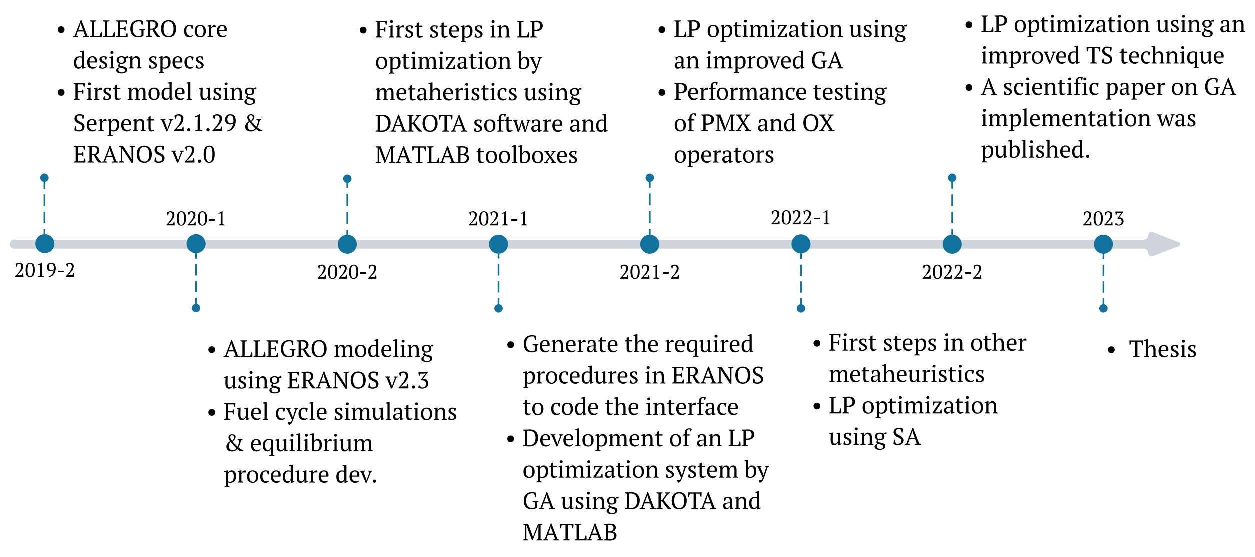

• ALLEGRO core design specs
• First model using Serpent v2.1.29 & ERANOS v2.0
• First steps in LP optimization by metaheristics using DAKOTA software and MATLAB toolboxes
• LP optimization using an improved GA
• Performance testing of PMX and OX operators
• LP optimization using an improved TS technique
• A scientific paper on GA implementation was published.
2019-2
2020-1
2020-2
2021-1
2021-2
2022-1
2022-2
2023
• ALLEGRO modeling using ERANOS v2.3
• Fuel cycle simulations & equilibrium procedure dev.
• Generate the required procedures in ERANOS to code the interface
• Development of an LP optimization system by GA using DAKOTA and MATLAB
• First steps in other metaheuristics
• LP optimization using SA
• Thesis

Figure 1.1: Timeline diagram showing the main activities carried out by semester.

The genetic algorithms technique was selected as the first solution method. The interface required to evaluate the LPs in ERANOS and compute the objective function was programmed. The develop of the genetic algorithms-based optimization system evolved from using DAKOTA software (Adams et al., 2022) and MATLAB Optimization toolbox (The MathWorks Inc., 2019) to programming an improved GA algorithm from scratch. MATLAB® software was used with an Academic License provided through the National Autonomous University of Mexico (UNAM). In the final stage, an improved tabu search technique for LP optimization was also implemented.

1.3. Structure of the Thesis

The document is organized into four main chapters. The introductory chapter provides an overview of the gas-cooled fast reactor, in-core nuclear fuel management, and the purpose and evolution of the research. Chapter 2 presents the neutron characterization of the ALLEGRO reactor core. Section 2.1 provides a detailed description of the core design and fuel composition. Section 2.2

contains the methodology for the core calculation in the Serpent and ERANOS simulation codes. Section 2.3 summarizes the results for different models, including important reactor core parameters such as the effective neutron multiplication factor, neutron spectrum, neutron flux and power distributions, Doppler effect, the effect of helium density on reactivity, the effective delayed neutron fraction, burnup calculation and equilibrium cycle calculation. The latter provides a comprehensive description of the procedures implemented in ERANOS to apply the proposed reloading and reshuffling scheme.

Chapter 3 focuses on the fuel loading pattern optimization of the ALLEGRO reactor using metaheuristics. This chapter starts with a brief overview of optimization fundamentals. Section 3.2 describes the objective function formulated for the optimization problem. Sections 3.3 and 3.4 provide a detailed description of the genetic algorithms and tabu search implementation for the ALLEGRO fuel loading pattern optimization.

Finally, Chapter 4 provides a summary and conclusions of the research presented in the previous chapters, highlighting the key findings, contributions, and limitations of the study. The thesis concludes with a list of References used throughout the study.

Chapter 2

2. Core Neutronic Characterization of the ALLEGRO Gas-cooled Fast Reactor

This chapter provides a detailed description of the ALLEGRO reactor core design specifications. The methodology for core simulation using the Monte Carlo Serpent code and the deterministic ERANOS code is discussed. Different calculation options are evaluated, and the results of core parameters such as k-eff, Doppler constant, neutron flux and power distributions, and beta-eff, are discussed. In addition, a fuel cycle study is carried out, including the equilibrium cycle calculation.

2.1. ALLEGRO Reactor Core Design

The present work considers the ALLEGRO reactor configuration corresponding to ceramic fuel assemblies with similar fuel composition to the GFR2400 reference design (Lima-Reinaldo & François, 2021). The current GFR design consists of a hexagonal fuel assembly consisting of cylindrical rods (pins) with fuel pellets, arranged in a hexagonal arrangement surrounded with a hexagonal SiC wrapper (Perkó et al., 2015).

A schematic view of the ALLEGRO reactor fuel pin is depicted in Figure 2.1, while Table 2.1 summarizes the geometrical properties at room temperature. This description for the fuel pin corresponds to the GFR2400 reactor (GFR reference design) described by (Perkó et al., 2015). The cladding (Clad) is made of SiC covered from the inside with thin metallic liners (rhenium, Re and tungsten-rhenium alloy, W14Re), which are metals with high melting temperature to improve the confinement of fission products. The gap between the pellets and the metallic liners is filled with helium at 1 MPa. The addition of SiC fiber (SiC-fib) to the SiC cladding material aims to improve its mechanical strength and at the same time its tightness, preventing gas diffusion to the coolant through porosities in the ceramic material. The total height of the fuel pin is 135 cm including the active height of 86 cm and the fission gas plenum at the top (9 cm high) and at the bottom (40 cm high). Each fuel assembly (FA) contains 90 fuel pins and a center pin of SiC structural material arranged in a hexagonal array with a hexagonal SiC wrapper (Čerba et al., 2014). Figure 2.2 shows a cross sectional view of the fuel assembly. The 30 cm high axial reflectors, above and below the plenums, are made up of zirconium carbide (ZrC). The axial shielding at the top and bottom of the axial reflector is made of natural boron carbide ($NATB_4C$) with an axial length of 50 cm.

The ceramic fuel material used is composed of a mixture of uranium and plutonium carbide, $(U, Pu)C$, where the isotopic composition of the fertile fuel, uranium, corresponds to natural uranium; while that of the fissile fuel, plutonium, is characteristic of twice-recycled MOX fuel. The assumed volume fraction of PuC in the core is 27.5%, according to (Čerba et al., 2014) (MOX and ceramic fuel assemblies comparative study). The porosity of the ceramic fuel is 20% (Perkó et al., 2015). The fuel material parameters are presented in Table 2.2.

Table 2.1: Geometrical properties of the ALLEGRO reactor core.

Pin		Fuel assembly		Core	
Region	Radius (cm)	Region	Radius (cm)	Parameter	Value
Fuel pellet	0.3355	Wrapper (in)	5.3117	No. of FAs	87
Gap	0.350	Wrapper (out)	5.5117	No. of CSDs	6
W14Re liner	0.354	Coolant (out FA)	5.6617	No. of DSDs	4
Re liner	0.355	Active height	86	No. of RRs	174
Clad	0.455	No. of pins	91[a]	No. of BRs	198
SiC-fib	0.458				
Lattice pitch	1.1				

a- The number of pins corresponds to 90 fuel pins plus a SiC center pin.

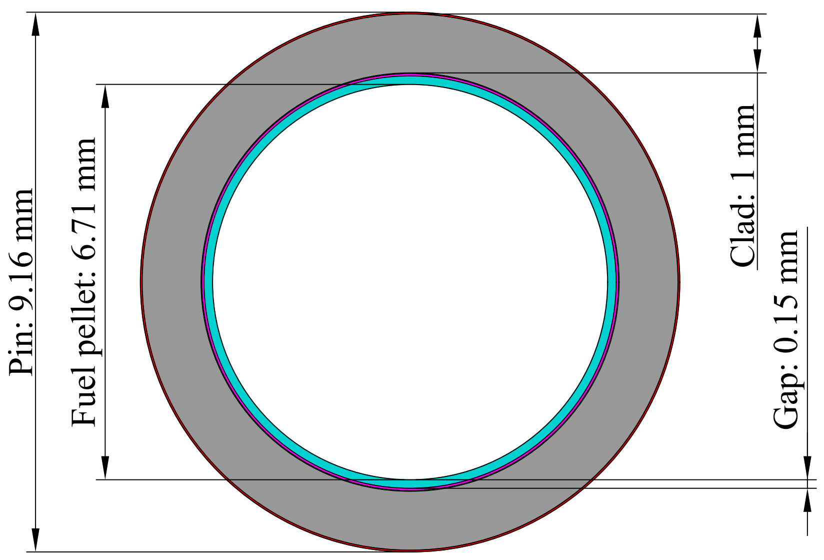


Figure 2.1: Cross sectional view of the fuel pin of the ALLEGRO reactor.

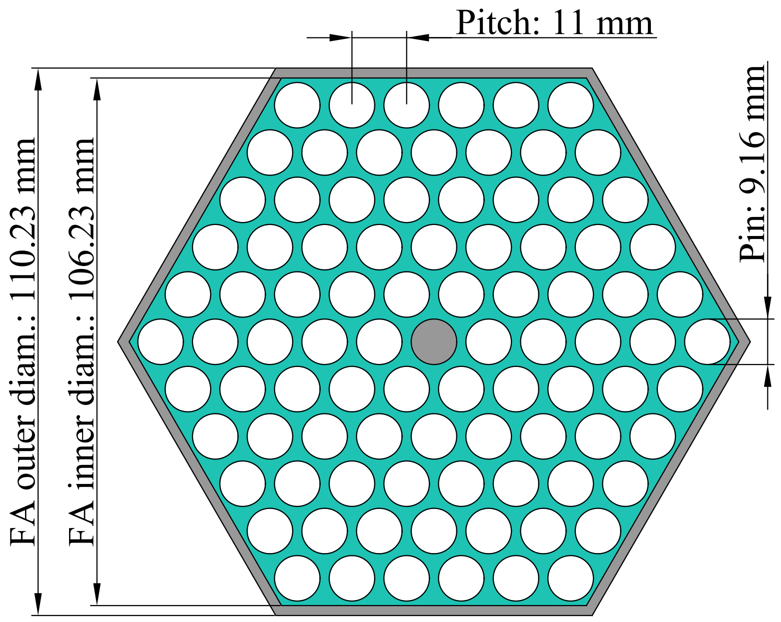


Figure 2.2: Cross sectional view of the fuel assembly of the ALLEGRO reactor.

Table 2.2: Fuel isotopic composition of the ALLEGRO reactor core.

Pu vector	$Pu_{fr,i}$ (%)	U vector	$U_{fr,i}$(%)
^{238}Pu	2.7	^{235}U	0.72
^{239}Pu	56.0	^{238}U	99.28
^{240}Pu	25.9		
^{241}Pu	7.4		
^{242}Pu	7.3		
^{241}Am	0.7		
PuC molar mass (g/mol)	251.677	UC molar mass (g/mol)	250.039
PuC density (g/cm^3)	10.880[a]	UC density (g/cm^3)	10.904[a]
PuC volume fraction (%)	27.5		
UC volume fraction (%)	72.5		

a- The density includes 20% porosity of the ceramic fuel.

Figure 2.3 and Figure 2.4 show the cross sectional and axial view of the reactor core, respectively. The active core consists of only one zone with 87 hexagonal fuel assemblies arranged in a hexagonal array. The 174 radial reflectors, like the lower and upper reflectors, are made up of ZrC. The 198 $NATB_4C$ assemblies surrounding the core make up the radial shielding. For these core structural elements, the homogeneous mixture with the composition given in Table 2.3 is considered and the axial layout is shown in Figure 2.4.

The reactivity is controlled by two systems of control rods: 6 CSDs (Control System Devices or control rods) and 4 DSDs (Diverse Safety Devices or safety rods) (Pónya & Czifrus, 2017). The control rods are fully withdrawn and located above the active core. The rod follower (RFOL) fills the region between the top of the lower reflector and the bottom of the control rod.

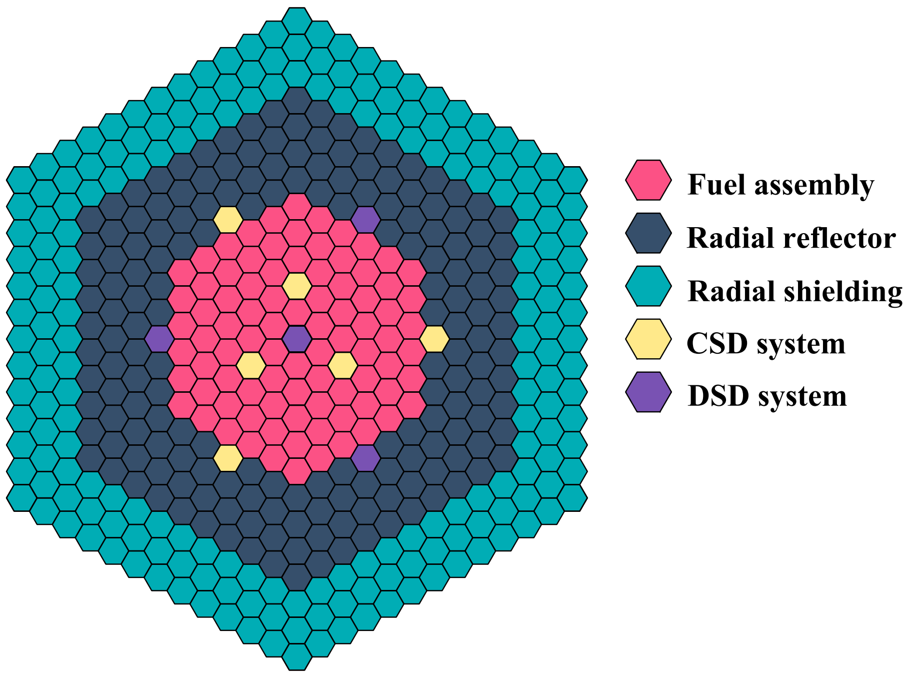


Figure 2.3: Cross sectional view of the ALLEGRO reactor core (xy plane). Note that the fuel pins are not represented.

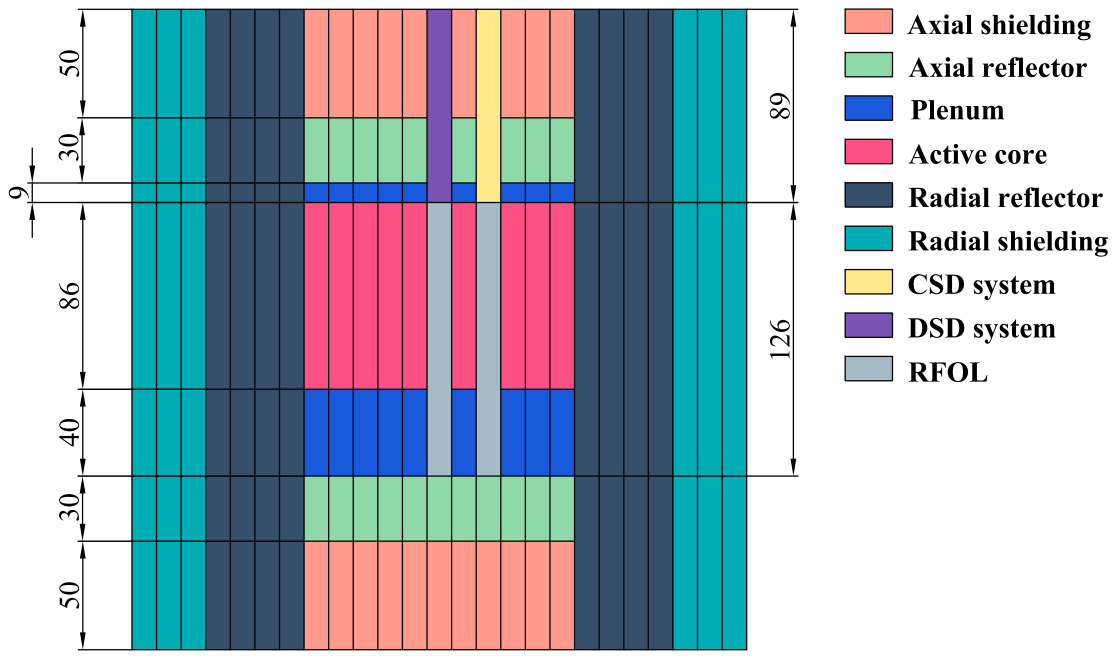


Figure 2.4: Axial view of the ALLEGRO reactor core (yz plane).

Table 2.3: Axial layout and structural elements composition of the ALLEGRO core.

Axial region	Height (cm)	Material	Volume fraction (%)
Upper shielding (US)	50	$NATB_4C$ AIM1 He at 7 MPa	50 10 40
Upper reflector (UR)	30	Zr He at 7 MPa	75 25
Upper plenum (UP)	9	He 1 MPa He 7MPa Re W14Re SiC	31.19 39.24 0.18 0.72 28.67
Fuel	86	–	–
Lower plenum (LWP)	40	He at 1 MPa He at 7MPa Re W14Re SiC	31.19 39.24 0.18 0.72 28.67
Lower reflector (LR)	30	ZrC He at 7 MPa	75 25
Lower shielding (LS)	50	$NATB_4C$ AIM1 He at 7 MPa	50 10 40
Total height	295	–	–
Radial reflector (RR)	195	ZrC He at 7 MPa	80 20
Radial shielding (RS)	195	$NATB_4C$ AIM1 He at 7 MPa	70 10 20
Control and shutdown rods (CSD and DSD)	89	B_4C AIM1 SiC He at 7 MPa	30.26 11.22 10.85 47.67
Rod follower (RFOL)	206	AIM1 SiC He at 7 MPa	1.2 10.85 87.95

2.1.1. Fuel material densities

To determine the mass fraction, the atomic density (N_i in cm^{-3}), and the mass (m_i in kg) of the individual heavy metal isotopes in the active core, were calculated. These fractions will be used to define the fuel material in both Serpent and ECCO/ERANOS codes. The isotopic composition is given in Table 2.1. The formulas described by (Perkó, 2012) for the GFR2400 fast reactor were adopted. Then, the atomic densities of Pu isotopes N_i^{Pu} (including ^{241}Am) are calculated as follows:

$$N_i^{Pu} = \frac{V_{PuC}}{V_{fuel}} \times \frac{\rho_{PuC}}{M_{PuC}} \times \frac{Pu_{fr,i}}{100} \times N_A \quad [cm^{-3}] \tag{2.1}$$

Where V_{PuC} and V_{fuel} are the volume of PuC and the total fuel volume, respectively. The volume fraction of PuC $v = \frac{V_{PuC}}{V_{fuel}}$ (in %), the density ρ_{PuC} (in $g.cm^{-3}$), the plutonium fraction $Pu_{fr,i}$ (in %), and the molar mass M_{PuC} (in $g.mol^{-1}$), are given in Table 2.2. Likewise, for the U isotopes, we get that N_i^U is equal to:

$$N_i^U = \left(1 - \frac{V_{PuC}}{V_{fuel}}\right) \times \frac{\rho_{UC}}{M_{UC}} \times \frac{U_{fr,i}}{100} \times N_A \quad [cm^{-3}] \tag{2.2}$$

For the natural carbon, N_i^C is:

$$N_i^C = \left[\frac{V_{PuC}}{V_{fuel}} \times \frac{\rho_{PuC}}{M_{PuC}} + \left(1 - \frac{V_{PuC}}{V_{fuel}}\right) \times \frac{\rho_{UC}}{M_{UC}}\right] \times N_A \quad [cm^{-3}] \tag{2.3}$$

Finally, the masses of the individual isotopes (m_i) are calculated as:

$$m_i = N_i \times \frac{V_{fuel}}{N_A} \times M_i \quad [g] \tag{2.4}$$

where, N_i are the previously calculated atomic densities for each isotope, V_{fuel} is the total fuel volume, M_i is the molar mass of the isotopes, and $N_A = 6.02214 \times 10^{23}\ mol^{-1}$ is the Avogadro number. Given these, the atomic densities and masses of each isotope calculated are summarized in Table 2.4.

As additional data, the total volume of fuel material is calculated as:

$$V_{fuel} = \pi \times r_{pellet}^2 \times n_{FA} \times n_{pins} \times h_{active} = 238.12 \times 10^3\ cm^3 \tag{2.5}$$

Table 2.4: Fuel inventory in the ALLEGRO reactor core.

Isotopes	$N_i\ (barn^{-1}cm^{-1})$	Mass (kg)
^{238}Pu	1.933E-04	18.195
^{239}Pu	4.009E-03	378.960
^{240}Pu	1.854E-03	176.003
^{241}Pu	5.298E-04	50.497
^{242}Pu	5.226E-04	50.021
^{241}Am	5.011E-05	4.777
Pu	–	673.676
^{235}U	1.890E-02	1779.265
^{238}U	1.371E-04	12.741
U	–	1792.006
C	2.620E-02	124.426
Total	–	2594.884

2.2. ALLEGRO Reactor Design in ERANOS and Serpent Codes

The neutron population in a nuclear reactor is governed by the transport equation, also called the Boltzmann equation. In practice, there are two main classes of codes for solving the transport equation. First, deterministic transport codes, which use a computational numerical resolution of the transport equation or its simplified version, diffusion equation, to determine the neutron flux (Parisot, 2015). They generally require the discretization of space, energy, and time variables. And secondly, the Monte Carlo neutron transport codes, where the life of each neutron is simulated individually and the events of interest are the interactions which they induce with the different nuclei of the atoms that constitute the environment gone through (scattering, capture, fission, etc.) (Parisot, 2015). This series of events is called a history and simulating enough neutron histories results in a detailed simulation of the transport process. Its main advantage is the capability to model the geometry and interaction physics without major approximations. Monte Carlo codes can use neutron interaction data in a tabular point-wise form, the opposite of deterministic codes that require data preprocessing into a group-wise format (Leppänen, 2007). This is the reason why they are often qualified as continuous energy Monte Carlo transport codes, and they have got the status of reference transport codes (Parisot, 2015). The disadvantage is that the modelling of complicated systems requires high computational cost. Monte Carlo codes are widely used in various reactor physics applications, including criticality calculations, spatial homogenization, fuel cycle studies, radiation shielding problems, research reactor modeling and validation of deterministic transport codes, etc.

In this study, for the ALLEGRO reactor core simulation, the Monte Carlo Serpent code was used for reference calculations. The deterministic ERANOS code was also employed to reduce the computational time. Subsection 2.2.1 and 2.2.2 describe the reactor core modeling using the ERANOS and Serpent codes, respectively.

2.2.1. ERANOS deterministic code

ERANOS (European Reactor Analysis Optimized calculation System) is a deterministic neutronic calculation code system for fast reactors analysis. ERANOS has been developed and validated with the aim of providing an adequate basis for reliable neutron calculations, as well as the ability to treat fuel assemblies of advanced reactors (Ruggieri et al., 2006). The latest ERANOS release (version 2.3N) is used in this study. The software package was obtained through computer program service of the OECD/NEA Data Bank.

ERANOS adopts a complex and extensive modular structure. LU (User Language) is the script language used as the interface between the user and the calculation code. This language has its own syntax with a few specific features, as the manipulation of basic data types. The LU interpreter controls the execution of the modules (or functions) and saves the results and data. A structured dataset is represented as a SET (Structured ERANOS Tree). A SET can be created or used by the modules themselves or by the user via the LU. It could be a geometry, cross-sections, flux, isotopic concentrations, etc.

ERANOS provides a full core calculation through a sequence of many complex steps. The simulation is mainly separated into two parts: cell and core calculations. First, the cell (or lattice) calculation is performed using the ECCO module to prepare the cross sections and matrices for a given number of energy groups and for an equivalent medium corresponding to the cell. These group cross sections are then used in the core calculation to determine the reactor characteristics. The flowcharts in Figure 2.5 and Figure 2.6 show the core calculation scheme.

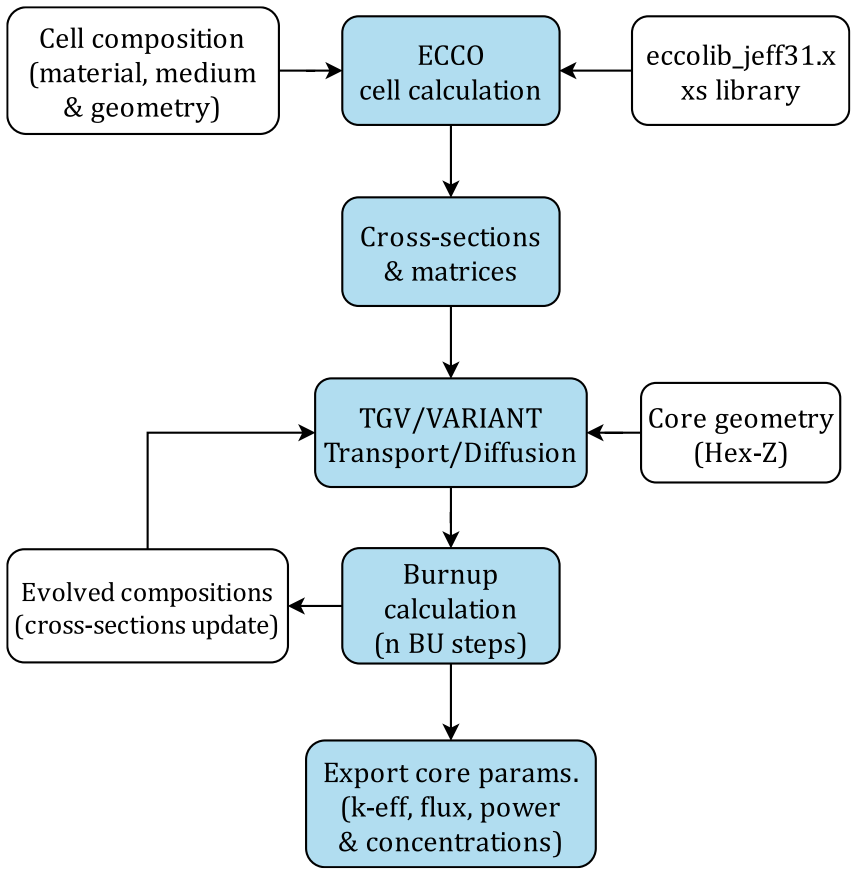

Cell composition (material, medium & geometry)
ECCO cell calculation
eccolib_jeff31.x xs library
Cross-sections & matrices
TGV/VARIANT Transport/Diffusion
Core geometry (Hex-Z)
Evolved compositions (cross-sections update)
Burnup calculation (n BU steps)
Export core params. (k-eff, flux, power & concentrations)

Figure 2.5: Core calculation scheme in the ERANOS/ECCO code.

For cell calculations, the European Cell Code (ECCO) provides cross sections and matrices for use in reactor core calculations performed by other modules available within ERANOS. ECCO has a resonance self-shielding solution algorithm based on the sub-group method combined with a fine group transport calculation in complex heterogeneous structures based on collision probabilities (Ruggieri et al., 2006). The ECCO/ERANOS code package contains four neutron cross section libraries derived from the JEFF-3.1 evaluated nuclear data files (ECCOLIB_JEFF_31.x; x: 1968, 33, 172 and 175 energy groups libraries). For core calculations, the ERANOS package includes several modules for solving the neutron transport equation. For example, the BISTRO module uses the discrete ordinates method (S_N method) and the finite difference method, while the TGV/VARIANT module uses the spherical harmonics method (P_n method) and the variational nodal method, for angular and spatial discretization, respectively.

The simulation starts using the ECCO module. First, the media (homogeneous regions) that make up the core are created. In this step the concentrations of the different isotopes that make up the medium are given. The composition used for the fuel and other elements is given in Table 2.2 and Table 2.3 respectively. Second, the geometrical description of the cell, the basic element to build a core, is defined. ERANOS allows defining cells with complex geometries such as 2D hexagonal lattices with pins arranged in a hexagonal array surrounded by a hexagonal wrapper. This geometry, shown in Figure 2.2, was used to define the cell corresponding to the fuel assembly.

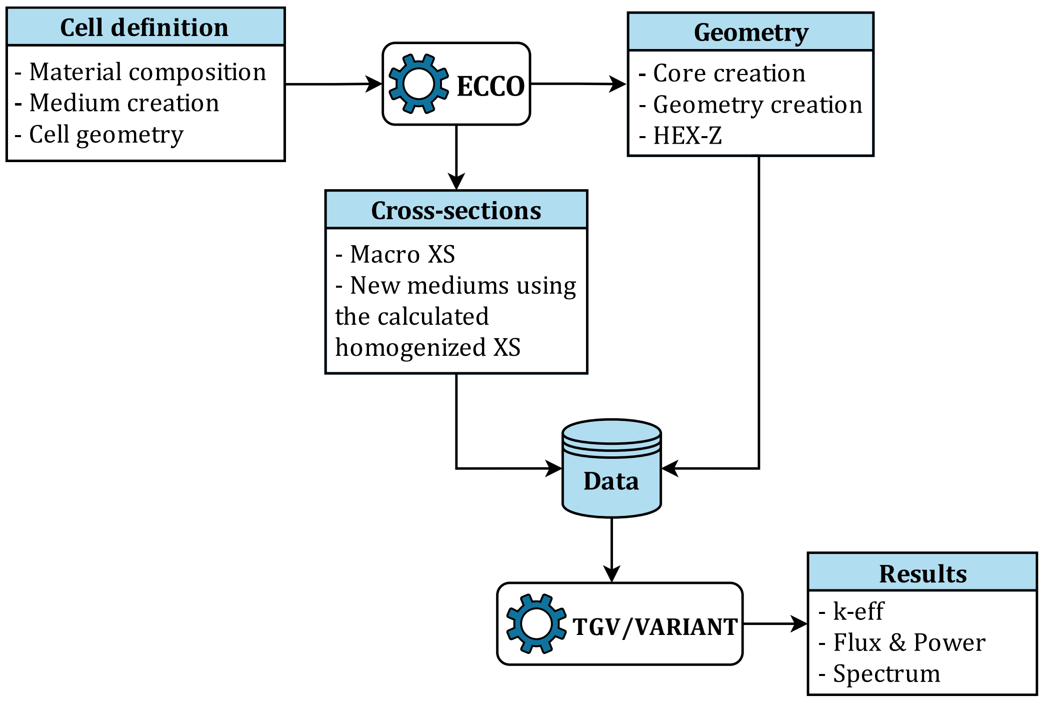


Figure 2.6: Core calculation procedure in ERANOS.

A homogeneous composition was defined for the other structural components of the core. Then ECCO generates the appropriate cross sections for each specified cell and the nuclear libraries by means of the collision probability method. These cells defined in the ECCO correspond to the fuel assembly, plenums, reflectors and shielding.

A full ECCO calculation may consist of several steps in which different geometrical models (heterogeneous or homogeneous), group structures, processing of the flux or the balance, and resonance shielding treatments are used. To perform the ECCO calculation for the fuel cell, the reference route represented in the diagram in Figure 2.7 was used. The calculation is performed in a sequence of 5 steps. Firstly, for a homogeneous geometry, the fission source is calculated and then the axial buckling value is searched to obtain a critical cell. In this step, the 33 energy groups library (or broad group cross section library for fast spectrum applications) is used. The resonance self-shielding effect is treated for all elements. Second, the calculation is performed for a heterogeneous geometry, the buckling found in the previous step and the broad group library. In the third step the calculation is refined using 1968 energy groups (or fine group library), the heterogeneous geometry, and specifying the nuclides to be treated. At the end, the calculated cross sections are condensed to 33 energy groups. The energy boundaries for condensation were defined from the 1968-group structure. Table 2.5 shows the 33-group structure used. Then, in the fourth step, the critical buckling is searched for this energy group structure. Finally, these self-shielded cross sections and matrices are condensed and smeared to provide effective cross sections and matrices for the whole in the required broad group scheme.

Step 1	Step 2	Step 3
Geometry: Homogeneous Group scheme: Broad group Elements: All Buckling: Search	Geometry: Heterogeneous Group scheme: Broad group library Elements: All Buckling: From STEP 1	Geometry: Heterogeneous Group scheme: Fine group Elements: from the fine lib Buckling: from STEP 1 Condensation to 33 groups
Step 4	Step 5	Step 6
Geometry: Heterogeneous Group scheme: 33 group Elements: All Buckling: Search	Geometry: Homogeneous Homogenization Group scheme: Broad group Elements: All Buckling: from STEP 4 Output library: XS and Flux	Geometry: Homogeneous Homogenization Group scheme: Broad group Elements: All Buckling: from STEP 5 Condense to 7 groups Output library: XS and Flux

Figure 2.7: Reference calculation route used in ECCO.

Table 2.5: ECCO 33-group energy structure.

Group	Energy (MeV)	Group	Energy (MeV)	Group	Energy (MeV)
1	1.96403E+01	12	6.73795E-02	23	3.04325E-04
2	1.00000E+01	13	4.08677E-02	24	1.48625E-04
3	6.06531E+00	14	2.47875E-02	25	9.16609E-05
4	3.67879E+00	15	1.50344E-02	26	6.79041E-05
5	2.23130E+00	16	9.11882E-03	27	4.01690E-05
6	1.35335E+00	17	5.53084E-03	28	2.26033E-05
7	8.20850E-01	18	3.35463E-03	29	1.37096E-05
8	4.97871E-01	19	2.03468E-03	30	8.31529E-06
9	3.01974E-01	20	1.23410E-03	31	4.00000E-06
10	1.83156E-01	21	7.48518E-04	32	5.40000E-07
11	1.11090E-01	22	4.53999E-04	33	1.00000E-07

These self-shielded macroscopic cross sections are saved to a file for use in subsequent full core calculations. The sixth step is added to condense the cross sections to 7 energy groups shown in

Table 2.6. This structure, proposed by (Palmiotti et al., 2011) covers the entire spectrum and could be used for studies of fast, epithermal, and thermal reactors.

Table 2.6: 7-group energy structure.

Group	Energy (MeV)	Group	Energy (MeV)
1	1.96403E+01	5	2.03468E-03
2	2.23130E+00	6	2.26033E-05
3	4.97871E-01	7	5.40000E-07
4	6.73795E-02		

For the subcritical assemblies, the procedure is different, the source is taken from the fuel cell, already calculated (fifth step) and the semi-empirical buckling value given by the following Equation (2.6) is used (Rimpault, 1997):

$$B^2 = \frac{5}{8}\left(\frac{\pi}{H}\right)^2 \tag{2.6}$$

Where H is the thickness of the subcritical medium.

After the cross sections processing in ECCO, the core model with a 3D hexagonal configuration (Hex-Z) shown in Figure 2.3 and Figure 2.4 was generated. To build the core, the $core\_creation$ and $geometry\_creation$ modules, were used. Firstly, the $core\_creation$ module generates the hexagonal lattice by defining the central position, the concentric rings corresponding to each radial zone (fuel, reflector and shielding), and the lattice pitch. Starting from a central position $(30/30)$ and using the shifting rule shown in the diagram in Figure 2.8, the core map is generated (two integers represent the coordinates of a hexagon).

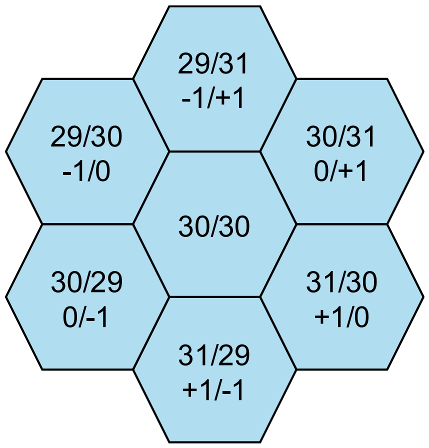


Figure 2.8: Rule used in ERANOS to generate the hexagonal lattice.

The control rods positions are specified according to their individual coordinates. Secondly, a coarse axial mesh of 58 bins (approximately 5 cm width each bin) was defined. The axial layout of the zones can be seen in Figure 2.9, which is a combination of Figure 2.4 and the Table 2.3 where the composition is given. To complete the core creation, the individual hexagonal assemblies are defined. The corresponding axial medium is associated with the previously generated axial mesh. Then, the assemblies are assigned to their corresponding zone in the core.

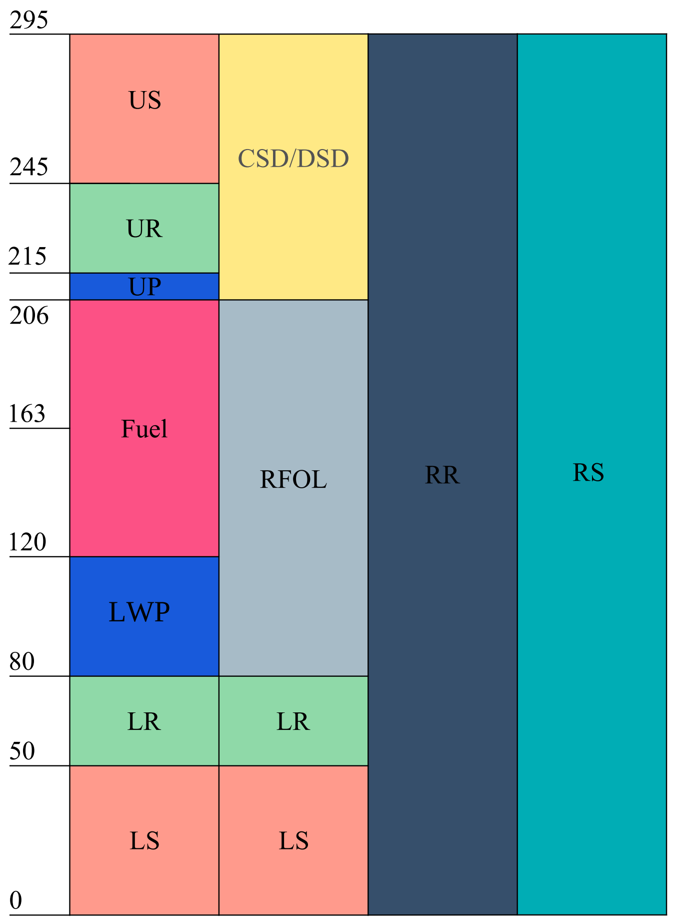


Figure 2.9: Axial layout of the ALLEGRO reactor core defined in ERANOS. LS/US: lower and upper shielding, LR/UR: lower and upper reflector, LWP/UP: lower and upper plenum, RR: radial reflector and RS: radial shielding, CSD/DSD: control and safety rods, RFOL: control rod follower. Dimensions are given in cm.

Finally, using the *geometry_creation* module, the geometry is defined by using the previously generated core configuration. In this step, the boundary conditions and symmetry options can be set.

For core calculations the TGV/VARIANT ERANOS module was used. VARIANT (VARIational Anisotropic Neutron Transport) solves the multigroup steady-state neutron diffusion and transport equations in two and three dimensional cartesian and hexagonal geometries using variational nodal methods with one mesh cell (node) per hexagonal assembly (Palmiotti et al., 1995). Angular variables are expanded with complete (P_n) or simplified spherical harmonics (SP_n) approximations ($n = 1, 3, 5$) with full anisotropic scattering capability. In the mathematical formulation of the spherical harmonics expansions, P_n are the associated Legendre polynomials where n represents the expansion order. The P_1 corresponds to the diffusion approximation. The spatial dependence of the flux variables is represented by complete polynomials within coarse mesh nodes, and along internode interfaces. Polynomials as high as fourth order for cartesian and sixth order for hexagonal geometries are implemented.

The proper execution of the TGV module depends on the input options set by the user. The macroscopic cross-sections and the core geometry sets are mandatory. These data were previously calculated and saved to a binary data file by the ECCO and geometry modules. In addition, the P_n transport (SP_n or diffusion) calculation options for flux and leakages, and the spatial approximations orders, i.e., source and fluxes, within the node and leakages on the surfaces of the nodes, must be specified. For the latter values, two calculation routes are recommended based on the CPU time, one for reference calculations (detailed calculation but high consumption) and the other for design calculations (less consumption by reducing the spatial expansion orders). In this work, to validate this model using the one designed in the Serpent Monte Carlo code, the reference route was used. The order of spatial approximations, flux and leakage expansion used are specified in Table 2.7. To reduce the calculation time at a reduced penalty in the precision of the results, the transport SP_3 and diffusion solutions are analyzed.

The core evolution in ERANOS is treated by solving the Bateman equations using a constant average flux per region for a given depletion time. The calculation procedure used starts by computing the change in isotope concentration, then the macroscopic cross sections are recalculated and the initial reactor concentrations are updated at each time step. The recalculation of the flux at each step is performed using the TGV/VARIANT module.

Besides the described modules related to core calculation procedure a variety of modules computes or extracts specific information from the code output. Among the results that can be processed are:

radial and axial neutron flux, reaction rates, material balance (mass and atoms), neutron balance, effective delayed neutron fraction, etc.

Table 2.7: Calculation parameters in the TGV/VARIANT module.

Calculation option	Angular Expansion		Spatial Expansion		
	Flux expansion	Leakage expansion	Interior	Interface	Source
Diffusion	P_1	P_1	6	1	3
Transport	P_3	P_3	6	1	3
Simplified Transport	SP_3	SP_3	6	1	3

Note: Interior, Interface and Source are the orders of the polynomial approximation of the fluxes within the node, the leakages on the surfaces of the nodes and the source within the node, respectively.

2.2.2. Serpent Monte Carlo code

To design a 3D heterogeneous reference model of the ALLEGRO core, Serpent, a three-dimensional continuous-energy Monte Carlo reactor physics burnup calculation code developed at the VTT Technical Research Center of Finland, Ltd. was used (Leppänen et al., 2015). The code version used in this work is Serpent 2.1.29.

Serpent is a powerful and user-friendly code, widely used for benchmark calculations. Unlike ERANOS, which is characterized by its modular structure, a full core simulation is more straightforward in Serpent. The input parameters can be set in a single text file without the need to call external modules. The user can execute complex routines by declaring cards that define the calculation options.

The geometry description in Serpent is based on a universe-based constructive solid geometry model, which allows the description of practically any two or three-dimensional fuel or reactor configuration (Leppänen et al., 2015). This means that the geometry is divided into levels. For the 3D heterogeneous active core, the top level corresponds to the fuel pin, where the pellets are surrounded by the cladding and coolant. The next level is the fuel assembly cell, in which the pin universes are arranged in a hexagonal lattice. Figure 2.2 shows the pin-based heterogeneous geometry corresponding to the built fuel assembly. For the structural elements, a homogeneous description was defined. At the last level, these assemblies are arranged in a hexagonal lattice to build the tree-dimensional active core, surrounded by the reflectors, and shielding, both with a homogeneous composition (see Figure 2.3). Each defined cell consists of homogeneous material regions, each of which have their own macroscopic total cross-sections. Serpent reads continuous-energy cross-sections from ACE format data libraries based on JEFF-3.1 evaluated nuclear data files (other libraries are available). The simulation is carried out in generations or cycles, where neutron tracking is performed for the defined geometry, based on the combination of conventional surface-tracking and the Woodcock delta-tracking method (Leppänen et al., 2015). By defining detectors, which are based on the collision flux estimator, integral flux and reaction rates in materials, cells or universes are calculated. The default calculation mode is the *k-eigenvalue* criticality source method for calculating an estimate of the effective neutron multiplication factor, k-eff, which determines the criticality of the system (Kaltiaisenaho, 2014).

The burnup calculation in Serpent is performed using built-in calculation routines and without coupling to external solvers which simplifies the use of this capability. Available burnup algorithms include the conventional Euler and predictor-corrector method with linear interpolation for the corrector calculation and the Bateman equations are solved by default using the Chebyshev Rational Approximation Method (CRAM), an advanced matrix exponential solution (Leppänen, 2015). The second available option is the Transmutation Trajectory Analysis (TTA) method, an advanced version of the linear chain method (Cetnar et al., 2021; Oettingen, 2021; Stanisz et al., 2016).

2.3. ALLEGRO Reactor Core Simulation Results

This section presents the ALLEGRO core modeling results for different case studies. Most of the discussion presented was published by the author in (Lima-Reinaldo & François, 2021).

2.3.1. Effective neutron multiplication factor (k-eff)

The following ALLEGRO reactor core models were simulated according to geometry, calculation method and energy groups:

- Model 1: 3D heterogeneous reference model using the Serpent code.
- Model 2 and Model 3: 3D homogeneous using the TGV/VARIANT module of the ERANOS code for diffusion calculation with 7 and 33 energy groups, respectively.
- Model 4 and Model 5: 3D homogeneous using the TGV/VARIANT module of the ERANOS code for transport calculation (with a P_3 angular expansion) with 7 and 33 energy groups, respectively.
- Model 6 and Model 7: 3D homogeneous using the TGV/VARIANT module of the ERANOS code for transport calculation (with simplified spherical harmonic approximation) with 7 and 33 energy groups, respectively.

The k-eff values obtained are shown in Table 2.8. For Monte Carlo calculations, 10^5 neutrons per generation and 600 generations, 500 active and 100 inactive, were used to ensure good accuracy of the results (relative standard error in the order of 10 pcm). The simulation was performed in hot condition, considering a nominal temperature of 1200 K and 900 K for the fuel and core structural materials, respectively. Both reactor physics codes were run on a multi-core workstation (Intel® Xeon® CPU E5-2623 v4 - 2.60GHz x 15) with 64 GB of RAM. Serpent was executed in OpenMP mode using 10 parallel threads, while ERANOS using a single core.

The discrepancies in the results are given by the geometry and the calculation method. The best agreement with respect to the reference case was obtained with the transport calculations using the 7-group structure. The relative difference between the models does not exceed 0.2% (0.68% with the 33-group structure). For the diffusion calculations lower k-eff values were obtained with respect to the Monte Carlo calculations (for both energy group structures). The worst value was obtained using the 33-group energy structure, with a relative difference of -2.8%. Large differences were also found for simplified transport calculations (Lima-Reinaldo & François, 2021).

Table 2.8: k-eff results.

Model	Geometry		Calculation method	Energy groups	k-eff value	Relative difference (%)	Running time
	Fuel	Structure					
1	Hetero.	Homo.	Serpent MC	–	1.06548 ± 1.17E-04	–	3.2 h
2	Homo.	Homo.	Diffusion	7	1.04590	-1.84	11.0 s
3	Homo.	Homo.	Diffusion	33	1.03541	-2.82	46.0 s
4	Homo.	Homo.	P_3	7	1.06759	0.20	2.8 min
5	Homo.	Homo.	P_3	33	1.05819	-0.68	20.4 min
6	Homo.	Homo.	SP_3	7	1.05313	-1.16	49.0 s
7	Homo.	Homo.	SP_3	33	1.04373	-2.04	7.6 min

Notes: 1-The JEFF-3.1 cross section library was used in both codes. 2- The core calculations in ERANOS were performed with the TGV/VARIANT module. 3- All calculations were performed at hot conditions, 1200 K and 900 K for the fuel and core structural materials, respectively. 4- The relative difference (in %) is defined as: $(k_{eff}^{calc}/k_{eff}^{ref} - 1)$, where the calculated value corresponds to the ERANOS models and the reference value, to the Serpent model.

As is known, problems arise in diffusion calculations when the spatial dependence of the neutron flux is too strong, near highly absorbing materials and regions with low density material, such as control rod followers. Leakage tends to be overestimated resulting in an underestimation of k-eff. In addition, there is a considerable impact on the determination of leakage-dependent reactivity effects such as the void reactivity coefficient (Lima-Reinaldo & François, 2021). The ALLEGRO reactor core, which is small and has a hard neutron spectrum, contains large helium regions, so the use of the diffusion method could have a negative effect on the precision of the results. On the other hand, in GFR reactors, due to the low density of helium, there is an effect known as *neutron streaming*, or neutron leakage in certain preferential directions due to the heterogeneities of the reactor core (Parisot, 2015). The large helium channels present in the core design, specifically the control rod followers and plenums, constitute preferential leak paths for neutrons whose direction is close to that of the gas flow, since neutrons can easily travel in helium. The treatment of these leakages in the core calculations is a challenge for the simulation of GFR reactors, especially when using the diffusion approximation. The use of transport theory or Monte Carlo techniques more adequately accounts for neutron streaming in the low density regions (Waltar et al., 2012).

Analyzing the influence of the energy structure, using 7 and 33 energy groups, the relative difference between the results remains around 1% for the calculation methods used. A comparison was also made between the execution times for each calculation method. In the diffusion calculation and the 7-group structure, the shortest times were obtained, in the order of seconds. In the transport calculations, a significant reduction (7 times shorter) of the execution time was achieved using the 7-group structure and obtaining k-eff values close to the reference model. In Serpent, the simulation took much longer, in the order of 3 hours considering that the computations were performed in parallel.

2.3.2. Neutron spectrum

The neutron spectrum for models 1, 5 and 7 are compared in Figure 2.10 as flux per unit lethargy (in cm^{-2} s^{-1}) in the neutron energy range from 1E-7 to 19.64 MeV. The spectrum is calculated at the center and midplane of the active core. In Serpent a fine energy grid (500 energy bins) was defined, while in ERANOS the 33-group energy structure was used. The calculated spectrum is characteristic of $(U, Pu)C$ fuel and is comparable to that obtained by (Perkó et al., 2015) for the GFR2400 reference design. A good agreement was obtained between the models mainly above 100 eV, for lower energies the difference is given by the energy structure used. The resonances corresponding to the ^{238}U capture cross section at energies 6.67 eV, 20.87 eV and 36.68 eV can be clearly identified in the fine energy grid, as well as the ^{240}Pu and ^{242}Pu resonances at 1.05 eV and 2.67 eV, respectively.

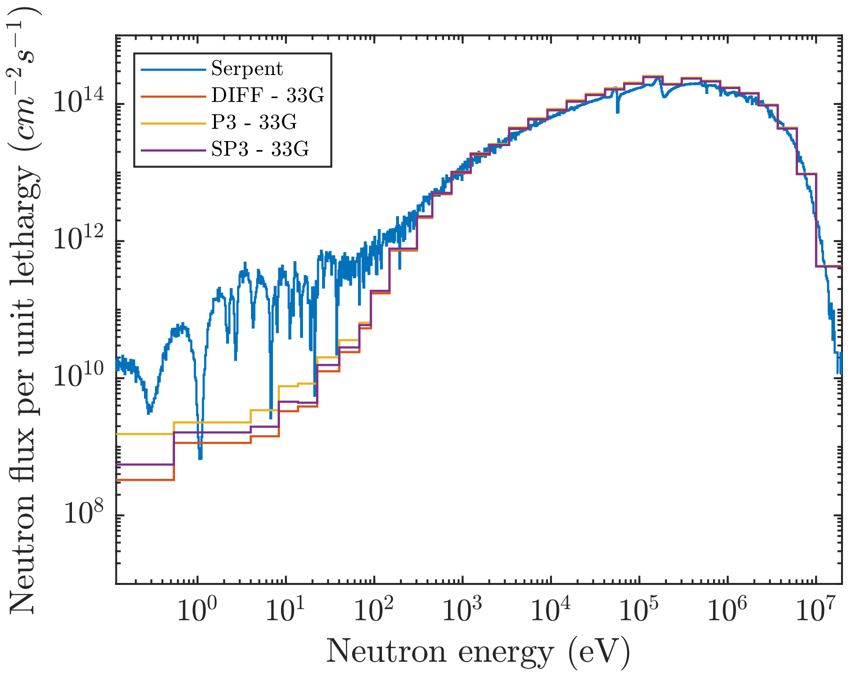


Figure 2.10: Neutron spectrum at the center and midplane of the active core.

2.3.3. Neutron flux and power distributions

Figure 2.11 shows the axial neutron flux distribution (in $cm^{-2}s^{-1}$) at the center of the reactor core for all simulated models. The plot represents the flux values for the 58 bins (number of axial mesh intervals) defined axially from the bottom to the top of the core (295 cm). Good agreement was obtained between both simulation codes, mainly for transport calculations. The values obtained in the center of the active core with the diffusion and simplified transport calculations are slightly lower than those obtained for the reference model. The distribution is not symmetrical, due to the axial non-symmetry of the core, since the upper and lower fission gas plenums have a different thickness (see axial layout of the core in Figure 2.9). This causes a hardening of the distribution at the top because the distance traveled by the scattered neutron from the reflector to the active core is shorter (Lima-Reinaldo & François, 2021).

Figure 2.12 shows the change of the neutron flux (in $cm^{-2}s^{-1}$) along the radial direction at the active core midplane. In general, there is good agreement between the results using both simulation codes, mainly between the reference model in Serpent and those obtained for the transport calculations in ERANOS. Although in the diffusion and simplified transport calculations the flux values in the center of the core are slightly lower (like the axial flux distribution). In the center of the core there is a depression of the neutron flux due to the central control rod position; a similar behavior occurs in the next position (see the core layout in Figure 2.3).

The reactor core power distribution was also determined; Figure 2.13 shows the power peaking per assembly for the reference model and model 5. As can be seen, there is good agreement between the distributions obtained and the peak does not exceed the value of 1.25 in both models.

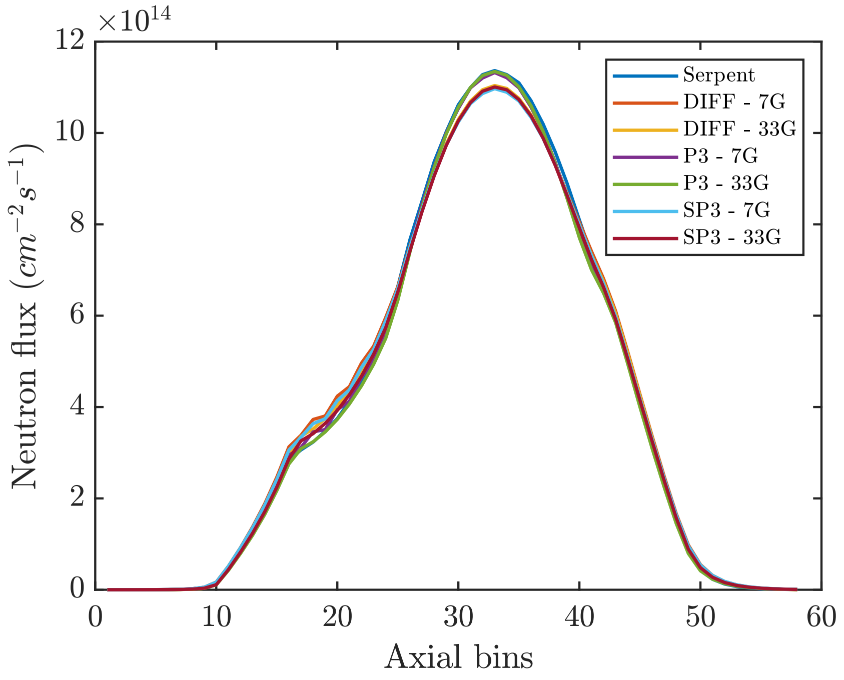


Figure 2.11: Axial neutron flux distribution in the center of the reactor core.

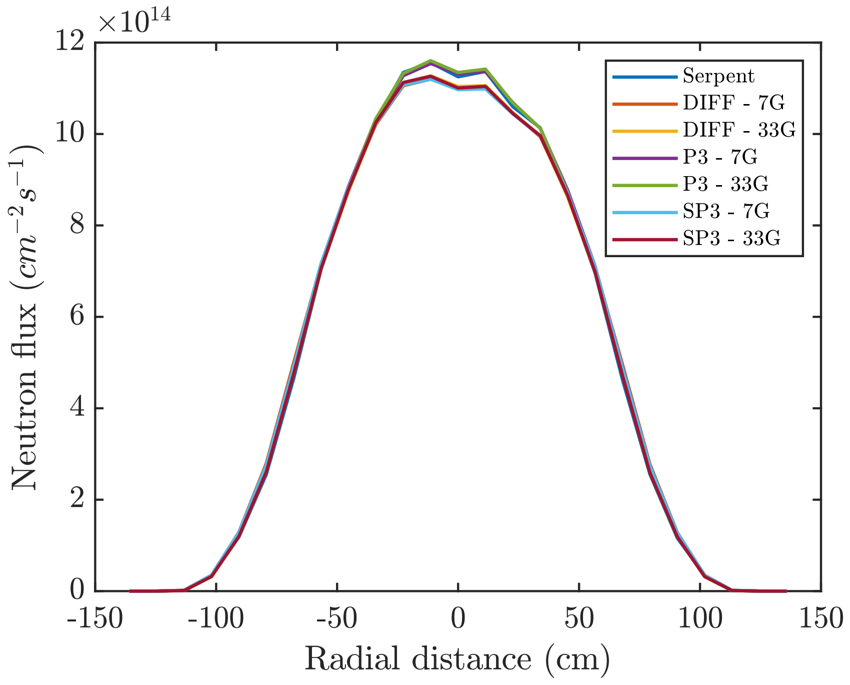


Figure 2.12: Radial change of neutron flux distribution at the active core midplane.

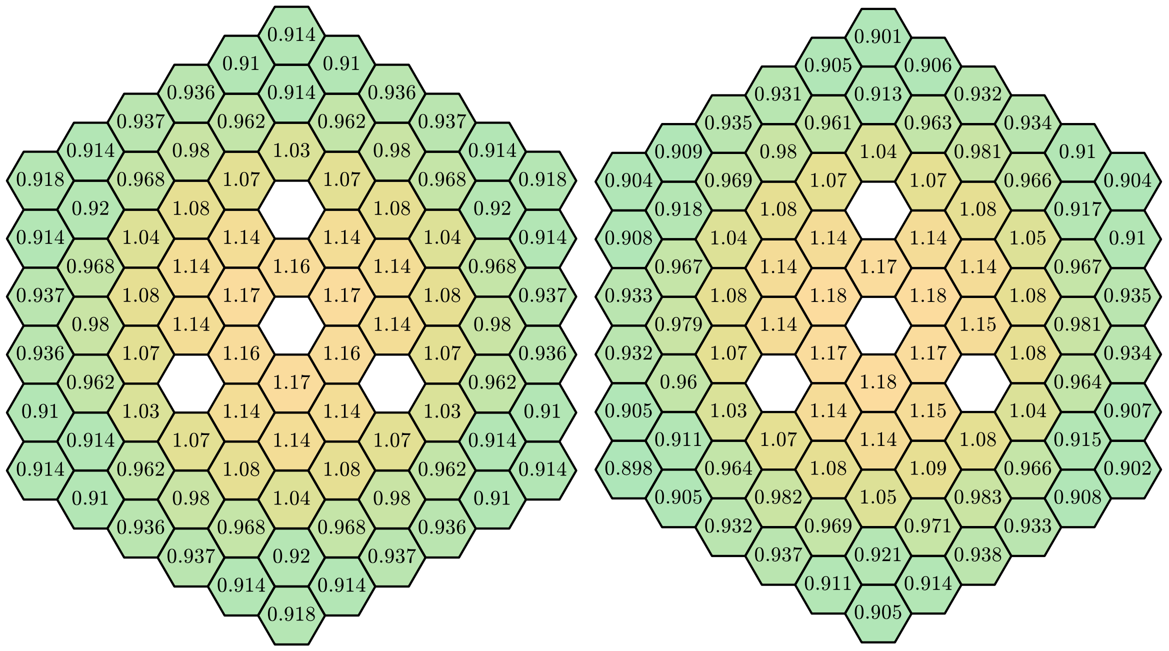


Figure 2.13: Power peaking per assembly for the reference model and model 5.

2.3.4. Doppler effect

An increase in fuel temperature produces a broadening of the effective resonance absorption (and fission) cross section, generally leading to an increase in neutron absorption and a corresponding reduction in reactivity (Lima-Reinaldo & François, 2021). Figure 2.14 shows the Doppler broadening of a radiative capture resonance of ${}^{240}Pu$ at 300, 900 and 1200 K simulated in Serpent. This effect, known as the Doppler effect, provides prompt negative reactivity feedback in fast reactors given mainly to the parasitic capture of low energy neutrons in the fertile isotope (${}^{238}U$) (Stacey, 2018; Waltar et al., 2012). For a uniform change in fuel temperature from the T_1 to T_2, the Doppler reactivity effect in fast reactors is characterized by the Doppler constant (K_D) defined by Equation (2.7) (Waltar et al., 2012):

$$K_D = \frac{\Delta\rho}{\ln\left(\frac{T_2}{T_1}\right)} \tag{2.7}$$

where $\Delta\rho$ (and $\rho = 1 - 1/k_{eff}$) is the reactivity change corresponding to this temperature variation. The K_D value was obtained from two successive calculations with effective cross sections at temperatures T_1 and T_2. In the case of the Serpent code, it uses a built-in Doppler broadening processor routine to adjust the nuclide temperatures (Leppänen et al., 2015). First, an increase in fuel temperature by 100 K from the nominal temperature of 1200 K to 1300 K was assumed. Considering 2200 K as the maximum temperature that prevents cladding failure (Čerba et al., 2017), in a second calculation the fuel temperature was increased from 1200 K to 2200 K. Table 2.9 summarizes the K_D results obtained for the assumed changes in fuel temperature.

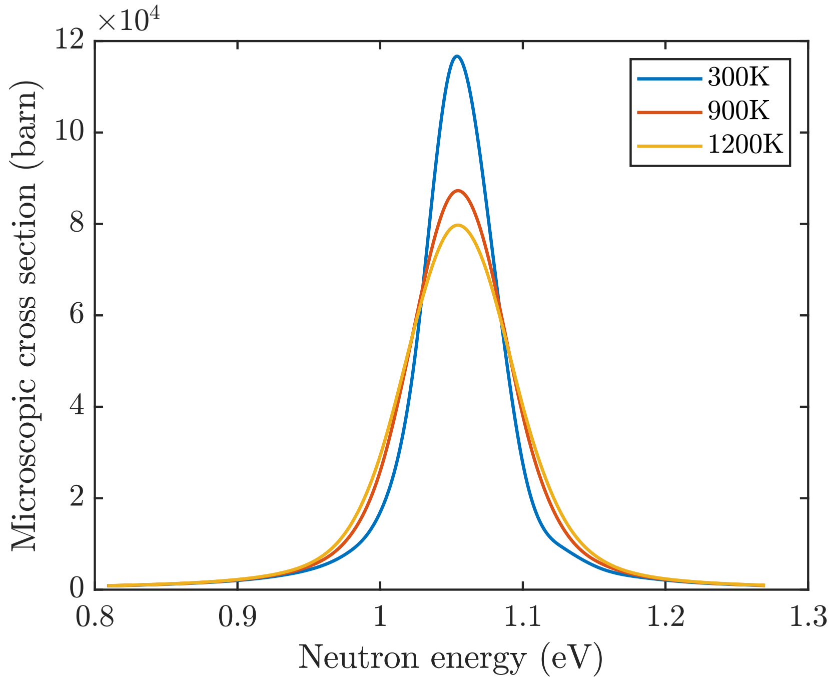


Figure 2.14: Doppler broadening of a radiative capture resonance of ^{240}Pu.

Table 2.9: Doppler constant K_D results.

Model	+100 K		+1000 K	
	Δk-eff (pcm)	K_D (pcm)	Δk-eff (pcm)	K_D (pcm)
1	-57.0	-627.6	-355.0	-517.6
2	-47.1	-538.3	-361.5	-547.1
3	-50.0	-583.0	-380.4	-587.6
4	-47.1	-516.3	-360.4	-523.5
5	-50.0	-557.8	-379.6	-561.3
6	-46.9	-528.5	-359.4	-536.5
7	-49.8	-571.1	-378.5	-575.3

In general, good agreement was obtained between the results, the maximum difference was -111 pcm for the 100 K variation. This difference is mainly due to the Doppler broadening treatment in both codes (Lima-Reinaldo & François, 2021).

2.3.5. Effect of helium density on reactivity

To investigate the influence of helium density on reactivity, the pressure was varied from its nominal value (7 MPa) to atmospheric conditions (0.1 MPa), assuming total depressurization of the core. Figure 2.15 shows the change in reactivity as a function of helium pressure (in MPa). In Serpent, for a total depressurization of the core, an increase in reactivity of 88.8 pcm was obtained, while for the transport and simplified transport calculations, 71.8 pcm and 5.5 pcm, respectively. With simplified transport, although the effect is positive, it is much smaller than in the reference model. In the diffusion calculations, due to the decrease in helium density with decreasing pressure, the effect on reactivity is negative (Lima-Reinaldo & François, 2021). This is because to what was discussed in Subsection 2.3.1 about the limitations of the diffusion approximation in low density regions (overestimation of leakage and neuron streaming effect).

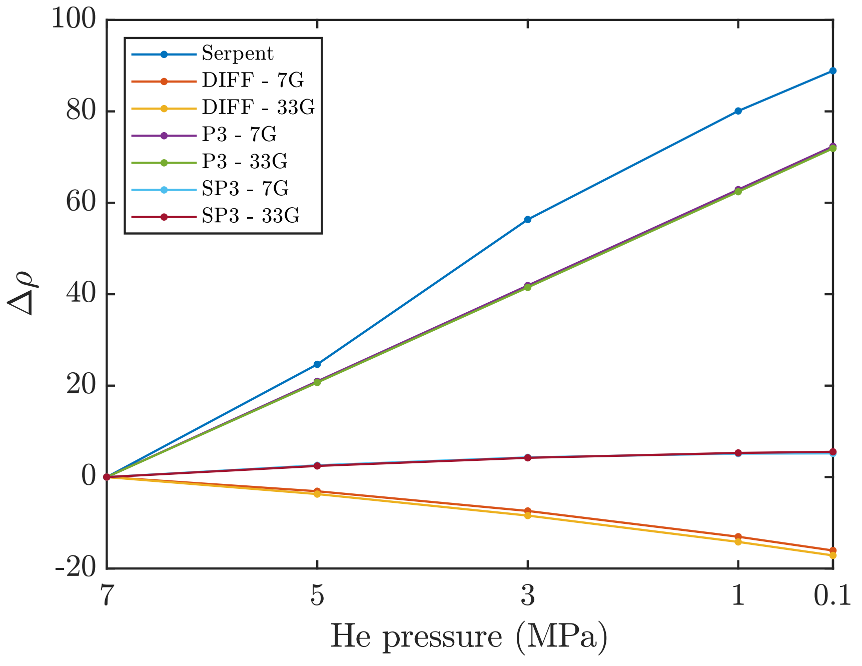


Figure 2.15: Reactivity variation as a function of helium pressure.

Also, to quantify the effect of the control rod followers which are filled with helium, a calculation was performed without control rods. The positions are replaced by the assembly corresponding to the core region. Table 2.10 summarizes the results obtained and shows that the relative difference between the calculation methods is smaller than those shown in Table 2.8. This better agreement in the results is because the diffusion and transport calculations are not affected by the large helium gas channels corresponding to the control rods and rod followers.

Table 2.10: k-eff results for the reactor core model without control rods.

Model	Geometry		Calculation method	Energy groups	k-eff value	Relative difference (%)
	Fuel	Structure				
1	Hetero.	Homo.	Serpent	–	1.10428 ± 1.03E-04	–
2	Homo.	Homo.	Diffusion	7	1.10503	-1.84
3	Homo.	Homo.	Diffusion	33	1.09365	-2.82
4	Homo.	Homo.	P_3	7	1.11212	0.20
5	Homo.	Homo.	P_3	33	1.10185	-0.68
6	Homo.	Homo.	SP_3	7	1.11085	-1.16
7	Homo.	Homo.	SP_3	33	1.10041	-2.04

2.3.6. Effective delayed neutron fraction (β-eff)

In Serpent, β-eff is calculated using the iterated fission probability (IFP) method (Leppänen et al., 2015), while in ERANOS an available procedure was used to compute this parameter. In this procedure provided with the software package, the flux calculation can only be performed through diffusion method. As can be seen, a good agreement was obtained between the results with both codes despite the different calculation methods.

Table 2.11: β-eff results.

Model	β-eff value (pcm)
1	362.7
2	364.4
3	360.9

2.3.7. Burnup calculation

The core evolution in ERANOS is treated by solving the Bateman equations using a constant average flux per region for a given depletion time. The calculation procedure used starts by computing the change in isotope concentration, then the macroscopic cross sections are recalculated and the initial reactor concentrations are updated at each time step. The recalculation of the flux at each step is performed using the TGV/VARIANT module (Lima-Reinaldo & François, 2021).

The burnup calculation in Serpent is performed using built-in calculation routines and without coupling to external solvers which simplifies the use of this capability. Available burnup algorithms include the conventional Euler and predictor-corrector method with linear interpolation for the corrector calculation and the Bateman equations are solved by default using the Chebyshev Rational Approximation Method (CRAM), an advanced matrix exponential solution (Leppänen, 2015). The second available option is the Transmutation Trajectory Analysis (TTA) method, an advanced version of the linear chain method (Cetnar et al., 2021; Oettingen, 2021; Stanisz et al., 2016).

To calculate the k-eff and fuel isotope mass evolution, an operating time of 365 effective full power days (EFPD) was assumed. Initially, small time steps were defined during the beginning of cycle: steps 1 and 2 of 5 days, steps 3 and 4 of 10 days, step 5 of 43 days and from 6 to 9 larger steps of 73 days. The results were compared with those obtained using only 5 steps at 73-day intervals (Lima-Reinaldo & François, 2021).

Figure 2.16 shows the k-eff evolution for models 1 and 4 using the two time step distributions and as can be seen there are no significant differences throughout the cycle. From this analysis, 5 steps of 73 days were defined, which allowed to reduce the calculation time considerably at a reduced penalty in the precision of the results, mainly in the Serpent calculations.

Figure 2.17 shows the evolution of k-eff as a function of operating time for all models (refer to Subsection 2.3 for model specifications). The best agreement with the reference model was obtained for the transport calculations and the 7-group structure as discussed in Section 2. There is a decrease in reactivity by about 2200 pcm in all cases, but the system remains supercritical at the end of cycle. This decrease is due to fuel depletion and accumulation of fission products.

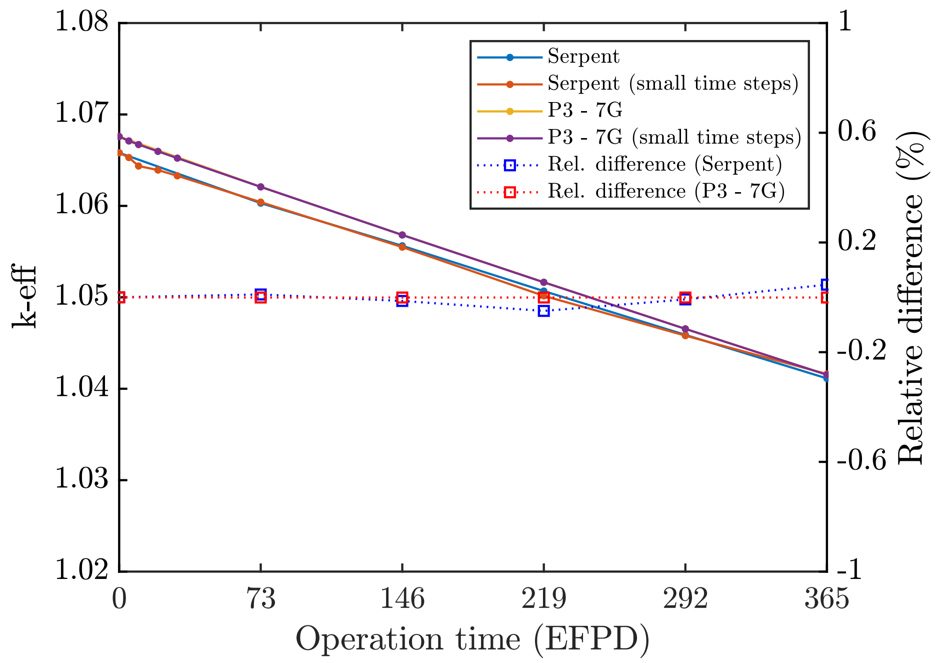


Figure 2.16: k-eff evolution using small time steps at the beginning of cycle.

The running time for each model is summarized in Table 2.12 and, as shown, using the 7-group structure the computational cost is considerably reduced compared to Serpent and the transport calculations with the 33-group structure.

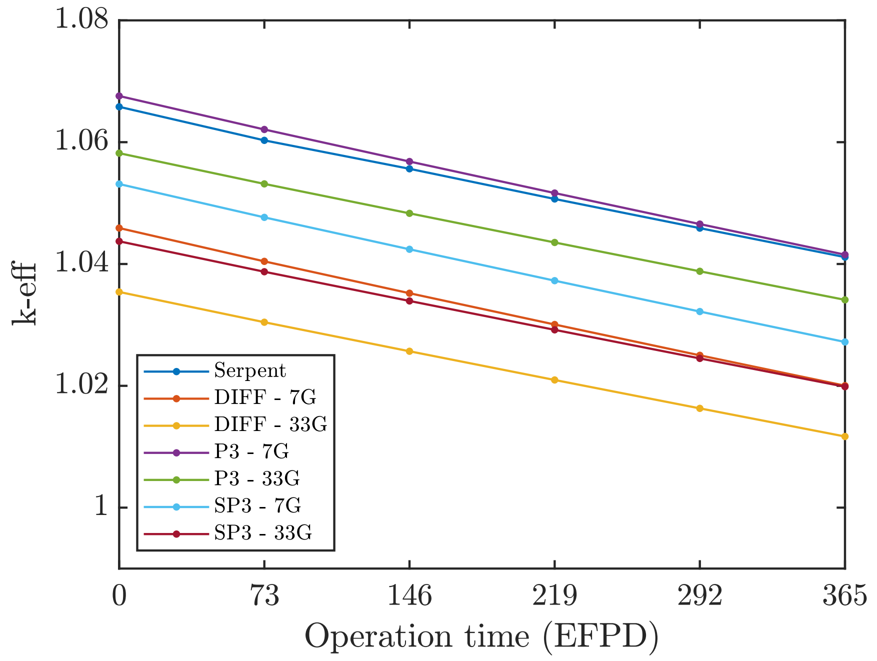


Figure 2.17: k-eff evolution.

Figure 2.18 shows the mass evolution (in kg) of the main fuel isotopes (^{238}U, ^{239}Pu, ^{241}Pu, ^{241}Am) as a function of operating time. The results obtained show an excellent agreement between the cases simulated with both codes considering the different burnup calculation methods. The consumption of the fertile material, ^{238}U, and at the same time of fissile isotopes of ^{239}Pu and ^{241}Pu can be observed. The production of minor actinides is dominated by ^{241}Am produced from the β-decay of ^{241}Pu and is also present in the initial fuel composition.

Table 2.12: Running time in the burnup calculation.

Model	Running time
1	13.4 h
2	1.0 min
3	4.3 min
4	17.7 min
5	2.0 h
6	4.6 min
7	44.8 min

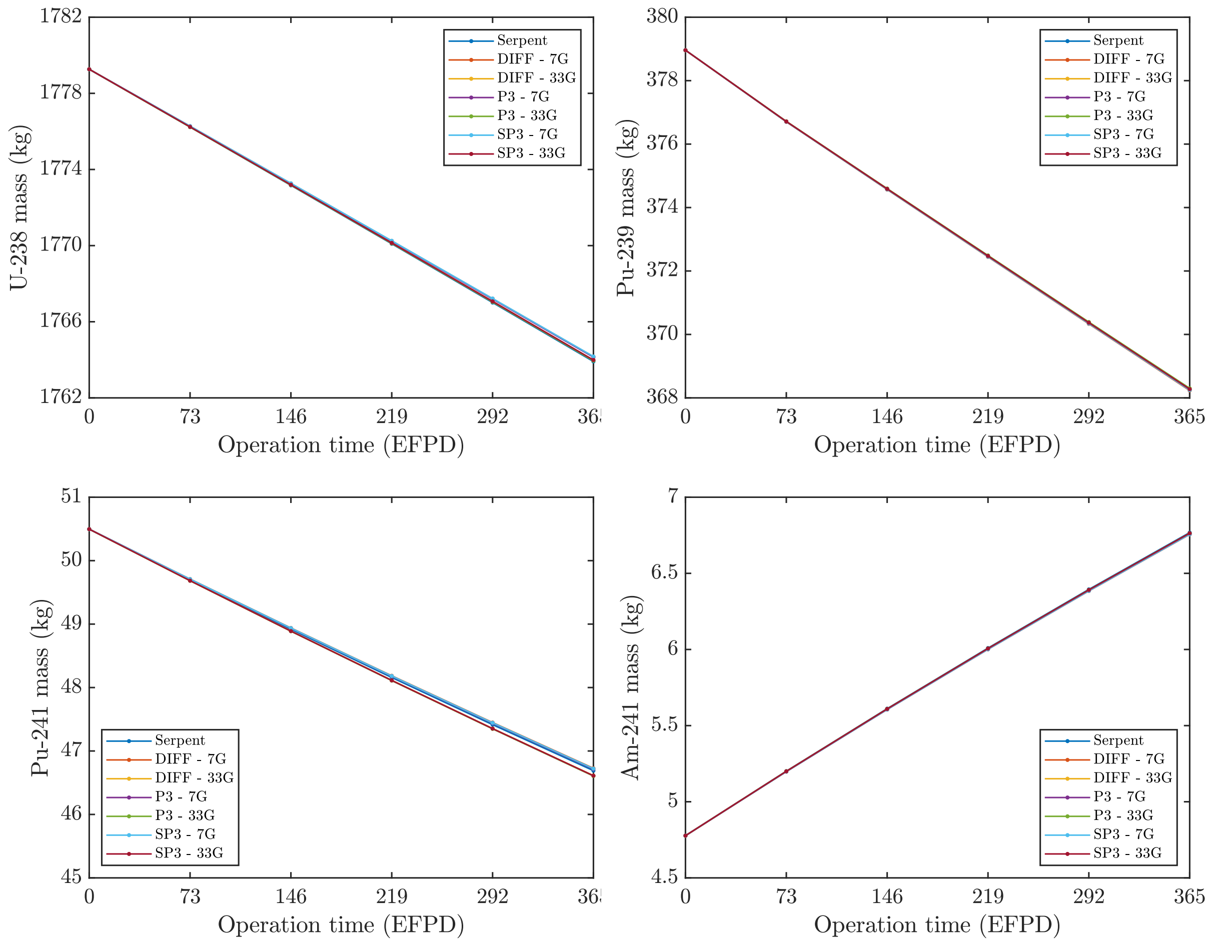


Figure 2.18: Mass evolution of the main fuel isotopes.

Table 2.13 summarizes the fuel inventory relative to the beginning of cycle (BOC) and end of cycle (EOC) for models 1, 4 and 5 (Serpent and transport calculations). The total Pu mass shows a decrease of 0.6% in all cases, dominated by the change in the mass of ^{239}Pu, since it represents 56% of the initial Pu vector. At the end of cycle, the mass of ^{239}Pu shows a decrease of 0.43%. In addition, the breeding ratio (BR), calculated as the ratio between the total mass of fissile material (^{235}U, ^{239}Pu and ^{241}Pu masses) at the end and at the beginning of cycle, is presented. This value of $BR < 1$ confirms that the system is a burner reactor. The average fuel burnup reached 11.08 $MWd/kgHM$ at the end of the cycle.

Table 2.13: Relative fuel inventory.

Element	Serpent (Model 1)		ERANOS (Model 4)		ERANOS (Model 5)	
	Loaded (m/m%)	Discharged (m/m%)	Loaded (m/m%)	Discharged (m/m%)	Loaded (m/m%)	Discharged (m/m%)
U total	72.54	71.91	72.54	71.92	72.54	71.91
^{234}U	0.00	0.01	0.00	0.01	0.00	0.01
^{235}U	0.52	0.48	0.52	0.48	0.52	0.48
^{236}U	0.00	0.01	0.00	0.01	0.00	0.01
^{238}U	72.02	71.41	72.02	71.42	72.02	71.41
Pu total	27.26	26.66	27.26	26.64	27.26	26.66
^{238}Pu	0.74	0.70	0.74	0.7	0.74	0.70
^{239}Pu	15.34	14.91	15.34	14.91	15.34	14.91
^{240}Pu	7.12	7.15	7.12	7.14	7.12	7.15
^{241}Pu	2.04	1.89	2.04	1.89	2.04	1.89
^{242}Pu	2.02	2.01	2.02	2.00	2.02	2.01
MA total	0.19	0.31	0.19	0.30	0.19	0.30
^{237}Np	0.00	0.01	0.00	0.00	0.00	0.00
^{239}Np	0.00	0.00	0.00	0.00	0.00	0.00
^{241}Am	0.19	0.27	0.19	0.27	0.19	0.27
^{243}Am	0.00	0.03	0.00	0.03	0.00	0.03
^{242}Cm	0.00	0.00	0.00	0.00	0.00	0.00
^{244}Cm	0.00	0.00	0.00	0.00	0.00	0.00
FP	0.00	1.13	0.00	1.12	0.00	1.12
Total	100.00	100.00	100.00	100.00	100.00	100.00
k-eff	1.065480	1.041160	1.067585	1.041530	1.058196	1.034117
BR	–	0.9654	–	0.9653	–	0.9655
Burnup (MWd/kgHM)	0	11.081	0	11.082	0	11.082

Notes: 1- Values equal to zero are less than 1E-3 %. 2- m/m% represents the mass percentage.

2.3.8. Equilibrium cycle calculation

Since there are no studies on the ALLEGRO reactor fuel cycle, some features presented in the study by (Krepel et al., 2010) were adopted to design the refueling and reshuffling schema. This is an advanced benchmark study of the GFR2400 reactor fuel cycle using an ERANOS-based in-house developed procedure for fast reactor equilibrium cycle analysis.

The equilibrium can be obtained by applying a fixed in-core fuel management throughout several consecutive operating cycles until the reactor core characteristics (k-eff, power distribution, materials composition, etc.) converge and remain constant cycle-by-cycle. Then, this reactor core composition will be used later in the optimization calculations.

An open fuel cycle was simulated, resulting from a periodic operation with a fixed reloading and reshuffling scheme without any recycling until the equilibrium state was reached. Unlike a close cycle, which involves the recycling of spent nuclear fuel, in an open fuel cycle, recycling is not considered in the implemented fuel management scheme.

The developed procedure starts by defining the number of fresh assemblies, their location in the core and isotopic composition. The methodology used in (Krepel et al., 2010) corresponding to a mixed one-batch scheme to determine the fresh fuel positions and design the refueling scheme was employed. A constant reloading fuel pattern is established in which the fresh fuel is always placed in selected positions based on our expertise. Due to the $120°$ symmetry, it was decided to replace $1/3$ of the fuel assemblies with fresh fuel at the beginning of each cycle. The positions will then rotate in a $120°$ symmetrical triangular layout at the end of cycle and the spent fuel is removed from the core, i.e., the inserted fuel will remain in the core for three cycles. This procedure is simulated over 11 consecutives cycles, with a cycle length of 365 days (11×365 days).

Figure 2.19 illustrates the defined scheme. The red color represents fresh fuel, while the purple and green colors represent the first and second positions for reshuffling. The assemblies are labeled to represent their ordered number and position according to the ERANOS coordinate system. Note that the control rods are not included in the count. The core is divided into 5 concentric rings and 3 sections with 120° symmetry. In each ring, 1/3 of the positions correspond to the fresh fuel (in red) to be inserted. It was considered not to place more than two assemblies contiguously to avoid high power peaks and to preserve the uniformity of the power distribution. The other 2/3 of the assemblies correspond to the next two positions (in purple and green) for the following two cycles. The reshuffling is nothing more than a 120° rotation of the positions between sections. For example, if fresh fuel is inserted in section 1, in the next cycle it will be rotated to the defined position in section 2 or 3. The direction of rotation is represented by arrows in Figure 2.19, i.e., $red \rightarrow purple \rightarrow green \rightarrow red$.

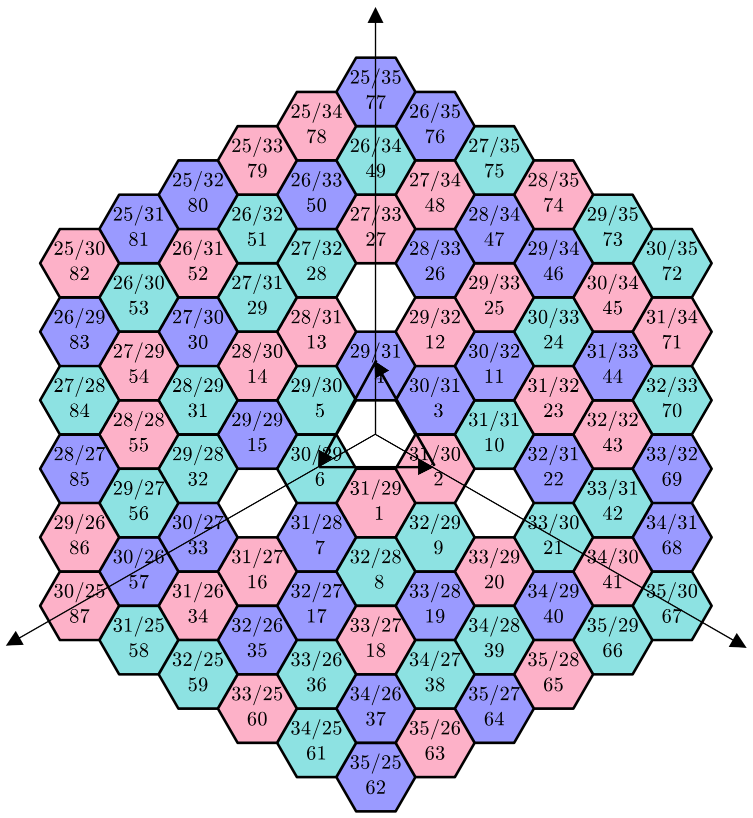


Figure 2.19: Reloading and reshuffling schema. The arrows in the center indicate the direction of the 120-degree rotational movement. The X/Y coordinate and the corresponding assembly number are specified.

In practice, the described procedure, although simplified, is quite complex to carry out in ERANOS. The latest version, ERANOS 2.3, has modules and in-built subroutines to perform a detailed core follow-up, where each assembly can be tracked throughout its lifetime. The proper use of these modules is complicated due to the number of input options and the limited documentation. The following modules were employed:

1. $new\_sub\_assembly\_creation$
2. $reactor\_concentration\_modification$
3. $concentration\_set\_modification\_for\_loading\_drawing$

Taking advantage of the modular structure of ERANOS and these modules, the procedure shown in the flowchart in Figure 2.20 was designed. The algorithm starts by defining and storing all ERANOS calculation procedures that will be used in the simulation. Those procedures include the nodal flux calculation using VARIANT, burnup, flux and power extraction, material balance and the reloading and reshuffling procedure. The latter was generated by combining the three modules listed above. First, with module 1, the new individual assembly composition is defined. Subsequently, with module 2 and using the new assembly composition vector, the reloading is procedure is generated. The reloading is nothing more than exchanging the compositions in the defined positions. Finally, using module 3, the reshuffling procedure or the rotation between compositions (according to the defined positions in Figure 2.19) is generated.

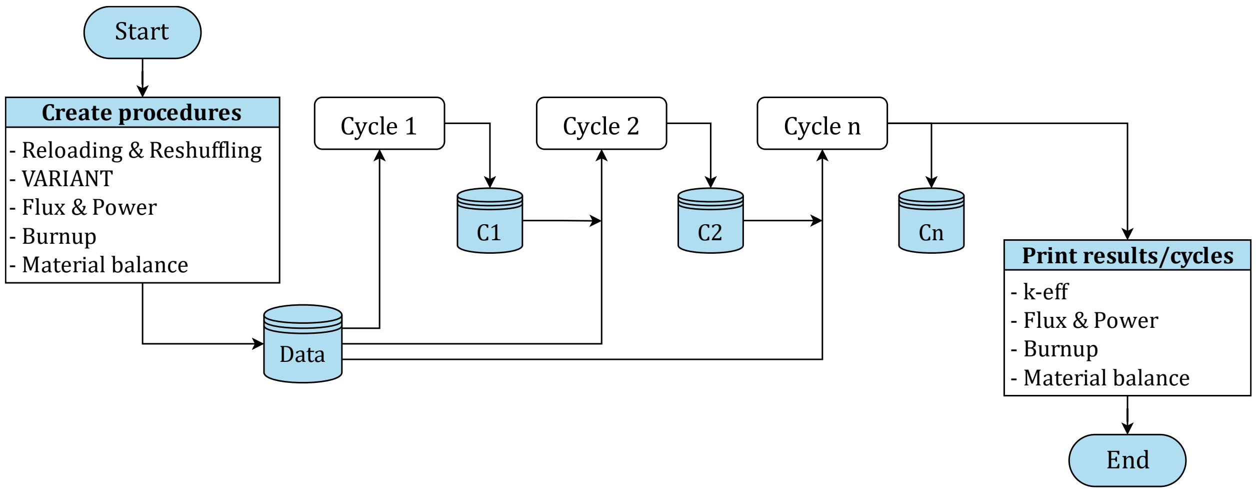


Figure 2.20: Flow diagram corresponding to the procedure developed in ERANOS for the cycle-by-cycle reactor evolution.

Once the first step has been executed, all calculation procedures are stored in the $Data$ file. The algorithm can simulate as many cycles as defined. At the end of each cycle, the results (from running the calculation procedures) are stored in a separate binary file (C_n). The results are loaded in the subsequent cycle execution, and so on, until the defined number of cycles is reached. Finally, the results are extracted and processed to obtain the reactor parameters evolution throughout the n cycles. The algorithm was developed in MATLAB programming environment, along with bash scripting to deal with the ERANOS output files.

The fresh fuel composition (in kg) given in Table 2.14 was calculated using the Equation (2.4) and the isotopic composition summarized in Table 2.2.

Table 2.14: Fresh fuel composition.

Element	Mass (kg)
^{235}U	0.147
^{238}U	20.536
^{238}Pu	0.207
^{239}Pu	4.309
^{240}Pu	2.001
^{241}Pu	0.574
^{242}Pu	0.569
^{241}Am	0.054

In summary, the following assumptions are made:

- 11 cycles are considered, each with a length of 365 days (4015 days) and 6 burnup steps;
- Diffusion calculations and 7 energy groups are used.

The evolution of k-eff, burnup, mass of the main isotopes and power are reported. As shown in Figure 2.21, k-eff converges approximately after the 5th cycle. The zigzag behavior is a result of the addition of fresh fuel as external feed at the beginning of each new cycle. The same behavior is present in the plotted average burnup (in $MWd/kgHM$).

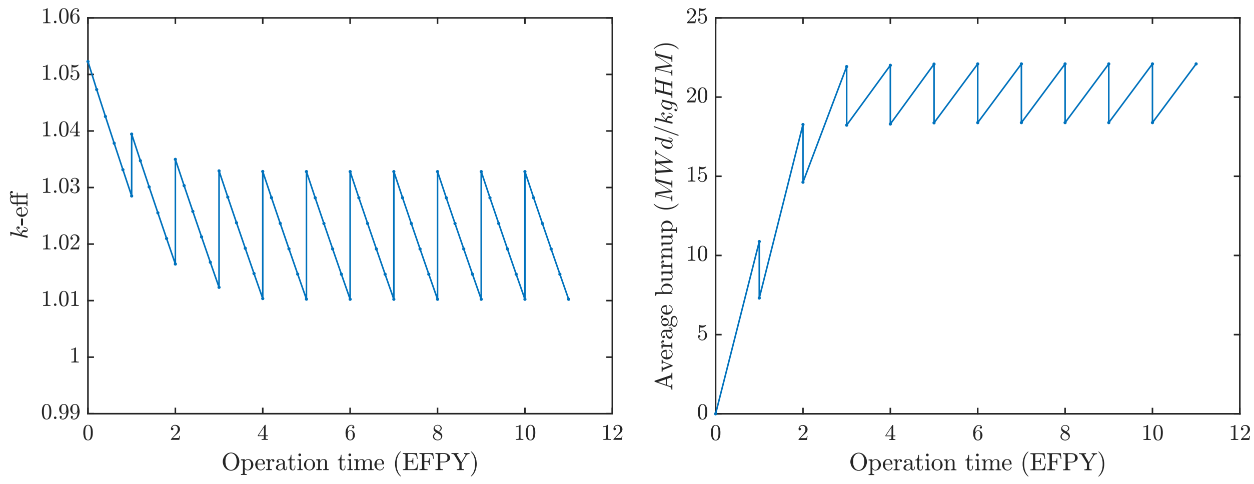


Figure 2.21: k-eff and average burnup values evolution cycle-by-cycle.

Figure 2.22 shows the ^{238}U mass evolution (in kg). The value at BOC remains constant at 1770.9 kg from cycle 4 onwards. In addition, Figure 2.22 shows the plutonium vector evolution (^{238}Pu, ^{239}Pu, ^{240}Pu, ^{241}Pu, and ^{242}Pu). ^{239}Pu predominates, which converges from 5th cycle (~ 355kg at BOC). The mass of the other Pu isotopes practically remains constant throughout the reactor operation.

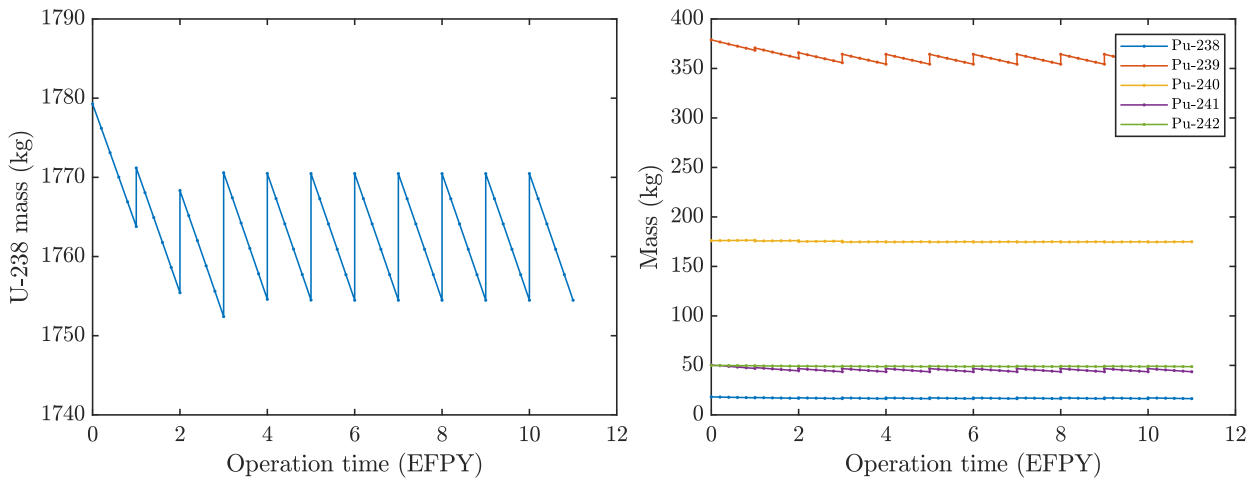


Figure 2.22: Evolution of the masses of ^{238}U and Pu vector cycle-by-cycle.

Figure 2.23 shows the ^{241}Am and total minor actinides (MA: Am + Np + Cm) evolution cycle-by-cycle. The MA evolution in the core is driven by the ^{241}Am production from the radioactive decay (β^-) of ^{241}Pu. It is also present in the fissile fuel vector. A smaller amount of ^{243}Am is produced from the radioactive decay (β^-) of ^{243}Pu which is produced by radiative capture of ^{242}Pu. The total mass of MA, 10.2 kg, remains constant throughout reactor operation. The evolution of other MA (^{237}Np, ^{239}Np, ^{242}Cm, and ^{244}Cm) was also plotted. Among the latter, the mass increase of ^{237}Np is predominant due to the ^{235}U in the fresh fuel (generated by successive reactions of radiative capture of ^{235}U and radioactive decay (β^-) of ^{237}U).

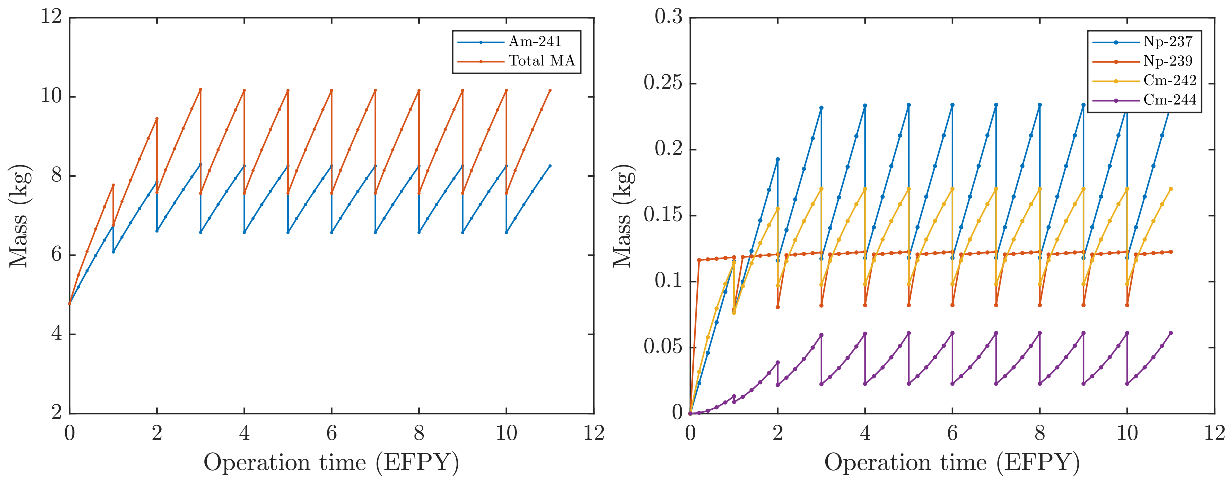


Figure 2.23: Evolution of the masses of Am, total MA, Np, Cm cycle-by-cycle.

Table 2.15 compares the fuel inventories at the BOC and EOC for the first and the equilibrium cycles. The values are reported as a percentage of the total fuel mass in the core. The equilibrium state is reached at cycle 5, since the concentrations, k-eff value, and power distribution do not change from cycle to cycle. At the BOC of the equilibrium cycle the core composition is characterized by a 26.39% Pu content. Since the volume fraction of PuC in the fresh fuel is lower than in the first cycle, the Pu content in the core decreases by 0.88%. The total mass of MA at the equilibrium cycle BOC (characterized by ^{241}Am) shows an increase of 0.11% with respect to the first cycle. The fission products (FP) fraction was also calculated, and there is a 1.1% increase at the EOC of the equilibrium cycle due to the FPs accumulation over the operating time.

Table 2.15: Relative fuel inventory for the first and equilibrium cycles.

Element	First cycle		Equilibrium cycle	
	BOC	EOC	BOC	EOC
U total	72.54	71.89	72.18	71.52
Pu total	27.27	26.67	26.39	25.83
^{238}Pu	0.74	0.70	0.70	0.66
^{239}Pu	15.34	14.91	14.75	14.35
^{240}Pu	7.12	7.14	7.07	7.08
^{241}Pu	2.04	1.90	1.89	1.77
^{242}Pu	2.02	2.01	1.99	1.97
MA total	0.19	0.31	0.30	0.41
^{241}Am	0.19	0.27	0.27	0.33
^{242m}Am	0.00	1.75E-03	1.96E-03	4.05E-03
^{243}Am	0.00	2.49E-02	2.50E-02	4.93E-02
^{237}Np	0.00	4.67E-03	4.53E-03	9.22E-03
^{239}Np	0.00	4.79E-03	3.31E-03	4.96E-03
^{242}Cm	0.00	4.64E-03	4.05E-03	6.86E-03
^{243}Cm	0.00	3.22E-05	4.81E-05	1.12E-04
^{244}Cm	0.00	5.31E-04	9.03E-04	2.44E-03
^{245}Cm	0.00	4.82E-06	1.46E-05	5.32E-05
FP	0.00	1.122	1.119	2.239

Figure 2.24 shows the power distribution for the initial four cycles, and it was noticed that the distribution uniformity deteriorates near the control rods when the reloading procedure begins. The primary reason for this is that the arrangement of fresh fuel is not symmetrical (see Figure 2.19).

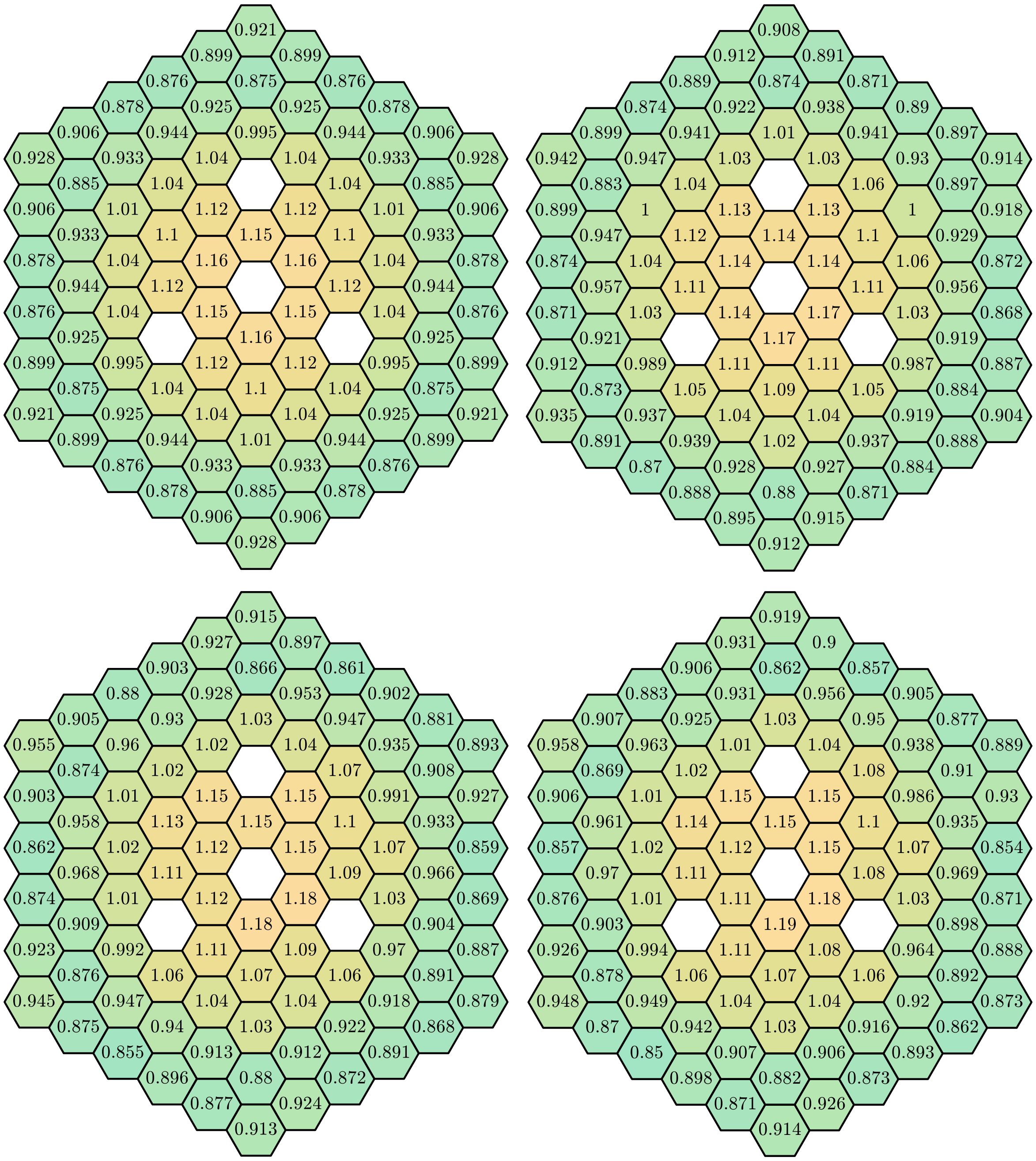


Figure 2.24: Radial power distribution. From left to right, cycle 1, 2, 3, and 4.

Chapter 3

3. Fuel Loading Pattern Optimization of ALLEGRO Reactor by Metaheuristics

This chapter briefly introduces the fundamentals of optimization focusing on metaheuristic techniques. The main advantages of these techniques over traditional methods in solving complex combinatorial optimization problems are presented. Finally, the implementation of genetic algorithm and tabu search techniques for the ALLEGRO fuel load pattern optimization is discussed.

3.1. Fundamentals of Optimization

Optimization is a branch of applied mathematics and numerical analysis. It refers to the process of finding the optimal or near-optimal solution among a set of feasible solutions. Any problem in engineering, science, economics, and life where certain parameters must be determined to satisfy constraints can be formulated as an optimization or search problem (Du & Swamy, 2016). Mathematically, optimization algorithms minimize an objective function $f(x)$ subject to constraints on design variables and responses (or maximize which is equivalent to minimize $-f(x)$). Constraints can be classified as linear or nonlinear and as inequalities or equalities. A design is considered feasible if it meets with all constraints, and infeasible if it does not meet at least one of the constraints.

A general optimization problem can be formulated as follows (Adams et al., 2022):

$$\begin{aligned} minimize: \quad & f(X) \\ & X \in \Re^n \\ subject\ to: \quad & X_L \leq X \leq X_U \\ & g_L \leq g(X) \leq g_U \\ & h(X) = h_t \\ & a_L \leq A_i X \leq a_U \\ & A_e X = a_t \end{aligned} \tag{3.1}$$

Where:

1. $X = [x_1, x_2, \dots, x_n]$ is an *n*-dimensional vector of real-valued design variables.
2. $X_L$and X_U are the lower and upper bounds, respectively, on the design variables. These limits define the allowable values for the X elements.
3. $g_L \leq g(X) \leq g_U$ are the nonlinear inequality constraints have both lower and upper bounds, g_L and g_U, respectively.
4. $h(X) = h_t$ are the nonlinear equality constraints that have target values specified by h_t.
5. $a_L \leq A_i X \leq a_U$ are the linear inequality constraints. $A_i X$ is a linear system where A_i is the coefficient matrix for the system. a_L and a_u are the lower and upper bound, respectively.
6. $A_e X = a_t$ are the linear equality. $A_e X$ is a linear system where A_e is the coefficient matrix for the system and a_t are the target values.

There are multiple methods for solving that optimization problem that involve iterating over X in some way. The process begins by choosing an initial value for each parameter X, and through a simulation, the response quantities $f(x)$, $g(x)$, and $h(x)$ are computed. Subsequently, using an algorithm, new X values will be generated that will reduce the $f(x)$, the number of infeasibilities, or both.

There are several ways to categorize an optimization problem. The most comprehensive way is the one provided by (Adams et al., 2022), where it is divided into three categories, depending on the type of constraints, the search goal, and the solution method. We could add another category according to the type of design variable, although it is implicit in the search method (Haupt & Haupt, 2004). In summary, we can classify an optimization problem according to:

1. Type of constraints and linearity of the objective and constraint functions:
    - Unconstrained problem;
    - bound-constrained problem: lower and upper bounds on the design variables;
    - linearly constrained problem: linear and bound constraints. It is also called a linear programming (LP) problem. On the other hand, when a quadratic objective function is subject to linear constraints, it is called a quadratic programming (QP) problem;
    - nonlinearly constrained problem: may contain nonlinear, linear, and bound constraints. Also known as nonlinear programming problems (NLP), they are more complex to solve and are predominant in engineering applications;
    - equality constrained problem: all linear and nonlinear constraints are equality constraints;
    - inequality constrained problem: all the linear and nonlinear constraints are inequality constraints.
2. Search goal:
    - local optimization problems: they focus on finding a local minimum/maximum of the objective function in a restricted region or neighborhood of the feasible solution space;
    - global optimization problems: they aim to find the global minimum/maximum of a given objective function over the entire feasible solution space. They are typically non-convex, meaning that the objective function may have multiple local minima/maxima, but only one global minimum/maximum. In general, global optimization will be more computationally expensive than local optimization.
3. Solution method:
    - gradient-based methods: the gradients of the objective functions are calculated to find the direction of improvement. This algorithm is used in many efficient local optimization methods;

- non-gradient based methods: they are useful when function gradients are computationally expensive, inaccurate or do not exist.

4. Type of design variable:
    - continuous: infinite number of possible values;
    - discrete: finite number of possible values. It is also known as combinatorial optimization because the optimal solution consists of a particular combination of variables from the finite set of all possible variables.

A typical engineering design optimization process is shown in Figure 3.1 (Kochenderfer & Wheeler, 2019). The role of the designer is to provide a problem specification that details the parameters, initial design, objectives, and constraints that are to be achieved. An optimization algorithm will be used to automate the process and progressively improve design performance. This optimization process will provide a systematic, logical design procedure.

The final optimal design depends on a proper formulation of the design optimization problem. This process for practical problems is iterative. The following six-step procedure can be defined to formulate a general optimization problem, which will help us to formulate ours (J. S. Arora, 2017):

1. Problem description.
2. Data and information collection.
3. Definition of design variables.
4. Optimization criterion.
5. Formulation of constraints.
6. Solution and model evaluation.

The formulation process begins with the development of a descriptive problem statement that describes the objective and the requirements to be met. The next step is to gather data to define a mathematical model for the problem. In addition, procedures and tools required for design analysis (e.g., simulation code) should be identified at this stage. The next step in the formulation process is to identify a set of variables that describe the system or design variables (see Equation (3.1)).

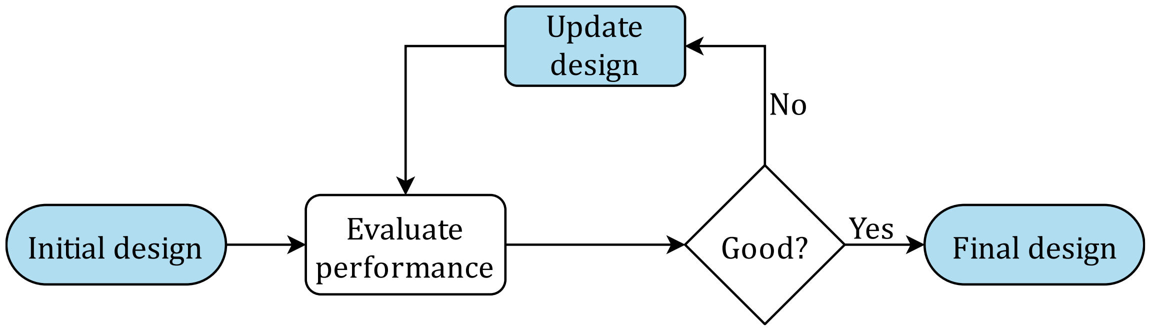


Figure 3.1: Design optimization process.

Different values for the variables produce different designs. The number of independent design variables, also known as design degrees of freedom, gives the number of variables that can be optimized to find the best solution to the problem. It is important to note that optimization becomes increasingly complex as the number of dimensions increases. In the next step a merit of a given design is specified. It is nothing more than a criterion that assigns a number to each design, and it is a function of the design variable vector X. This merit is the objective function $f(X)$, which will be maximized or minimized depending on the problem. Situations where two or more objective functions are identified are called multi-objective optimization problems. For some design problems, it may not be immediately clear what the objective function should be or how it should be expressed in terms of the design variables, and it is often necessary to use some insight and experience to identify an appropriate objective function. Often, to evaluate the objective function a black-box simulation approach is used, where the objective function and constraints are treated as a “black box”, and the optimization algorithm only has access to the inputs and outputs, but not the internal workings of the function (J. S. Arora, 2017). This approach is used in cases where the function is too complex to be represented mathematically, or it must be evaluated by an external simulation code. The next step in the formulation procedure is to identify the constraints and develop an expression for them. Note that these constraints will depend on the design variables. The last step is to solve and evaluate the model by selecting a suitable optimization method. The selection may depend, for example, on the type of design variable (continuous, discrete or integer), whether the function is continuous and differentiable, and whether the derivatives of the function are available. The categorization described above will help in choosing the appropriate technique.

Practical applications of an extensive list of optimization techniques to solve real-world problems or test functions, in programming languages such as MATLAB, Python, Julia, etc. can be consulted in up-to-date references such as: (Adams et al., 2022; J. S. Arora, 2017; R. K. Arora, 2015; Haupt & Haupt, 2004; Kochenderfer & Wheeler, 2019)

3.1.1. Metaheuristics

In real-world optimization scenarios, multiples complexities such as nonconvexity, nonlinearities, discontinuities, high dimensionality, multiple objectives, etc. often render traditional algorithms ineffective, impractical, or inapplicable. To overcome these challenges, heuristic-based search and optimization techniques are a popular choice to find an approximate solution in a reasonable computational time (J. S. Arora, 2017; R. K. Arora, 2015; Bandaru & Deb, 2016; Bianchi et al., 2009; Dréo et al., 2006; Du & Swamy, 2016; Kochenderfer & Wheeler, 2019).

In optimization, a heuristic (it derives from the Greek verb *heuriskein* that means “to find”) is an approximate technique for finding an optimal solution to a complex optimization problem by using an intuitive, trial-and-error approach (Bianchi et al., 2009). It involves making a sequence of choices or decisions based on a set of rules or guidelines, rather than using an exact, systematic method to find the optimal solution. The goal of a heuristic is to produce a feasible solution that is good enough to be useful, even though it may not be the best possible solution. In addition, it is computationally efficient since it does not explore all possible solutions in the search space and usually does not have a rigorous proof of convergence to the optimal solution.

Metaheuristics (the Greek suffix “*meta*” means “*beyond, in an upper level*”) are self-learning optimization techniques that provide a general and flexible framework for solving hard optimization problems (Bianchi et al., 2009; Du & Swamy, 2016). They are characterized by a high level of generality and versatility, allowing them to be applied to a wide range of problem domains. They typically use a combination of heuristics, probabilistic approaches, and other mathematical techniques to search for optimal solutions in large or complex problem spaces. Because they use heuristics as part of the search strategy, they inherit the characteristics described above. They do not guarantee finding the exact optimal solution but can lead to a near-optimal solution in a computationally efficient way.

Most metaheuristic methods are stochastic in nature and mimic a natural, physical, ethology or biological principle that resembles a search or optimization process. Exploration (or diversification) and exploitation (or intensification) are two important concepts in metaheuristics. Exploration helps to find new and diverse solutions in the search space, while exploitation focuses on refining and improving the existing solutions to intensify the search. The balance between exploration and exploitation is crucial in determining the performance of a metaheuristic, and it can be tuned to achieve a trade-off between global search and local refinement, depending on the specific requirements of the problem.

Metaheuristics are usually classified into two main categories (although there are several ways) based on the number of initial solutions: single-solution and population-based (Bandaru & Deb, 2016). The single-solution approach focuses on iteratively modifying and improving a single candidate solution. Population-based metaheuristics maintain and improve (via operators) multiple candidate solutions over iterations. In this case, the size of the population is set by the user.

Among the advantages of these techniques over classical mathematical optimization methods, and why they have become increasingly popular, are the following: (J. S. Arora, 2017; Bandaru & Deb, 2016; Dréo et al., 2006):

1. They do not require gradient information and can therefore be used with non-analytical, black-box or simulation-based objective functions. This feature makes them a flexible tool, relatively easy to use and program for solving a wide range of optimization problems.
2. Most metaheuristics can recover from local optima due to inherent stochasticity or by using heuristics, making them well-suited for global optimization.
3. They are versatile, as many of them can be combined with other optimization algorithms to create even more effective hybrid approaches to solve complex optimization problems.
4. They can find good enough solutions for NP-hard problems (there is no known exact algorithm that can solve them in a reasonable computation time).

5. They can handle problems with multiple objectives and mixed design variables.
6. They are often robust to the presence of noisy or uncertain data, making them ideal for applications where the optimization problem is subject to uncertainty.

However, it is important to acknowledge some of the drawbacks associated with these techniques, including (J. S. Arora, 2017; Bandaru & Deb, 2016):

1. Non-mathematical nature and lack of a proof of convergence. There is no absolute guarantee that a global solution has been obtained due to its stochasticity. It can be overcome to some extent by running the algorithm several times or by increasing the number of iterations.
2. Non-reproducibility of the results. Uniformly seeded pseudo-random number generators are commonly used to ensure reproducibility between different runs.
3. It can be difficult to fine-tune the parameters of the method.
4. High computation time because it requires a large number of evaluations of the objective function. It can be overcome, for example, by using parallelization.

In recent years, the use of metaheuristics in optimization has seen a significant increase in popularity, leading to the proposal of numerous new techniques. This growing number of techniques, which share similar basic principles but differ in the specific metaphorical models or biological systems they are inspired by, has also been criticized.

The following is a brief list of the most popular metaheuristic techniques; for more details on their implementation, please refer to the references provided:

1. Evolutionary Programming (EP) – (Fogel et al., 1966).
2. Evolutionary Strategies (ES) – (Rechenberg, 1973) (Rudolph, 2012).
3. Genetic Algorithms (GA) – (Holland, 1975).
4. Simulated Annealing (SA) – (Kirkpatrick et al., 1983).
5. Tabu Search (TS) – (Glover, 1986).
6. Ant Colony Optimization (ACO) – (Dorigo, 1992) (Marco Dorigo et al., 2006).

7. Particle Swarm Optimization (PSO) – (Kennedy & Eberhart, 1995).
8. Artificial Bee Colony (ABC) – (Karaboga & Basturk, 2007).
9. Firefly Algorithm (FA) – (Yang, 2009).
10. Cuckoo Search (CS) – (Yang & Deb, 2009).

Among the many practical applications of these techniques is the fuel loading pattern optimization in nuclear reactors. This task is recognized as a complex combinatorial optimization problem. The Introduction chapter of this thesis presents an overview of in-core fuel management decisions in nuclear engineering. Despite the continuous emergence of new techniques, tabu search and genetic algorithms have been widely used in this field, either in their basic form or as part of hybrid approaches. The following sections will provide a detailed description of the implementation of these two techniques to the ALLEGRO reactor fuel loading pattern optimization.

3.2. The Objective Function

The constrained optimization with penalty function approach was used, where the multi-objective problem is transformed into a single-objective optimization problem. Since this method is easier to implement, it has been widely used in LP optimization problems. Throughout the research, the objective function (OF) formulation evolved from a single constraint on power peaking factor (PPF) to a more complex one considering other reactor parameters. The final proposed objective function maximizes the k-eff value at the EOC while satisfying the power peaking factor (PPF), the excess reactivity at BOC (HER^{BOC}) and the maximum linear heat generation rate ($MLHGR$) constraints.

The rationale behind this formulation is to find LPs with as high excess reactivity as possible at EOC as to extract more energy to the cycle, meanwhile thermal limits are not exceeded and that the reactor can be controlled at the point of the highest reactivity, which in this case is at BOC. Then, the first optimization problem to be solved can be written as (Lima-Reinaldo & François, 2022):

$$\begin{aligned} OF(x) = k_{eff}^{EOC}(x) - w_h \times max\big(0, \Delta HER(x)\big) - w_p \times max\big(0, \Delta PPF(x)\big) \\ - w_l \times max\big(0, \Delta MLHGR(x)\big) \end{aligned} \tag{3.2}$$

Where:

1. x is a vector corresponding to a particular LP configuration.
2. $k_{eff}^{EOC}(x)$ is the k-eff value at the EOC. From now on, this term will be referred to as k-eff.
3. $\Delta HER(x) = HER^{BOC}(x) - HER_{max}$; where $HER^{BOC} = (k_{eff}^{BOC} - 1) \times 100$ and the HER_{max} = 4.5%. This is a conservative value related to the shutdown capability of the control rods. The worth of the control rods was calculated at the beginning of life and compared with the value obtained in a previous study of the GFR2400 reactor (Lima-Reinaldo et al., 2019).S

4. $\Delta PPF(x) = PPF(x) - PPF_{max}$. The PPF value is taken over the cycle and $PPF_{max} = 1.3$ is the maximum value allowed. This is a conservative value related to 1.4 which is used as a rule of thumb in core design.
5. $\Delta MLHGR(x) = MLHGR(x) - MLHGR_{max}$; where $LHGR$ is calculated as the linear heat generation rate per unit length of fuel rod over the cycle given in W/cm. The maximum value assumed is 330 W/cm (Grasso et al., 2013). This is a conservative value used for MOX fuel from the ELSY project. Compared with mixed oxide fuels, mixed carbide fuels have higher thermal conductivity and higher linear heat rate capability as well.
6. $w_h = 0.02$; $w_p = 0.06$; $w_l = 0.05$ are the weighting factors. These are positive penalty coefficients or weight factors considered as a measure of the importance associated to HER, PPF and $MLHGR$, respectively. They were chosen by evaluating several patterns and computing the objective function. In the experiment, the values were set to zero initially and adjusted to scale the objective function, that is, if one of the constraints was violated, not to get negative values of the objective function. With these factors obtained, if the constraint is violated, a positive value will be introduced and the value of k-eff will be penalized. On the other hand, if the constraint is not violated the weight factor will be zero so no penalty will occur. The latter can be deduced from the formulation given in Equation (3.2), if Δ is less than zero, the $max()$ function will return zero.

During the GA code development, we decided to experiment with other OF formulation to validate the proposed method and investigate its behavior. Since the maximum peak power is in the center of the core, it was considered to add the $U(x)$ term that represents the flatness of the PPF distribution at the BOC to obtain a uniform power distribution. It will also be useful to evaluate the code performance in the search for a feasible solution. This term has been included in several PWR LP optimization studies (Khoshahval et al., 2010; Poursalehi et al., 2013; Tran et al., 2021). Then, the second proposed OF is (Lima-Reinaldo & François, 2021):

$$\begin{aligned} OF_2(x) = w_k \times k_{eff}^{EOC}(x) - w_h \times max\big(0, \Delta HER(x)\big) - w_p \times max\big(0, \Delta PPF(x)\big) \\ - w_l \times max\big(0, \Delta MLHGR(x)\big) - w_f \times U(x) \end{aligned} \tag{3.3}$$

Where:

$$U(x) = \sum_{i=1}^{N} \left| PPF_i^{BOC}(x) - 1 \right| \tag{3.4}$$

And:

1. $N = 87$ is the number of fuel assemblies.
2. $PPF_i^{BOC}(x)$ corresponds to the PPF at the BOC of the i-th fuel assembly.
3. $w_k = 10$; $w_h = 0.2$; $w_p = 0.6$; $w_l = 0.5$; $w_f = 0.3$. These factors were chosen using the same rationale described above in Equation (3.2).

To evaluate the LPs and compute the corresponding OF value, the interface shown in the diagram in Figure 3.2 was implemented. The input and output values of the interface will depend on the algorithm to which it is embedded. In principle, it will allow communication between the optimizer and the external ERANOS simulation code. The data flow is handled by reading and writing text files. First, the output LP vector is read and checked for no more than two contiguous fresh assemblies. According to the formulation described by Equation (3.5), if this condition is false, $I(x) = 0$ is returned, otherwise, the LP vector is converted to the ERANOS coordinate system for the OF evaluation.

$$I(x) = \begin{cases} 0, & if\ (num\_contiguos > 2) \\ OF(x), & otherwise \end{cases} \tag{3.5}$$

Next, the ERANOS input file is written, and the core calculation is executed. Finally, the parameters are passed to the GA optimization code for the OF calculation.

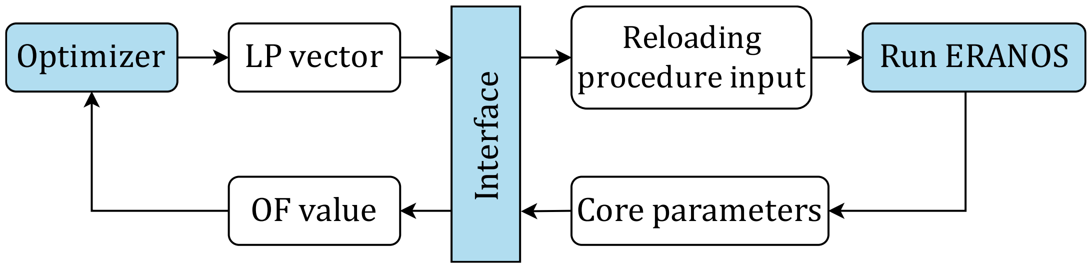


Figure 3.2: Flowchart of the interface between the optimization algorithm and the ERANOS simulation code.

3.3. Genetic Algorithms applied to the ALLEGRO Fuel Loading Pattern Optimization

A genetic algorithm is a derivative-free stochastic global search technique based on biological evolutionary processes proposed by (Holland, 1975) and later described by (Goldberg, 1989). GA is the most popular form of evolutionary algorithm (EA). The basic idea of a GA is to generate a new set of designs, or population, from the current set such that the average OF value of the population is improved. For this purpose, the so-called genetic operators, reproduction, crossover, and mutation, are used. The iterative process will continue until a stopping criterion is met or the number of iterations exceeds a specified limit. There is an extensive list of references on practical applications of genetic algorithms, from basic implementations to advanced variants. The following can offer significant guidance and knowledge for implementing GAs in real-world scenarios (J. S. Arora, 2017; R. K. Arora, 2015; Haupt & Haupt, 2004; Kochenderfer & Wheeler, 2019).

As described in Section 1.1 of Chapter 1, LP optimization is a complicated task considered as a multi-objective combinatorial optimization problem. It has been demonstrated that GA is an efficient metaheuristic technique for dealing with this type of problem, since it does not require any derivative information and covers the search space relatively fast. Although it cannot be guaranteed that they will find an optimal solution, if properly applied, an acceptable solution close to the true optimal solution will be found (Lima-Reinaldo & François, 2022). More advantages were discussed in Subsection 3.1.1. The GA technique has been widely used and accepted in this field, even with the rise of new computational techniques.

For the implementation of GA, the question is where to place the fresh fuel satisfying the operational and safety constraints imposed (see the arrangement in Figure 3.3). Each LP is represented as a vector of 29 non-repeating integers (number of fresh fuel assemblies), where the values correspond to the decision variables or fresh fuel positions p_n in the reactor core, which can take values from 1 to 87 (total number of fuel assemblies). Then, a LP vector can be denoted as:

$$LP = (p_1, p_2, p_3, \dots, p_n);\ n = 29;\ p \in [1,\ 87] \tag{3.6}$$

This encoding will be used in the optimization algorithm for its simplicity but is transformed into the ERANOS coordinate system for the reactor calculation. Figure 3.4 shows an example of three different configurations considering two rings of the hexagonal lattice supposing that one third of the positions will be loaded. Note that the order does not matter, i.e., a vector $LP = (16, 4, 14, 12, 1, 2) = (1, 2, 4, 12, 14, 16)$. Moreover, as can be seen, the constraint imposed on the number of contiguous fresh assemblies is met.

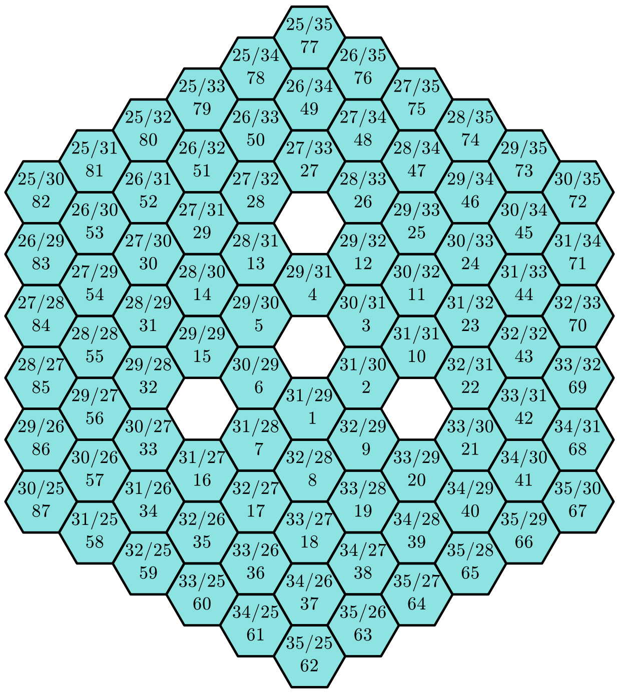


Figure 3.3: Arrangement of fuel assemblies in the core. The blank spaces are the control rods positions. The X/Y coordinate and the corresponding assembly number are specified.

The flow-chart in Figure 3.5 and the pseudocode in Figure 3.6 summarize how the developed GA code works. The process begins by defining the GA parameters such as the number of variables, population size, crossover rate, mutation rate, and the number of generations. The integer encoding scheme, described in Equation (3.6), was used to represent the 29 positions or variables of each LP. Subsequently, the initial population of size pop_size is generated from a feasible LP to accelerate the exploration of the design space.

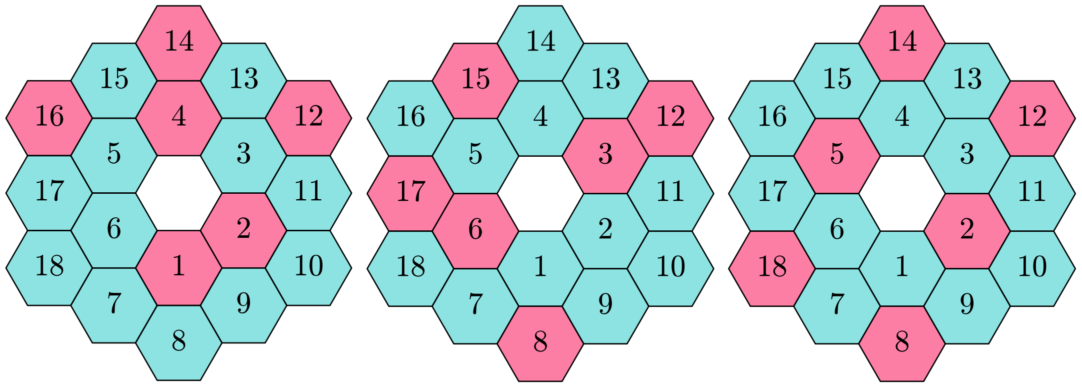


Figure 3.4: Schematic representation of three different LPs vectors (in red) in a two-ring hexagonal lattice example.

The next step is to evaluate the initial population in ERANOS to compute the core parameters and the OF value (see Equation (3.5)). For this purpose, an interface between ERANOS and the GA algorithm was developed, which transforms the LP vector to the ERANOS coordinate system and executes the core calculation. Next, the reproduction phase begins where the selection criteria and the crossover and mutation operators are applied to produce the offspring. First, pairs of different parents are selected by tournament selection. Then, using the crossover operator, the offspring population of size $C = crossover\_rate \times pop\_size$ is produced, and the mutation operator is applied. Subsequently, the population is evaluated, and the next generation is formed by inserting the offspring into the previous population and selecting the best pop_size individuals.

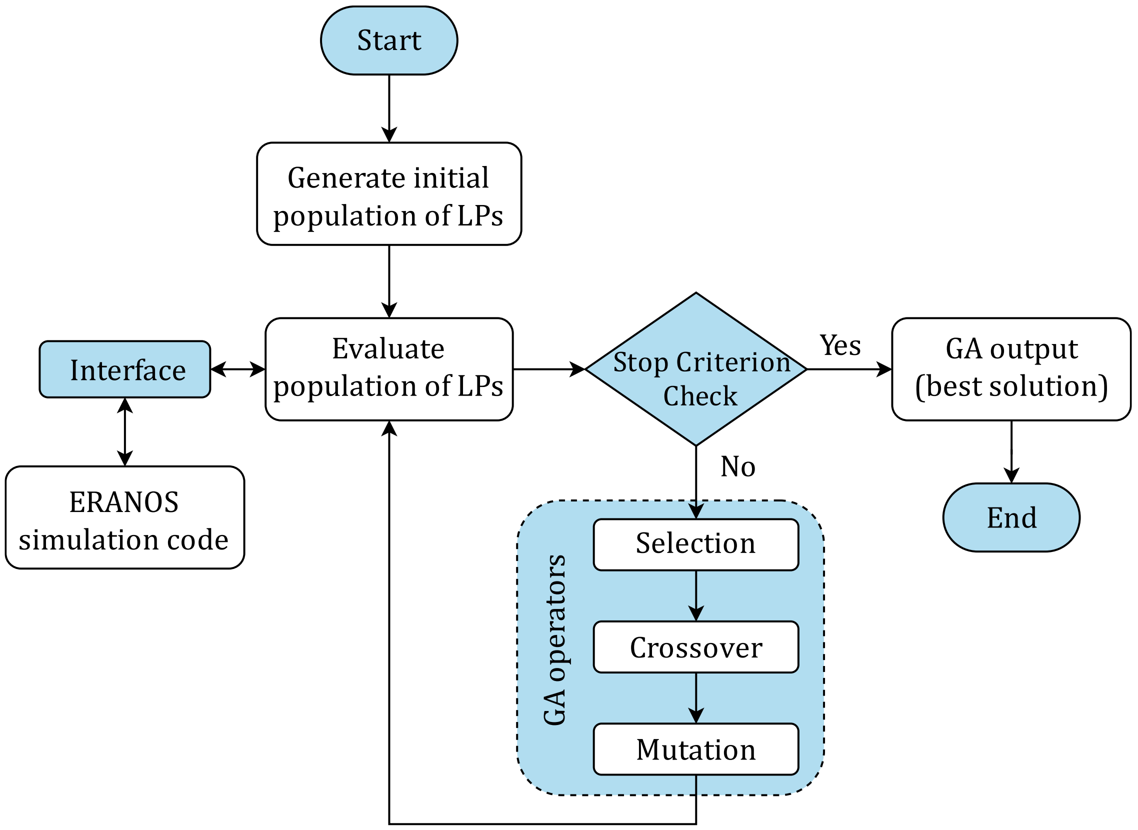


Figure 3.5: Flowchart of a GA.

The algorithm stops when the maximum number of generations is reached. The output is the best solutions over generations, including LPs, OF values and core parameters. The code has additional interesting features. First, an algorithm was developed to detect repeated individuals in each generation or clones and avoid re-evaluations. Secondly, each population is stored, and in case an LP reappears, the already calculated OF value is taken, without executing the ERANOS code. With these two improvements the computational cost is considerably reduced. Third, in the interface module each pattern is checked to ensure that it meets the constraint on the maximum number of contiguous fresh assemblies, if true it is evaluated, otherwise, the OF will be zero (see Equation (3.5)).

```
Initialize GA params (num_vars, pop_size, crossover_rate, mutation_rate, num_gens)
Generate initial population (array dimensions: pop_size × num_vars)
Evaluate initial population (call the interface and objective function):
   pop = evaluate(pop);
   pop = sort(pop);
Set best_sol:
   best_sol = pop(1);
Main loop:
for g = 1:num_gens
   C = crossover_rate × pop_size;
   for i = 1:C/2
      Perform selection:
         P1 = tournament(pop, tournament_size);
         P2 = tournament(pop, tournament_size);
      Perform crossover:
         [O1, O2] = PMX(P1, P2);
         offspring(i, 1) = O1;
         offspring(i, 2) = O2;
   end
   Perform mutation:
      offspring = random_mutation(offspring, mutation_rate);
   Evaluate offspring population:
      offspring = evaluate(offspring);
   Insert offspring into the population and choose best pop_size members
      pop = sort([pop; offspring]);
      pop = pop(1:pop_size);
   Update best solution:
      best_sol(g) = pop(1);
end
Print results (best_sol)
```

Figure 3.6: Pseudocode of the developed GA code.

3.3.1. GA operators

In GA, the population of best individuals evolves in successive generations using selection, crossover, and mutation. The GA algorithm procedure is shown in Figure 3.6. In the selection phase, pairs of parents are selected for reproduction based on their OFs values. For the LP optimization, the commonly used selection operators are roulette wheel selection and tournament selection.

It has been shown that tournament selection has better, or equivalent, convergence properties and computational time when compared to any other selection operator (Deep et al., 2009). Therefore, the tournament selection operator was used. In this method, k individuals are randomly chosen from the population and participate in a tournament where the best individual from this set is selected as the parent. A tournament size value of 4 was used. In addition, a procedure was developed to ensure that the pairs of parents to be crossed were different. After selection, applying the crossover and mutation operators, offspring are produced and added to the next generation.

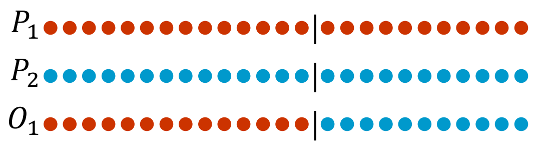
The crossover operator combines the genetic information between individuals to explore the search space. The classical crossover operators, one- and two-point crossover, do not work for the defined optimization problem. The offspring resulting from applying these operators does not represent a valid pattern since they may have repeated fuel assembly positions. Considers a simple GA that uses a binary encoding. The *n*-dimensional solution vector $X = [x_1, x_2, x_3 \dots, x_n]$ (or LP), where x_n is the position value in our problem, is encoded in the form:

$$X = \underbrace{01 \dots 00}_{x_1}\underbrace{11 \dots 10}_{x_2}\underbrace{10 \dots 00}_{x_3} \dots \underbrace{10 \dots 00}_{x_n} \tag{3.7}$$

If the chromosome is *l*-bit long, it has 2^l possible values. If the variable x_n is in the range $[x_i^-, x_i^+]$ with a coding $(s_{li} \dots s_2 s_1)$, where l_i is its bit-length in the chromosome and $s_i \in \{0, 1\}$, then the encoding/decoding function is given by (Du & Swamy, 2016):

$$x_i = x_i^- + \frac{(x_i^+ - x_i^-) \times DV(s_{l_i})}{2^l - 1} \tag{3.8}$$

Where $DV(s_{li}) = \sum_{j=0}^{l_i-1} s_j 2^j$ is the decoded value of the string. Now if we apply the single-point crossover by randomly selecting a crossover point along the string we obtain the following offspring:



(3.9)

Due to the O_1 is formed by the first portion of P_1 and the latter portion of P_2, and decoded by Equation (3.8), some positions will be repeated. Even if we apply other commonly used operators such as multi-point crossover the problem will persist. Another example is to consider the integer vectors P_1 and P_2, and applying the single-point operator generates the child O_1, which is not valid because the position 2 is repeated.

$$\begin{aligned} P_1 &= (7\ \ 3\ \ 8\ \ 2\,|\,1\ \ 4\ \ 5\ \ 6) \\ P_2 &= (1\ \ 3\ \ 5\ \ 6\,|\,4\ \ 8\ \ 7\ \ 2) \\ O_1 &= (7\ \ 3\ \ 8\ \ 2\,|\,4\ \ 8\ \ 7\ \ 2) \end{aligned} \tag{3.10}$$

The initial intention to apply genetic algorithms was to use validated optimization codes such as MATLAB Optimization Toolbox (ga solver) and DAKOTA software (SOGA optimizer). After developing several models that tested different types of crossover operators, it became clear that the same problem described above was occurring.

To address this problem, the partially mapped crossover (PMX) and the order crossover operator (OX), commonly used to solve permutation-based problems such as the traveling salesman problem (TSP), were selected (Hussain et al., 2017; Larrañaga et al., 1999). The results with both operators will be compared to check the algorithm behavior.

The PMX operator works by randomly selecting two crossover points on the parents' strings, which determine the segments of genetic material that will be exchanged between the parents to create the offspring. The genetic material between the two crossover points is mapped from one parent to the other, such that the offspring inherits some genetic material from each parent. Consider, for example the following two parents P_1 and P_2 (or LP vectors):

$$\begin{aligned} P_1 &= (7\ \ 3\ \ 8\ \ 2\ \ 1\ \ 4\ \ 5\ \ 6) \\ P_2 &= (1\ \ 3\ \ 5\ \ 6\ \ 4\ \ 8\ \ 7\ \ 2) \end{aligned} \tag{3.11}$$

The offspring is built using the following procedure. First, two cut points along the strings are randomly selected. Suppose that the first cut point is selected between the third and the fourth string element, and the second one between the sixth and seventh string element. For example:

$$\begin{aligned} P_1 &= (7\ \ 3\ \ 8 \mid 2\ \ 1\ \ 4 \mid 5\ \ 6) \\ P_2 &= (1\ \ 3\ \ 5 \mid 6\ \ 4\ \ 8 \mid 7\ \ 2) \end{aligned} \tag{3.12}$$

The substrings between those cut points are called the mapping sections. The mappings will be $2 \leftrightarrow 6$, $1 \leftrightarrow 4$ and $4 \leftrightarrow 8$. Next, the mapping section of the first parent is copied into the second offspring O_2, and the mapping section of the second parent is copied into the first offspring O_1:

$$\begin{aligned} O_1 &= (-\ -\ - \mid 6\ \ 4\ \ 8 \mid -\ -) \\ O_2 &= (-\ -\ - \mid 2\ \ 1\ \ 4 \mid -\ -) \end{aligned} \tag{3.13}$$

Then offspring i (i = 1, 2) is filled up by copying the elements of the i-th parent. If an element is already present in the offspring, it is replaced according to the mappings. Then we can fill in the elements, from the original parent, for those which have no conflict as follows:

$$\begin{aligned} O_1 &= (\ 7\ \ 3\ \ - \mid 6\ \ 4\ \ 8 \mid 5\ \ -) \\ O_2 &= (-\ \ 3\ \ 5\ \mid 2\ \ 1\ \ 4 \mid 7\ \ -) \end{aligned} \tag{3.14}$$

Hence, the first slot ($-$) in O_1 is 8 which comes from first parent but 8 is already in this offspring, so we check the mapping $4 \leftrightarrow 8$ and see again 4 existing in this offspring, again check the mapping $1 \leftrightarrow 4$, so 1 occupies at first ($-$). Similarly, the second ($-$) in first offspring is 6 which comes from first parent but 6 exists in this offspring; we check the mapping $2 \leftrightarrow 6$ as well, so 2 occupies at second ($-$). Analogously, we can find O_2. Thus, O_1 and O_2 are:

$$\begin{aligned} O_1 &= (7\ \ 3\ \ 1 \mid 6\ \ 4\ \ 8 \mid 5\ \ 2) \\ O_2 &= (8\ \ 3\ \ 5 \mid 2\ \ 1\ \ 4 \mid 7\ \ 6) \end{aligned} \tag{3.15}$$

The OX operator produces offspring by choosing a section of one parent and preserving the relative order of the other parent. For example, consider the parents P_1 and P_2 given in Equation (3.11) and the same two random cut points. The offspring O_1 and O_2 are built as follows. First, the segments between the cut point are copied into the offspring, which gives:

$$\begin{aligned} O_1 &= (-\ -\ -\ |\ 2\ \ 1\ \ 4\ |\ -\ -) \\ O_2 &= (-\ -\ -\ |\ 6\ \ 4\ \ 8\ |\ -\ -) \end{aligned} \tag{3.16}$$

Unlike the PMX operator, the OX operator does not perform a mapping of genetic material between the parents' string. Next, starting from the second cut point of one parent, the rest of the bits are copied in the order in which they appear in the other parent, also starting from the second cut point and omitting the position that are already present. When the end of the parent string is reached, we continue from its first position. In our example the sequence in the P_2, starting from the second cut point is $7 \rightarrow 2 \rightarrow 1 \rightarrow 3 \rightarrow 5 \rightarrow 6 \rightarrow 4 \rightarrow 8$. After removing 2, 1, and 4, which are already in O_1, the sequence is $7 \rightarrow 3 \rightarrow 5 \rightarrow 6 \rightarrow 8$. This sequence is placed in O_1 starting from the second cut point. Analogously, we can find O_2. Then, the following offspring O_1 and O_2 are generated:

$$\begin{aligned} O_1 &= (5\ \ 6\ \ 8\ |\ 2\ \ 1\ \ 4\ |\ 7\ \ 3) \\ O_2 &= (3\ \ 2\ \ 1\ |\ 6\ \ 4\ \ 8\ |\ 5\ \ 7) \end{aligned} \tag{3.17}$$

The probability of a crossover operation being performed to generate new offspring is specified by the crossover rate. The number of crossovers in one generation is calculated as the crossover rate multiplied by the pop_size.

The mutation operator is used to maintain adequate diversity in the population and avoid premature convergence. The random mutation was used, which introduces random variation by first randomly choosing a position from a randomly selected pattern and reassigning a randomly valid value for that position. The operator works similarly to the mutation operator proposed by (Eddy & Lewis, 2001) with the modification that the replacement value must be different from the other pattern positions, thus ensuring that there are no repeated positions. The number of mutations per generation is calculated as the product of the mutation rate and the $pops\_size$.

The diagram in the Figure 3.7 shows how the mutation operator works. An example of an array $[m, n]$, where $m = 3$ is the population size and $n = 6$ is the number of variables or positions, is considered. To modify the population, the position $[2, 4]$ is randomly chosen and the random value $r = 10$ is assigned to that position. This process is repeated until the defined number of mutations is reached.

$$\begin{array}{|c|c|c|c|c|c|}\hline 22 & 71 & 35 & 67 & 24 & 4 \\ \hline 56 & 16 & 43 & 84 & 27 & 58 \\ \hline 78 & 1 & 19 & 3 & 11 & 18 \\ \hline \end{array} \rightarrow \begin{array}{|c|c|c|c|c|c|}\hline 22 & 71 & 35 & 67 & 24 & 4 \\ \hline 56 & 16 & 43 & 10 & 27 & 58 \\ \hline 78 & 1 & 19 & 3 & 11 & 18 \\ \hline \end{array}$$

Figure 3.7: Example of how the proposed mutation operator works. An array of dimensions [3, 6] is shown in which a modification is applied at position [2, 4] (shaded).

3.3.2. Genetic Algorithms Implementation Results

Two strategies were used to evaluate the performance of the optimization code. In the first case, the results were compared using the PMX and OX crossover operators and the OF function (see Equation (3.2)). Then, another case was analyzed using the PMX operator and the OF_2 function that includes the flatness term. The defined GA parameters are shown in Table 3.1. The code was executed on a multi-core processor Ubuntu workstation (Intel® Xeon® CPU E5-2623 v4 – 2.60 GHz x 15) with 64 GB of RAM, but single-core calculations were performed due to the limitation of the ERANOS simulation code.

To encourage the exploration of the design space, known feasible LPs were inserted into the initial population and the rest were filled with random solutions. Feasible LP means that there are no more than two contiguous positions. This allows to increase the convergence speed and the quality of the solution. A population size of 150 and a maximum number of generations of 300 were defined. A crossover rate of 1.0 and a mutation rate of 0.1 were defined. The crossover rate equal to 1.0 implies that 100% of the individuals in the population are crossed, which accelerates the search. The low mutation rate is used to generate random changes that give rise to new offspring different from the parents, which encourages diversity in the population of LPs. During the choice of GA parameters, it was found that a high mutation rate affected the performance of the algorithm since mutations produce infeasible patterns.

For the first case, 10 independent runs were performed using the OF function and both crossover operators. Table 3.2 summarizes the best OFvalue and the PPF for each run and the average value. In addition, Figure 3.8 and Figure 3.9 show the evolution of the best OF over the generations for each run for the PMX and OX operators, respectively.

Table 3.1: GA parameters.

Parameter	Value
Population size	150
Number of generations	300
Selection operator	Tournament ($tournament\_size$ = 4)
Crossover operators	OX/PMX
Crossover rate	1.0
Mutation operator	Random mutation
Mutation rate	0.1

It can be seen how the algorithm explores better solutions in the search process, which means that the crossover operators work correctly. The algorithm shows good convergence (on average from the 200th generation onwards) although the worst solutions are given by premature convergence. This is one of the major drawbacks associated with GAs and is the trend to converge to local optima of the OF being the cause of a decrease in the quality of the solutions found.

As shown in Table 3.2, the best solution on average was found with the PMX operator and twice as fast as using the OX operator. Note that the execution time depends on the calculation time in the simulation of the patterns in ERANOS, which takes on average 30 seconds. The calculation time will be governed by Equation (3.5), i.e., the patterns that meet the condition are evaluated in ERANOS.

Table 3.2: Optimization results. Best OF and PPF for 10 independent runs using the PMX and OX crossover operators.

Run number	PMX		OX	
	Best OF	PPF[1]	Best OF	PPF
1	1.011991	1.1726	1.012032	1.1729
2	1.012087	1.1746	1.011973	1.1723
3	1.012082	1.1739	1.012087	1.1743
4	1.012090	1.1743	1.012063	1.1752
5	1.012032	1.1729	1.012021	1.1729
6	1.012081	1.1735	1.012068	1.1747
7	1.012114	1.1745	1.011993	1.1746
8	1.012073	1.1743	1.012026	1.1747
9	1.012021	1.1721	1.012066	1.1735
10	1.012082	1.1739	1.012114	1.1745
Average	1.012065	1.1737	1.012044	1.1740
Maximum	1.012090	1.1743	1.012087	1.1743
Running time[2]	30.9 h	–	57.5 h	–
LP evaluation time in ERANOS[3]	30 s		30 s	

1. PPF value over the cycle.
2. Average GA code execution time per case study.
3. This calculation time corresponds to the simulation of the operating cycle of each LP in the ERANOS code (365 days and diffusion 7-groups approach).

Figure 3.11 and Figure 3.12 show the average PPF value and the HER^{BOC} and MLHGR values over the generations, respectively. As can be seen, the HER^{BOC} increases because the k_{eff}^{EOC} increases since the typical curve of k-eff vs operation time (see Figure 2.17 in Subsection 2.2.7). In addition, the MLHGR has similar behavior to PPF and remains below the limit value. This was expected, since the power is proportional to the energy released (then, the heat produced) by fissions.

Figure 3.13 shows the best LP in the first generation and the best LP found for the PMX operator. OX tends to produce offspring that resemble both parents so it may have difficulty exploring the search space more efficiently than PMX. It can be seen how the constraint of two contiguous fresh assemblies is satisfied. Over the generations, the algorithm places pairs of assemblies since higher OF values are obtained. If the constraint did not exist, the 29 assemblies would be stacked towards the center of the core. Figure 3.14 shows the PPF distribution at the BOC for the best LP where the maximum value does not exceed the value of 1.25.

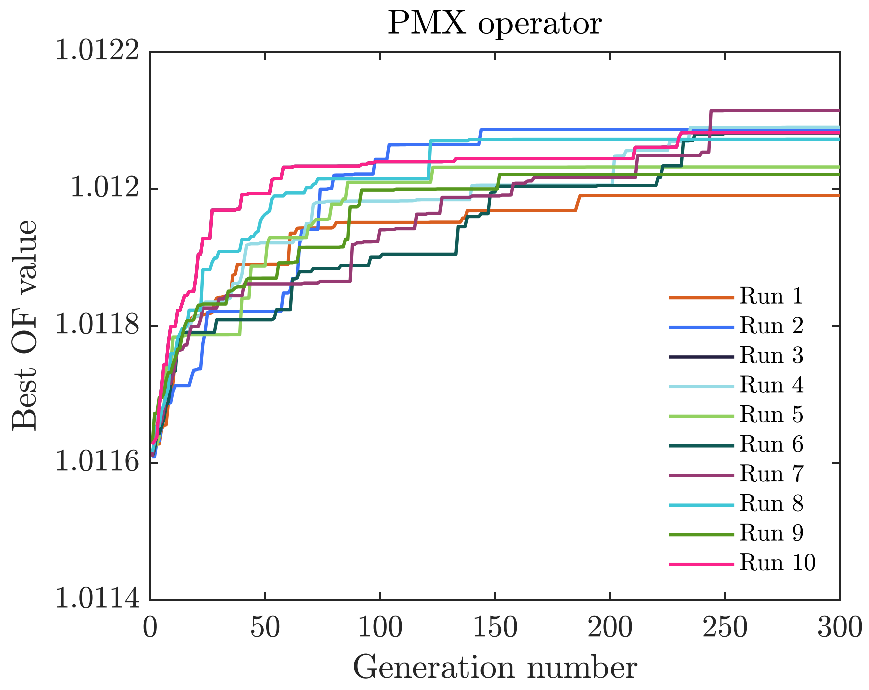


Figure 3.8: Best OF value evolution for 10 independents runs using the PMX operator.

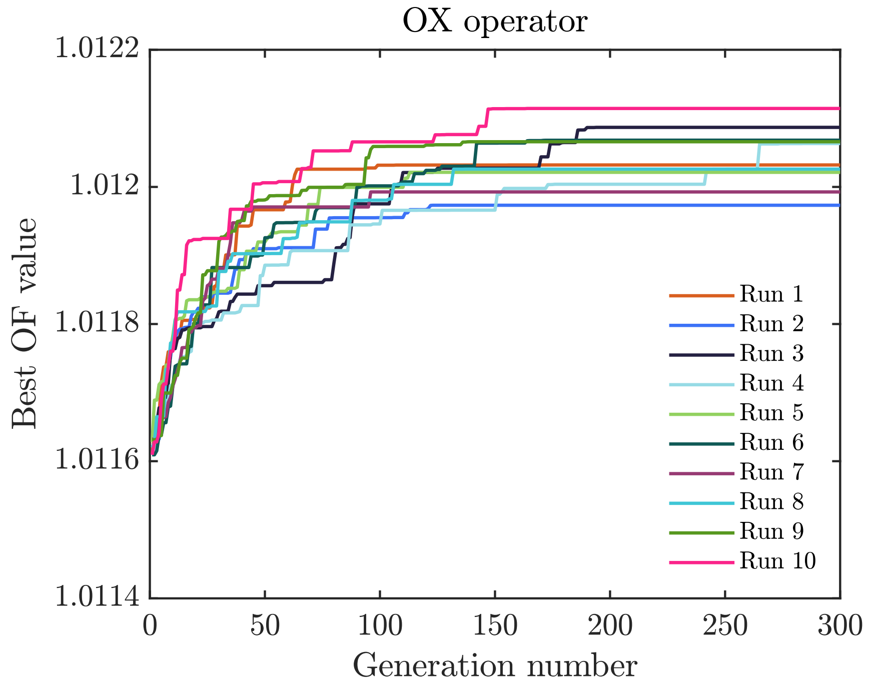


Figure 3.9: Best OF value evolution for 10 independents runs using the OX operator.

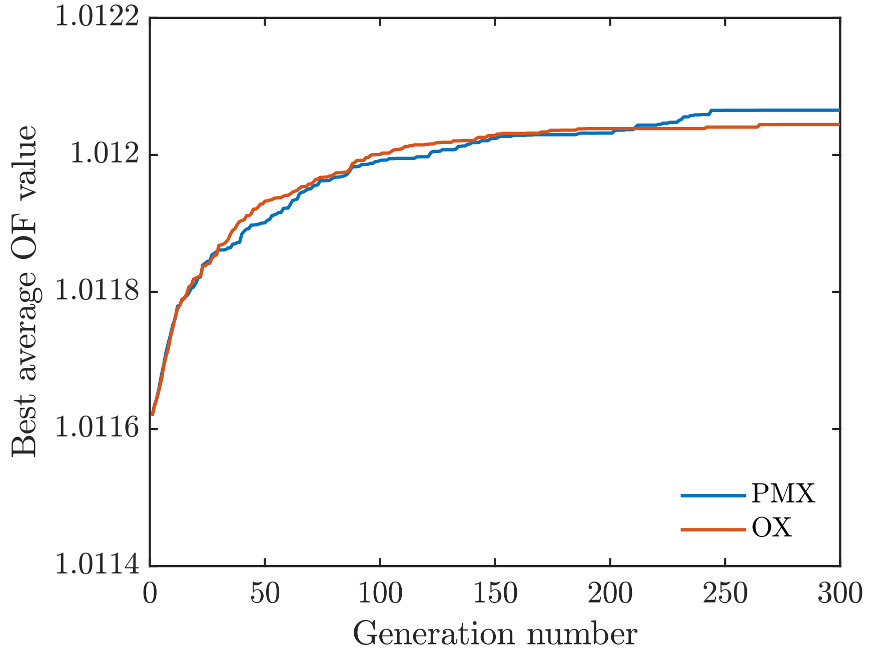


Figure 3.10: Average OF value evolution for 10 runs using the PMX and OX operators.

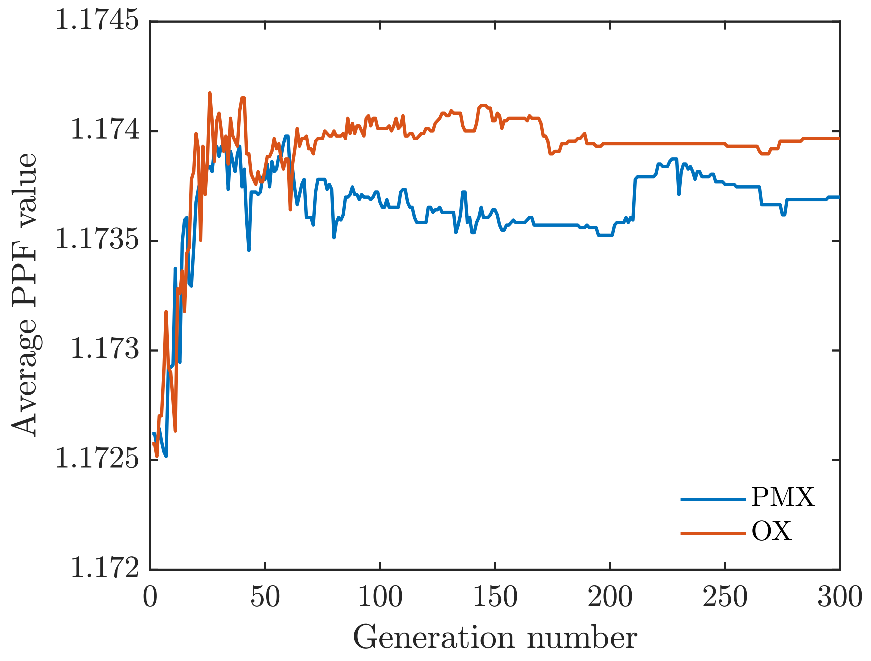


Figure 3.11: Average PPF value evolution for 10 runs using OF and the PMX and OX operators.

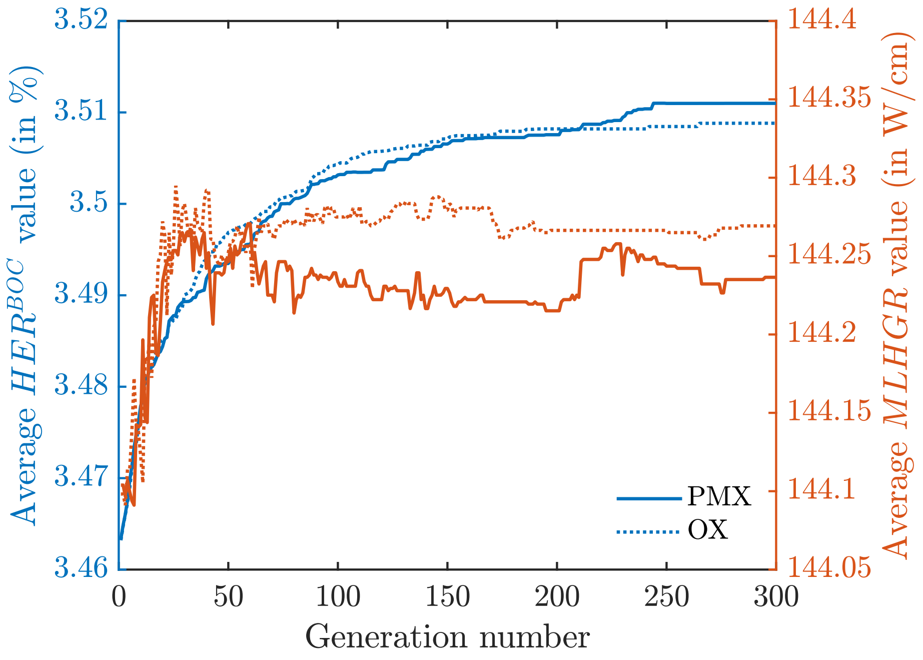


Figure 3.12: Evolution of the average values of HER^{BOC} and MLHGR for 10 runs using OF and the PMX and OX operators.

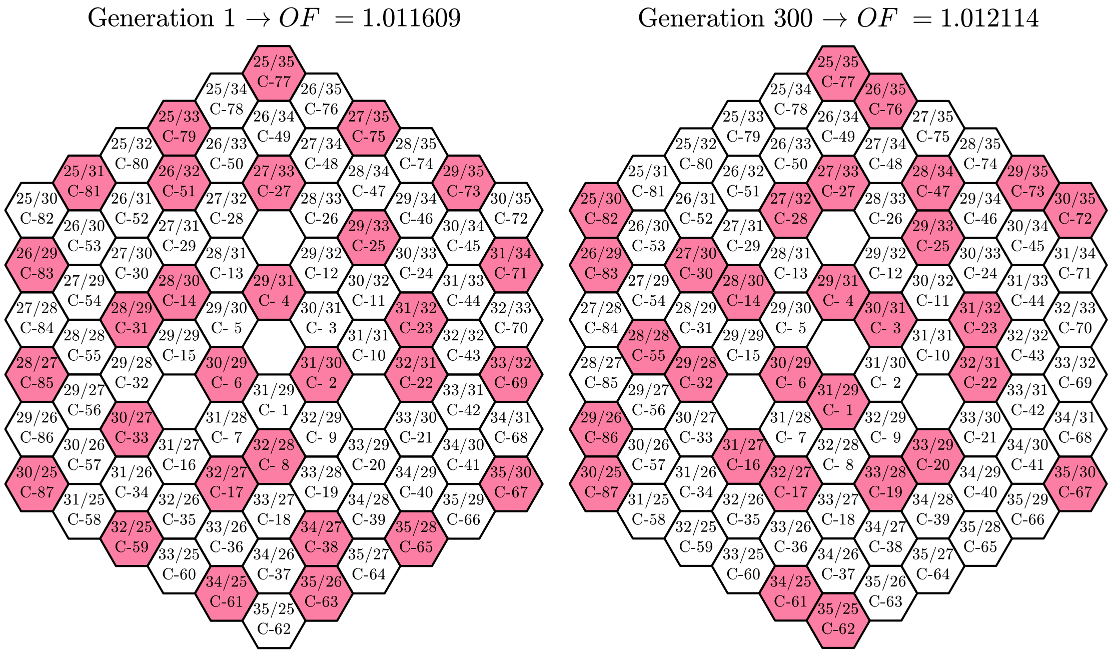


Figure 3.13: The best LP in the 1st generation and the best LP found using OF and the PMX operator.

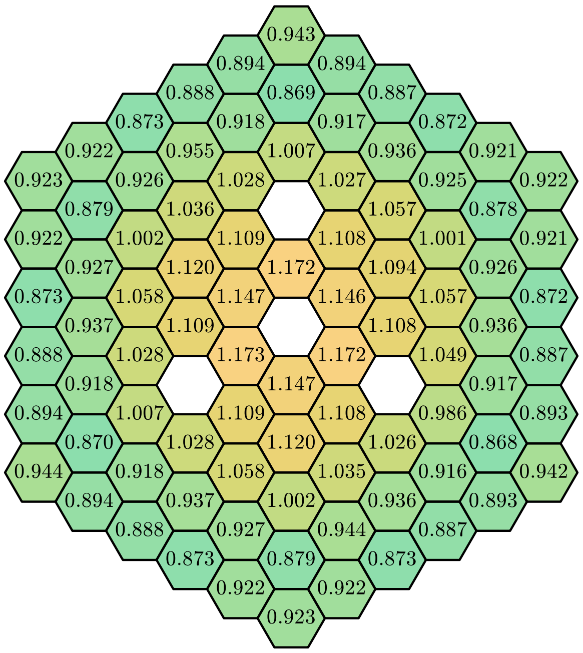


Figure 3.14: PPF distribution at the BOC corresponding to the best LP found using OF and the PMX operator.

Figure 3.15 depicts the k-eff value over the operating cycle corresponding to the initial LP and the best LP. Note that the k-eff at the EOC increases 187.7 pcm, leading to an increase in cycle length.

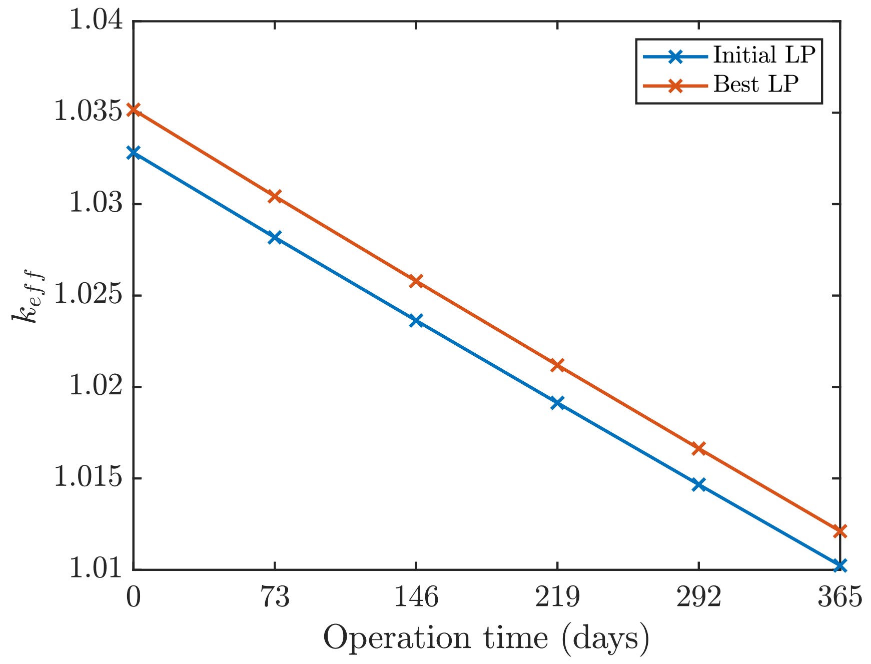


Figure 3.15: Evolution of the k-eff value over the operating cycle for the initial LP and the best LP found.

As a second test of the algorithm performance, the OF_2 function and the PMX operator were used. The GA parameters defined were given in Table 3.1. Figure 3.16 shows the evolution of the best OF_2 value, while Figure 3.17 and Figure 3.18 show the evolution of the k-eff at the EOC, HER^{BOC}, $MLHGR$ and PPF parameters. The OF_2 behavior is as expected, the population of LPs evolves towards a better solution and the convergence starts at the 200th generation.

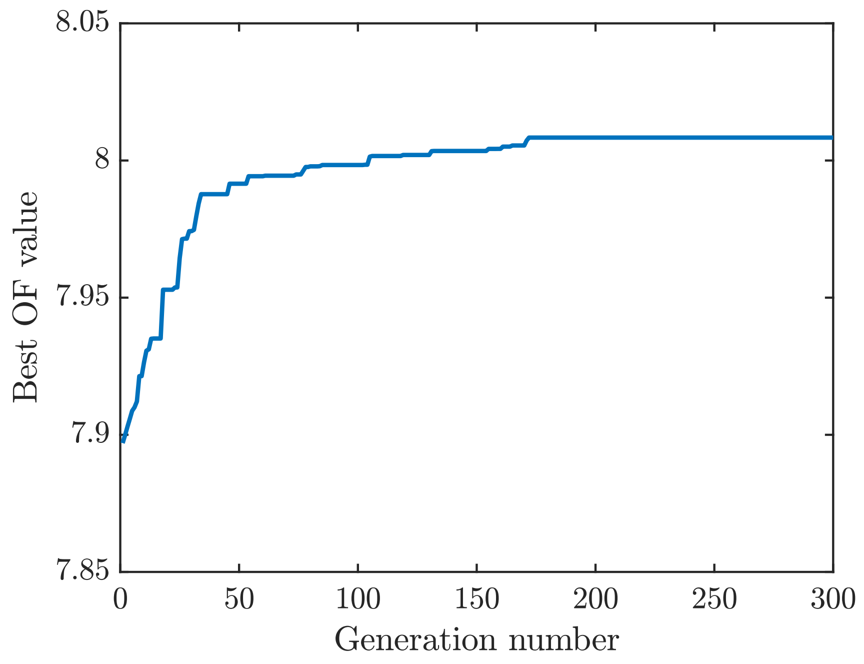


Figure 3.16: Best OF_2 value evolution.

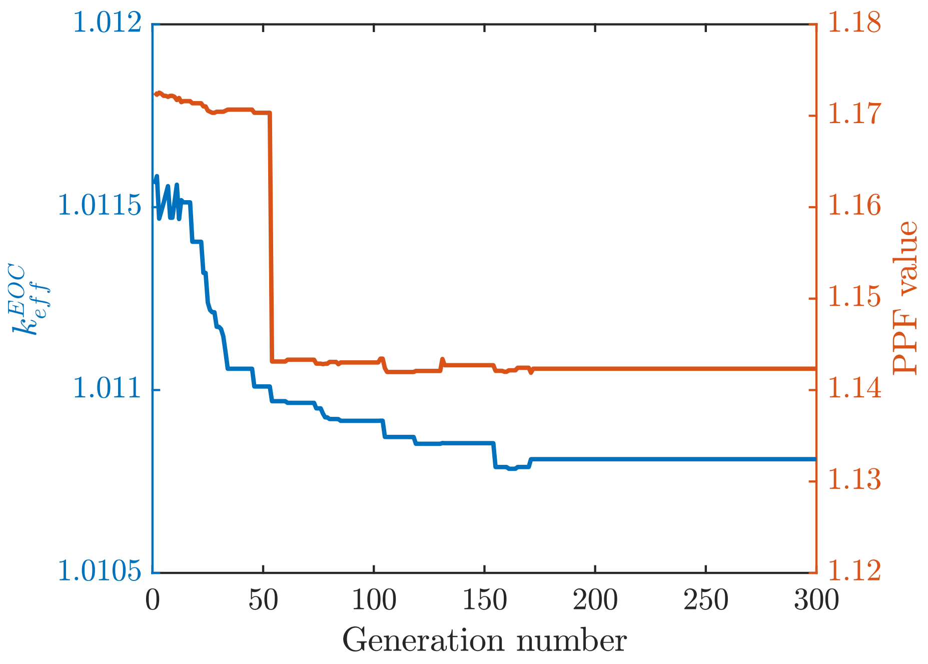


Figure 3.17: k-eff and PPF parameters evolution of the OF_2 function.

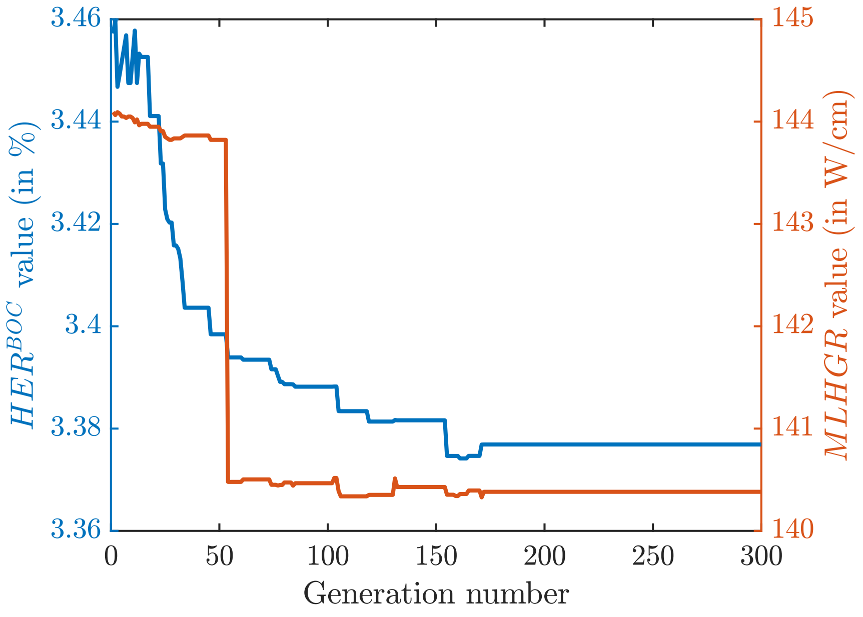


Figure 3.18: HER^{BOC} and MLHGR parameters evolution of the OF$_2$ function.

The impact of the $Flatness$ term can be verified in the decrease of the k-eff since with this penalty the algorithm is forced to place the assemblies towards the periphery of the core. Figure 3.19 shows the initial LP and best LP found while the PPF distribution at the BOC is presented in Figure 3.20. The k-eff at the EOC is maximized, but now the best patterns are those that result in a more uniform distribution. The PPF maximum is reduced from 1.17 to 1.14 but also the k-eff value and the cycle length decrease with respect to the first case using OF.

The algorithm performs as expected using both objective functions. In the first case by maximizing the k-eff at EOC without considering the power distribution uniformity, and subsequently, by penalizing with the $Flatness$ term, in both cases meeting the imposed constraints.

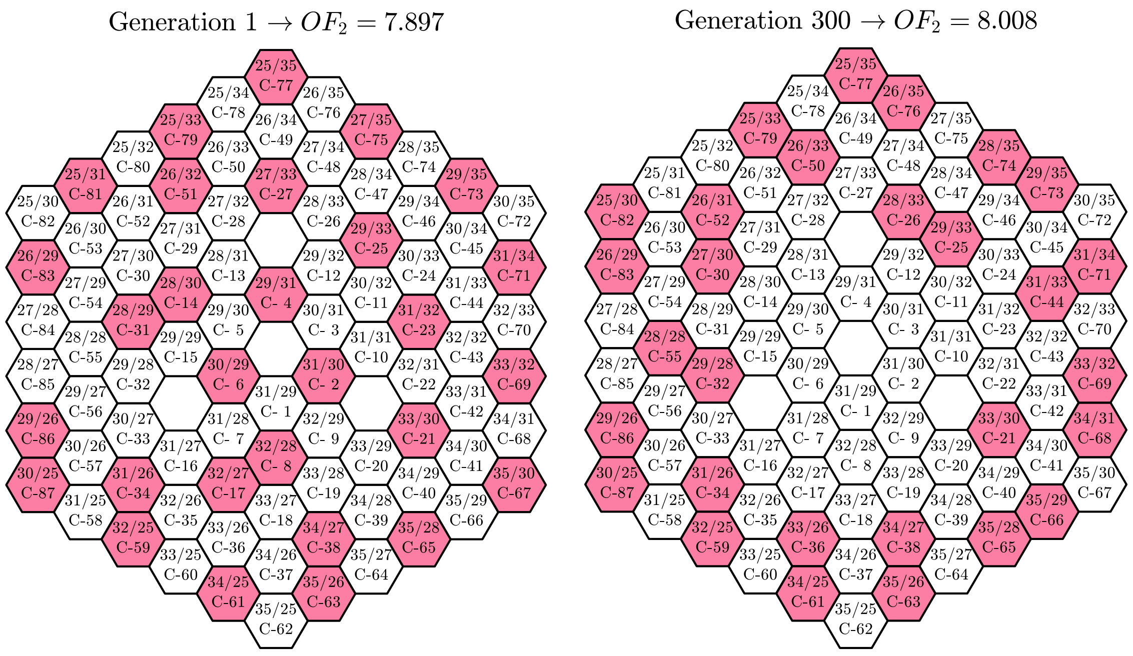

Generation 1 $\rightarrow OF_2 = 7.897$
Generation 300 $\rightarrow OF_2 = 8.008$

Figure 3.19: The best LP in the 1st generation and the best LP found using OF$_2$ and the PMX operator.

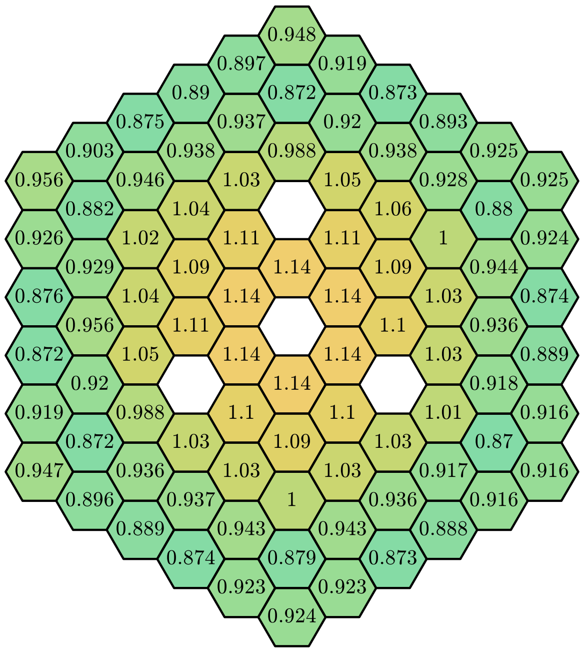


Figure 3.20: PPF distribution at the BOC corresponding to the best LP found using OF$_2$ and the PMX operator.

3.4. Tabu Search applied to the ALLEGRO Fuel Loading Pattern Optimization

Tabu search (TS) is a popular metaheuristic proposed by (Glover, 1986) and since then TS has been successfully widely applied to combinatorial optimization problems in different areas of engineering and research. A comprehensive introduction to TS can be found in the book by (Glover, 1989, 1990; Glover & Laguna, 1997). In nuclear engineering Tabu search technique has been successfully used in several works (Castillo et al., 2005, 2007; François et al., 2003; Hill & Parks, 2015; Jagawa et al., 2001; Wu et al., 2016), especially in optimization of nuclear fuel management.

In a first approximation, the TS could be seen as a simple local search method, that is, starting from a solution s_0 obtained randomly, a neighborhood $N(s_0)$ is generated by some known procedure. From the generated neighborhood, the best neighbor s_{best} is found. The best neighbor is defined as the one with the best value of an objective function. A new neighborhood is generated from the previously found best neighbor $N(s_{best})$ and the process is repeated until a stopping criterion is reached. At the end of the process, we have a succession of points $(s_1, s_2, s_3, \dots, s_n)$ which are the best neighbors of each step. An important point in many problems where the Tabu search is applied, is the fact that the obtaining of the different solutions is done from a movement, that is, if we have the solution s and this is a vector, in many cases the obtaining of other solutions will be realizing a movement between two elements of this vector. Up to this point, TS does not differ in any way from a simple local search method, however, the concept that characterizes this technique and makes it different from the simple local search, is the forbidden movement or tabu movement, which we will explain below.

Starting from the previous explanation, with the first neighborhood generated and with the choice of the best neighbor, TS technique uses a list that can be of fixed or variable size, known as tabu list. This list indicates the number of iterations that a move is forbidden. It means, that movement has the status tabu. At the beginning the tabu list is empty and is dynamically updated as the iterative process progresses. To clarify the ideas, let us suppose that the tabu list has a size equal to 5. At the beginning of the process, the best neighbor (s_{best}) enters to the tabu list in the first place. In the second iteration, the previous best neighbor moves to the second position in the tabu list, and the second-best neighbor takes the first place. Note that the best neighbor (s_{best}) will remain with the tabu status for 5 iterations (see Figure 3.21). The tabu list has two columns indicating the movement from one solution s_1 to another s_2.

	1	2	3	4	5
1	s_{best}^1	s_{best}^2	s_{best}^3	s_{best}^4	s_{best}^5
2		s_{best}^1	s_{best}^2	s_{best}^3	s_{best}^4
3			s_{best}^1	s_{best}^2	s_{best}^3
4				s_{best}^1	s_{best}^2
5					s_{best}^1

Figure 3.21: Tabu list update.

An important point of the tabu status is that, as long as a solution remains in the tabu list during the defined number of iterations, this solution cannot be chosen as the best neighbor, even if it is the best value of the neighborhood in any of the iterations of the tabu status. This tells us that in a maximization problem (the minimization problem is equivalent) the following condition is not always satisfied:

$$OF(s_1) > OF(s_2) > OF(s_3) > OF(s_4) \dots > OF(s_n) \tag{3.18}$$

So, it may happen that at some times that $OF(s_{n+1}) < OF(s_n)$. This condition, far from being detrimental, produces that during the iterative process it is possible to get out of a local optimum, which makes Tabu search technique an extremely powerful tool compared to traditional optimization methods.

The above explanation corresponds to tabu search in its most basic form, however, over time modifications have been made to this technique to improve its performance. Three of the most important modifications are the following: the aspiration criterion, the review of only a part of the neighborhood and the variable list size. Let us see what each of these consists of.

The first of the options is the so-called aspiration criterion, which consists of eliminating the tabu status of a solution or move, as long as this solution or move leads to the best value obtained with the objective function, of all the solutions obtained so far. This criterion allows considering solutions with which the best result is obtained in all iterations of the process.

With respect to the revision of only a part of the neighborhood, it is convenient to apply it when the cost of the evaluation of the objective function is too high, in computational terms. In such case it is recommended to revise only a percentage of the neighborhood. For example, if N_size it is size of the neighborhood $N(s)$, then it is recommended to check only $N\_fraction \times N\_size$, where $N\_fraction$ is a number between 0 and 1, which will depend on the problem.

Finally, in the early days of tabu search it was thought that working with a fixed tabu list size was sufficient. Later studies indicated that handling a variable list size helped to avoid cycling during the iterative process, which results in search diversification. For this reason, it is suggested to use a variable list size. In the same vein, small list sizes were initially considered sufficient, around 5 or 7, and again later studies indicated that variable sizes between 15 and 21 for the tabu list are much better.

We will now explain the details of the implementation for our problem. In the proposed reloading scheme, 29 fresh fuel assemblies (1/3 of the total assemblies) are loaded at BOC; the other 2/3 of the fuel assemblies are burnt assemblies (described in Subsection 2.2.8). Accordingly, the isotopic composition corresponding to each assembly is different, i.e., there are fresh and burnt fuel assemblies in the equilibrium core. Therefore, the problem will be to find the optimal configuration satisfying the operational and safety constraints imposed. The active core layout can be seen in Figure 2.19. Each LP (or solution s) is represented as a vector of 87 non-repeating integers. Starting from the center of the reactor and counter-clockwise, the following vector s is contructed:

$$s = (p_1, p_2, p_3, \dots, p_{87}) \tag{3.19}$$

This array corresponds to a particular configuration of the reactor and therefore a solution to our problem. The iterative process starts from a randomly generated solution, with the restriction that there cannot be more than two fresh fuel assemblies together. Starting from this solution and without loss of generality, let us assume that this solution is a vector $s_0 = s$. To obtain the elements of a neighborhood, the movements depicted in Figure 3.22 are performed. For example, the interchange between two entries ($p_2 \leftrightarrow p_{40}$) in the vector s_0, will give us a new solution s_1:

$$s_0 = (p_1, p_2, p_3, \dots, p_{87}) \to s_1 = (p_1, p_{40}, p_3, \dots p_2, p_{87}) \tag{3.20}$$

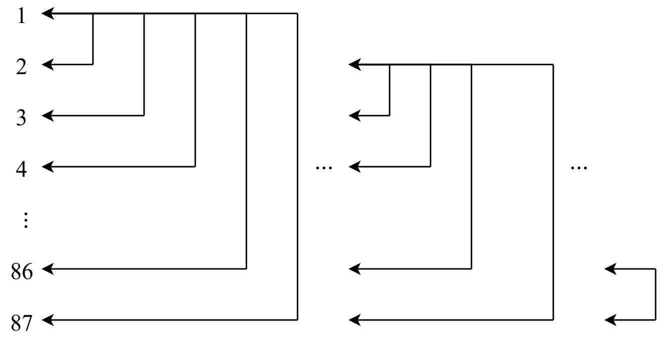


Figure 3.22: Representation of the movements made in the vector *s* to generate the neighborhood.

Figure 3.23 shows an example assuming a vector of size 6 where the total number of swaps will be $swaps = 6 \times (6-1)/2 = 15$. Under this procedure and if there were no restrictions, a neighborhood has $swaps = (87 \times (87-1))/2 = 3741$ different solutions.

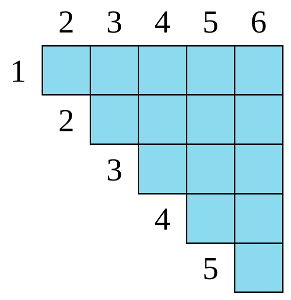


Figure 3.23: Swaps performed assuming a vector of size 6 (swaps = 15). The array m×n represents the positions to be swapped.

For our implementation the aspiration criterion was applied and a variable tabu list was used. For this purpose, we define $tabu\_time$, an array representing the moves performed corresponding to the best solutions found over the iterations. Note that $tabu\_time$ will be a two-dimensional array of size 87×87 (or a m-by-n matrix). The array is updated throughout the iterations with the value computed using the following equation:

$$tabu\_time(m, n) = iteration + tabu\_tenure \tag{3.21}$$

Where the $tabu\_tenure$ term is defined as a random integer between 7 and 21 that defines the tabu list size; m and n are the indexes of the swapped positions; $iteration$ is the number of the current iteration. Therefore, Equation (3.21) can be written as:

$$tabu\_time(m, n) = iteration + randi([7\ 21]) \tag{3.22}$$

Due to the computational cost of evaluating a solution is too high, only 5% of the entire neighborhood is checked; this is equivalent to evaluate about 50 solutions of each neighborhood.

3.4.1. Neighbor search procedure

In this section, the procedure to perform the interchange between two positions in ERANOS will be briefly discussed. Specific ERANOS modules and appropriate complex subroutines are available to perform a detailed core follow-up. Each individual assembly can be followed through its entire life.

Several modules for in-core fuel management were used. It is important to correctly define each procedure since the solution to the problem will depend on it. In fact, in the procedure developed, the positions are not interchanged, since the coordinates X/Y are fixed, but the isotopic composition. First, an array C_0 containing the isotopic composition of each fuel assembly at the beginning of the equilibrium cycle was generated. This array C_0 will have dimensions

(num_iso, num_FA) where num_iso is the number of isotopes and num_FA is the number of fuel assemblies. Then, using this array, the procedures to generate the movements are created. With these procedures and making use of the modular structure of ERANOS, the swaps for the neighbor search phase of the Tabu Search technique can be performed. Both the generation and execution of the ERANOS input files are automated using MATLAB scripts.

The solution s is a vector of non-repeating integers of length 87. The swaps between entries for the neighbor search will be performed. The entries are positions of fuel assemblies linked to the C_0 array which contains the isotopic composition at each position. This linkage between s and C_0 is required because the state of each assembly will be tracked throughout the iterations. The state of an assembly refers to whether it is fresh or burnt (F or B).

The vector s (or LP) is linked to a fixed $core\_xy$ array that contains the X/Y coordinates of each position in the reactor core. Swapping two entries in s will produce the same effect on the $core\_xy$ array. This procedure is incorporated in the interface developed between ERANOS and the TS algorithm, which is shown in Figure 3.2. This interface allows the communication between the TS algorithm and the external ERANOS simulation code. The interface transforms the LP vector and executes its evaluation in the ERANOS code. Once the simulation is completed, the core parameters required to calculate the OF value (k_{eff}^{EOC}, PPF, HER and $MLHGR$) are retrieved.

To generate a neighbor s_1 from the solution s, two entries are swapped, but the following conditions must be met:

1. $p_1 = B, p_2 = B$. Both fuel assemblies are burnt (state BB).
2. $p_1 = F, p_2 = B$. The fuel assemblies have different states, fresh or burnt (state FB).

According to these constraints, swapping $p_1 = F$ and $p_2 = F$ (both are fresh fuel) is not allowed. In addition, a maximum value of two contiguous fresh fuel assemblies was considered to avoid high power peaks in the core. By applying these set of rules, the neighborhood will be considerably reduced.

Figure 3.24 shows the flowchart of the developed algorithm and the following steps summarize how it works:

1. Initialization. In this phase the TS parameters are defined: number of iterations (num_iters), number of modifications to the initial solution (num_mod) and the subset of the neighborhood to be explored ($N\_fraction$ ∈ [0, 1]).
2. Generate the initial solution. A random solution s_0 and the corresponding C_0array are generated. For this purpose, random exchanges in the arrays, s and C, are performed. The number of modifications is set by the variable num_mod.
3. Main loop. The iterative process begins.
4. Generate the list of movements to be performed. A random list of size $N\_size = N\_fraction \times swaps$ is generated.
5. Neighbor search. In this phase, the vector s is modified by performing swaps between positions according to the imposed constraints. Each swap will produce a neighbor s_n that will be evaluated in ERANOS to calculate the corresponding OF value. Figure 3.24 shows how the interface between the optimization algorithm and the simulation code works. At the end of the search process the best neighbor s_{best} is stored. In addition, a file is exported that contains the patterns evaluated throughout the iterations, s_n, and the computed OF value.
6. Update solution. At this stage the aspiration criterion (global solution) and $tabu\_time$ values are updated with the best neighbor of the current iteration. The $tabu\_time$ array update is performed by assigning to each cell the value calculated by Equation (3.22).
7. Return to step 3. The best neighbor $s_{best}(i)$ at the iteration i will be the initial solution for the next iteration $(i + 1)$. The algorithm stops when the maximum number of iterations is reached.
8. Print results.

Figure 3.25 shows an example of how the search evolves over two iterations and considering a vector of length 6.

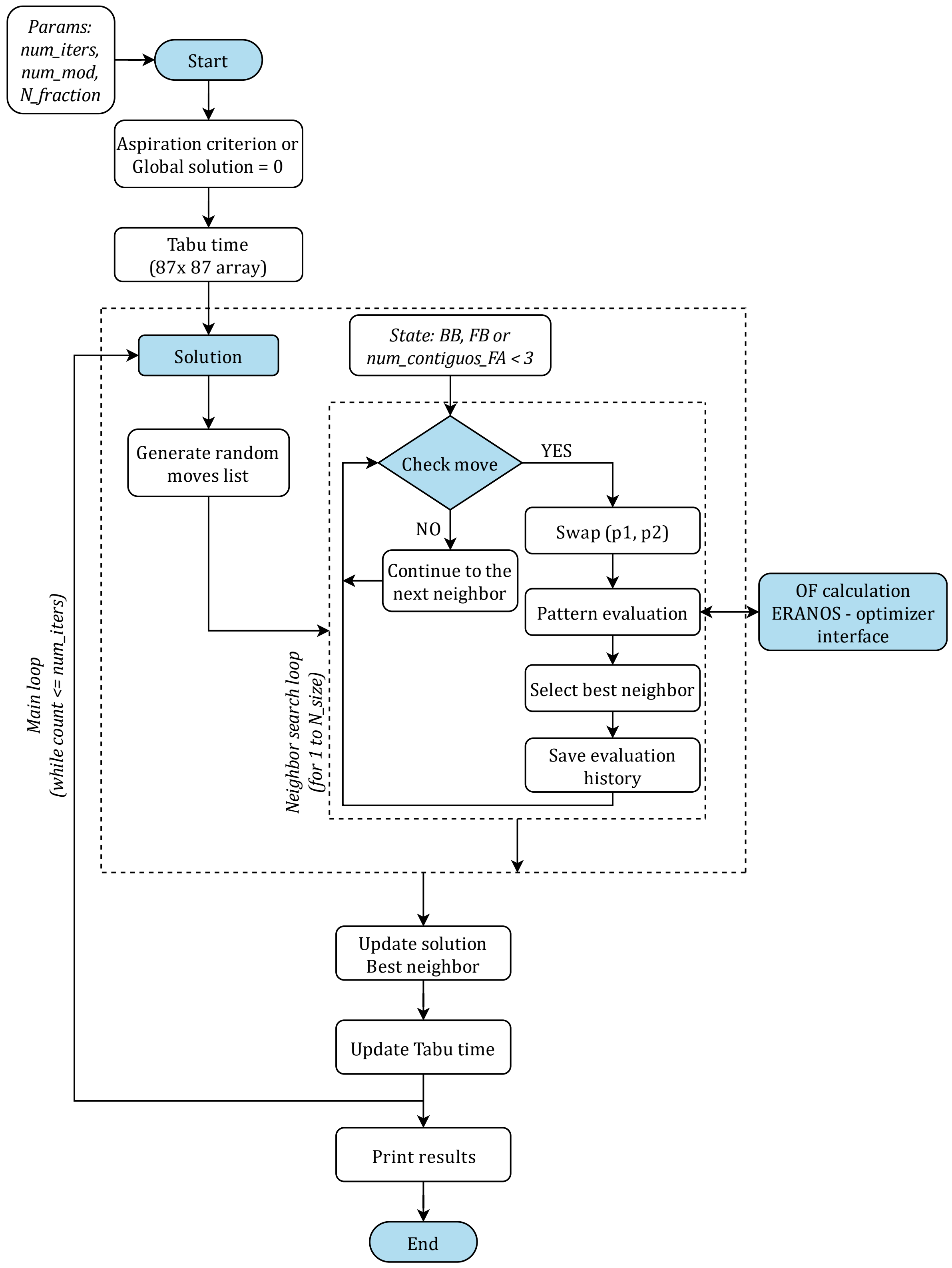

Params:
num_iters,
num_mod,
N_fraction
Start
Aspiration criterion or Global solution = 0
Tabu time (87x 87 array)
Solution
State: BB, FB or num_contiguos_FA < 3
Generate random moves list
Check move
YES
NO
Swap (p1, p2)
Continue to the next neighbor
Pattern evaluation
OF calculation ERANOS - optimizer interface
Select best neighbor
Save evaluation history
Main loop (while count <= num_iters)
Neighbor search loop (for 1 to N_size)
Update solution Best neighbor
Update Tabu time
Print results
End

Figure 3.24: Flowchart of the implemented tabu search algorithm.

vector	1	2	3	4	5	6
Concentration array C	c_1	c_2	c_3	c_4	c_5	c_6
state	B	F	F	B	F	B

Initialization (random solution)

s_0	2	4	5	3	6	1
C_0	c_2	c_4	c_5	c_3	c_6	c_1
state	F	B	F	F	B	B
set $s = s_0$						

Iter. #1

$s_1 - (2, 4)$	4	2	5	3	6	1
$s_2 - (3, 5)$	2	4	3	5	6	1
$s_3 - (1, 6)$ - best	2	4	5	3	1	6
set $s = s_3$						

Iter. #2

$s_1 - (2, 5)$	5	4	2	3	1	6
$s_2 - (4, 5)$ - best	2	5	4	3	1	6
$s_3 - (3, 4)$	2	3	5	4	1	6
set $s = s_2$						

Figure 3.25: Evolution of the neighbor search over 2 iterations. The swapped positions are highlighted in color.

3.4.2. Tabu search implementation results

Due to the random nature of the TS technique, ten independent runs were executed, and the mean and maximum OF values were reported. The study was carried out by defining 200 iterations (num_iter = 200), 1500 modifications to the initial solution (num_mod = 1500) and the 5% of the neighbourhood were checked ($N\_fraction$ = 0.05). The code was executed on a multi-core processor workstation (Intel® Xeon® CPU E5-2623 v4 - 2.60 GHz x 15) with 64 GB of RAM, with the Ubuntu operating system. Table 3.3 summarizes the best, maximum, and average OF values for each run, and the PPF, HER^{BOC} and $MLHGR$ parameters, are also reported.

Table 3.3: Optimization results. Best OF, PPF, HER^{BOC} and MLHGR for 10 independent runs.

Run number	Best OF value	PPF	HER^{BOC}	$MLHGR$
1	1.013036	1.186	3.612	145.79
2	1.013058	1.186	3.614	145.76
3	1.012990	1.185	3.608	145.62
4	1.013033	1.186	3.611	145.71
5	1.013042	1.187	3.612	145.90
6	1.012999	1.185	3.608	145.66
7	1.013036	1.186	3.613	145.76
8	1.013066	1.186	3.615	145.72
9	1.013069	1.186	3.614	145.78
10	1.013036	1.186	3.613	145.76
Average	1.013036	1.186	3.612	145.74
Best OF	1.013069	1.187	3.615	145.90
Running time[1]	45 h	–	–	–
LP evaluation time in ERANOS[2]	35 s	–	–	–

1- Execution time for each case
2- This calculation time corresponds to the simulation of the operating cycle of each LP in the ERANOS code (365 days and diffusion 7-groups approximation)

Figure 3.26 shows the evolution of the best OF value throughout the iterations. The algorithm works very well, and the expected behavior of the Tabu search technique is obtained. The inset shows how poor solutions during the iterative process are accepted. This is the advantage of implementing this metaheuristic technique, which allows escaping from local optima and encourages the exploration of the search space. The algorithm demonstrated to be a powerful tool compared to other metaheuristics methods (genetic algorithms, simulated annealing, etc.) where it is common to get trapped in local optimums for many iterations.

The main drawback found of the implemented TS algorithm is its dependence on the neighborhood size (number of moves evaluated in each iteration). However, as shown in Table 3.3, the high running time (45 h) is given by the simulation of the patterns in the ERANOS code.

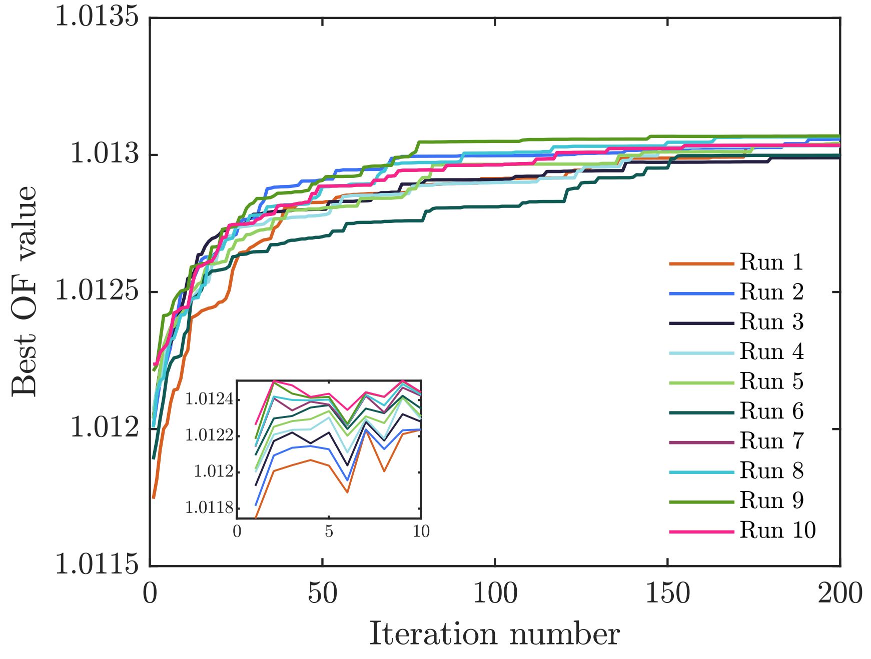


Figure 3.26: Evolution of the OF value for 10 independent runs. The inset is a zoom to the first 10 iterations.

To better understand the algorithm behavior over the iterations, the evolution of the average values of OF and PPF is shown in Figure 3.27. In addition, the plot in Figure 3.28 depicts the evolution of the HER^{BOC} and $MLHGR$ terms. The algorithm convergence starts from iteration number 150. Note that, from this value on, the algorithm does not get stuck, but the difference between the best solutions found ($\Delta = OF_{i+1} - OF_i$) in two successive iterations decreases.

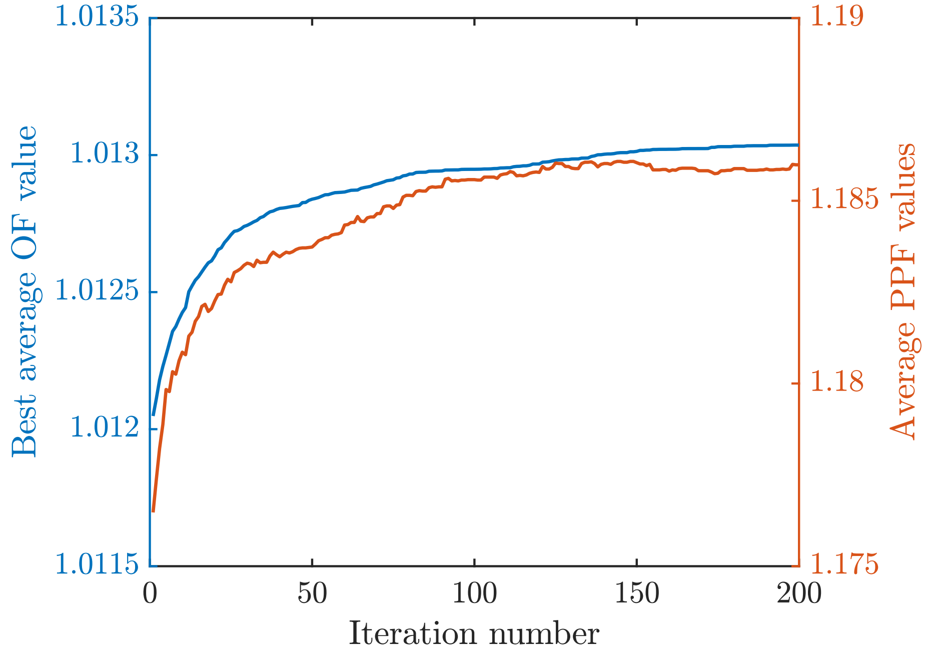


Figure 3.27: Evolution of the average value of OF and PPF.

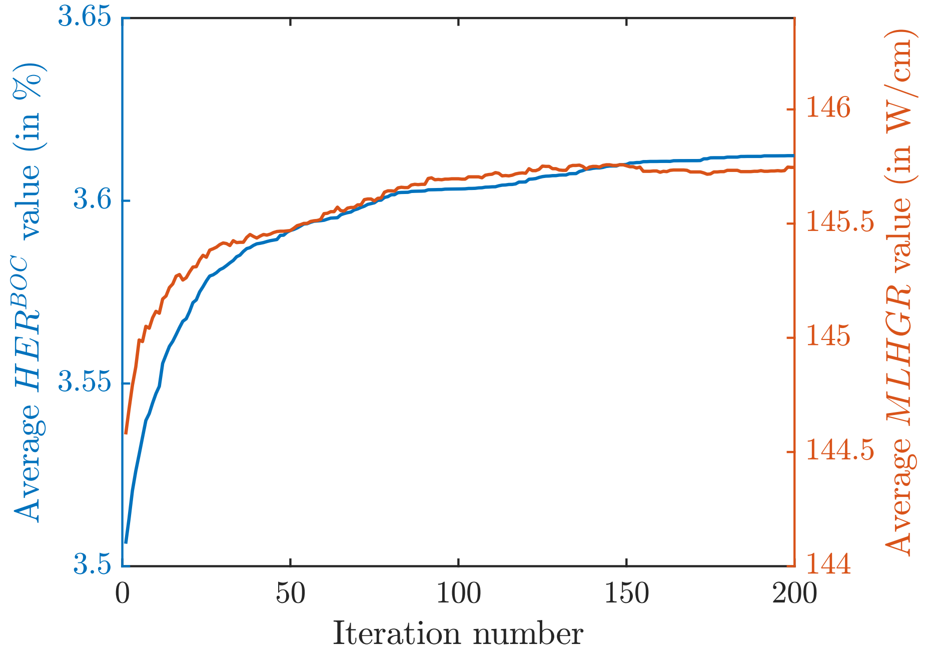


Figure 3.28: Evolution of the average value of HER^{BOC} and MLHGR.

The case with the highest OF value was selected to plot the best pattern and its corresponding power distribution. Figure 3.29 shows both graphs. Figure 3.30 represents the $tabu\_time$ array. This (87×87) array is the result of the implementation of the variable tabu list. Note that it has the shape of the example given in Figure 3.23. It contains the log of the moves performed for the best solutions found in each iteration. The value of each cell is computed using Equation (3.22). This structure is useful for testing the performance of the algorithm. The dispersion of the values reflects the diversity of the solutions found, so the implemented strategy works adequately in terms of exploring the search space.

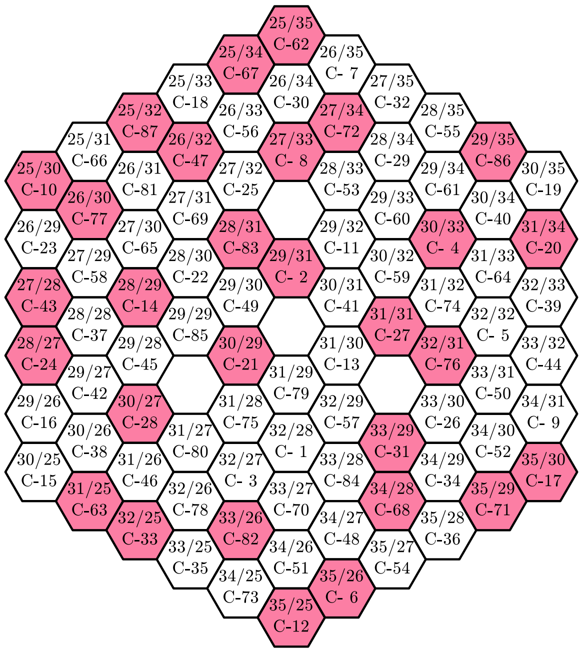
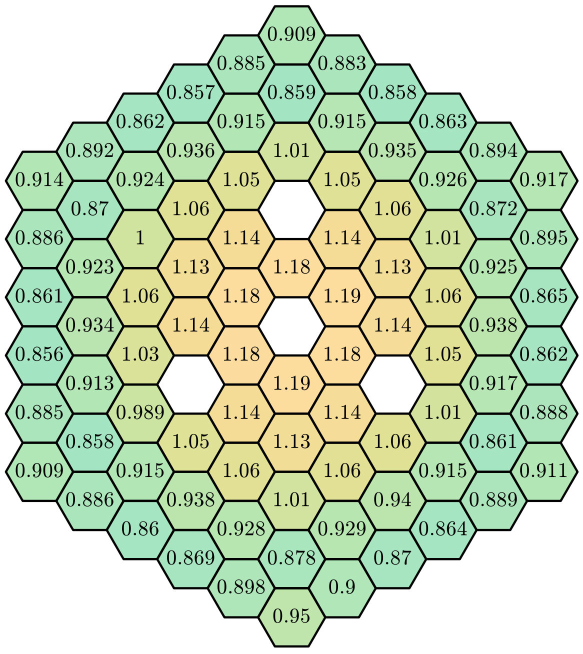



Figure 3.29: Best LP found (left) and the corresponding PPF distribution (right). Fresh fuel positions are shown in red. The PPF distribution shows the maximum values over the cycle.

Finally, Figure 3.31 depicts the k-eff value over the operating cycle corresponding to the initial LP and the best LP. Note that the k-eff at the EOC increases 283.2 pcm, leading to an increase in cycle length.

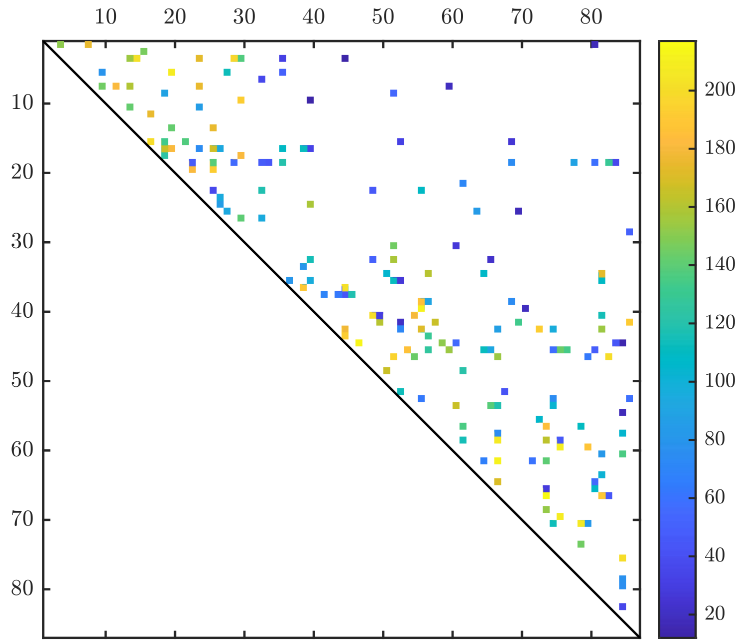


Figure 3.30: Tabu time array corresponding to the highest OF value. This is an 87×87 matrix where the cell values are the maximum *tabu_time* values assigned in each of the 200 iterations.

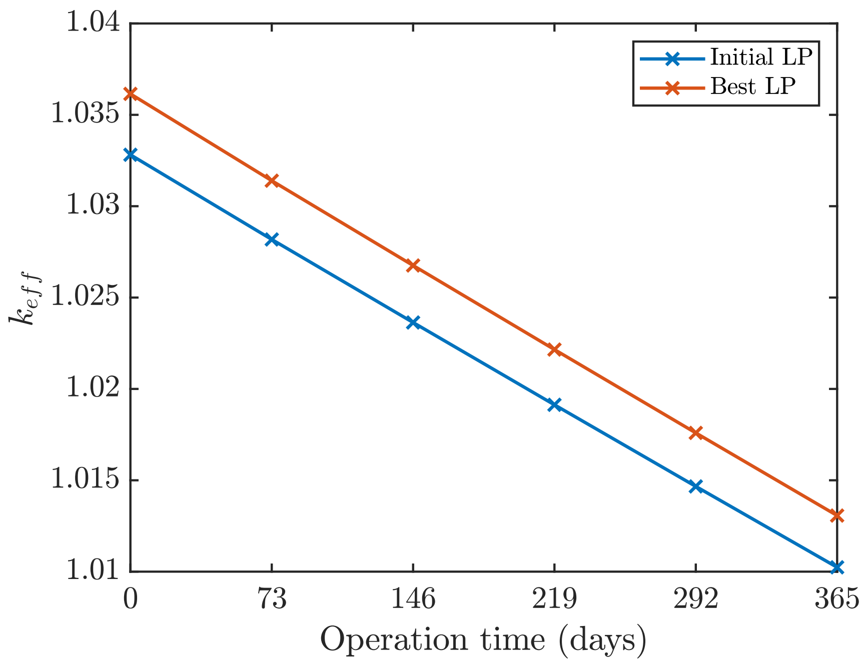


Figure 3.31: Evolution of the k-eff value over the operating cycle for the initial LP and the best LP found.

4. Conclusions and Recommendations

In the first stage of the research, the ALLEGRO reactor core was modeled using the Serpent and ERANOS reactor physics codes. Given the lack of available literature for the core configuration based on advanced ceramic fuel, the geometrical and material composition description provided may be helpful for future research. A three-dimensional heterogeneous design was developed in Serpent as a benchmark model. However, for optimization calculations, for which a large number of evaluations need to be performed, Serpent is not suitable (one of the drawbacks of the Monte Carlo method is its high computational cost). Therefore, the deterministic ERANOS code was selected. Core calculations in ERANOS can also be computationally expensive and depend on the calculation options set. Several case studies were analyzed, depending on the neutron flux calculation approximation using the TGV/VARIANT module and the energy group structure. Transport, simplified transport, and diffusion calculations were performed for 33 and 7 neutron energy groups. The neutron cross section library used in both codes is based on the JEFF-3.1 evaluated nuclear data file. The results obtained were compared with those obtained with Serpent.

Several important core parameters were calculated, such as k-eff value, neutron spectrum, neutron flux and power distributions, Doppler constant, the effect of helium density on reactivity, the β-eff, burnup, and the equilibrium cycle. The best agreement with the Serpent results was achieved by using the P_3 transport approximation and 7-group structure. The low density of helium caused large differences between the results for the diffusion and simplified transport calculations compared to the transport calculations. On the other hand, with the 7-group structure, the running time was considerably reduced compared to the 33-group structure and the Monte Carlo calculations. As expected, the shortest running time was obtained for diffusion calculations and a 7-group structure.

The neutron spectrum, flux and power distributions were calculated for all models and showed similar behavior. The Doppler constant was calculated for fuel temperature changes of +100 K and +1000 K, where similar results were obtained in the order of K_D = -550 pcm. The effect of helium density on core reactivity was analyzed by varying the helium pressure from nominal to atmospheric conditions. The major differences compared to the Serpent reference model, were found in the diffusion and simplified transport calculations. The β-eff value was also calculated, obtaining results with good agreement between both simulation codes. The k-eff evolution was determined over an operating time of 365 days. It was found that there is no significant difference between the results by varying the time steps in the burnup calculations. By using 5 steps at 73-day intervals in the burnup calculation, we were able to reduce the running time. The evolution of the main fuel isotopes was also analyzed, the relative fuel composition at the beginning and end of cycle, the BR and the average burnup in the core were calculated. In all these parameters, an excellent agreement was obtained between all the models. A fuel cycle study was carried out in ERANOS using several built-in modules for in-core fuel management. The core equilibrium conditions were determined through a designed reloading and reshuffling scheme. From the simulation of the core evolution over 11 cycles, it was found that equilibrium is reached after the 5th cycle. The diffusion approximation and 7 neutron energy groups were the calculation options used. They were selected to reduce the running time of the core simulation for subsequent optimization calculations. Although these options result in discrepancies compared to the reference model, it is the model that consumes the least computational resources. The potential causes of these differences in the results were identified and analyzed.

The genetic algorithms and tabu search techniques were applied to optimize the fuel loading pattern of the ALLEGRO reactor. The constrained optimization with penalty function approximation was used to transform this multi-objective problem into a single-objective problem. With the formulated objective function, the k-eff value at EOC was maximized, while satisfying the constraints on the power peaking factor over the cycle, the excess reactivity at the BOC, and the linear heat generation rate. To evaluate the objective function and constraints, the black-box approach was used. An interface was developed to allow communication between the optimizer and the ERANOS reactor physics code to simulate loading patterns.

The developed GA code works as a classical one adapted to the proposed combinatorial optimization problem. Two crossover operators, PMX and OX, combined with the random mutation operator, were used to explore the solution search space. To reduce the computational cost, additional features were incorporated into the code. A procedure is used to detect duplicate solutions in the population to avoid re-evaluations and to encourage exploration. In addition, each population is stored so that if a loading pattern reappears, its evaluation can be avoided. As there are no benchmark studies to compare with, an additional objective function was formulated. Thus, by comparing the results using two different objective functions, the performance of the algorithm could be assessed. In the first case, the results were compared using the OF (k-eff at the EOC is maximized) and both crossover operators. Overall, the code performed well, converging towards a good solution in a reasonable computation time. It was found that the algorithm tends to place the assemblies in pairs since a higher k-eff value is obtained. The best result corresponds to the PMX operator, and the running time was twice as short compared to the OX operator. In the second case study, the OF_2 function and the PMX operator were used. The addition of the power distribution flatness term resulted in a decrease in the k-eff value. This behavior occurs because the algorithm tends to place the assemblies towards the periphery of the core to obtain a more uniform power distribution.

Some drawbacks of this technique were also identified. First, the performance of a genetic algorithm was highly dependent on the choice of its parameters, such as the population size, number of generations, crossover and mutation rates, and appropriate selection, crossover, and mutation methods. Setting these parameters was a difficult and time-consuming process. Second, it was noted that the algorithm tends to get stuck in local optima. These are also known as optimum traps, which are essentially valleys or plateaus, observed in the evolution of the best OF value over generations. This disadvantage is highlighted in the literature, especially if the function landscape is complex and, therefore, there are multiple local optima. Small mutations introduced using the mutation operator attempt to prevent falling into these traps in the search for global optima, but sometimes it is not enough. Third, the technique proved to be computationally expensive. Although the high running time is mainly due to the execution of the ERANOS simulation code, the genetic operators also contribute, especially the mutation operator. The mutation is unguided, i.e., random variations are introduced to individuals in the population resulting in slow convergence of the algorithm.

The TS metaheuristic technique was selected to overcome some of the inherent shortcomings of GAs discussed above. An improved version of this technique was successfully implemented. The developed code performed as expected throughout the iterative process. The strength of the TS with respect to GA, and other local search methods, is that current solution is always replaced by its best neighbor, even if it is worse than the current solution. This is the way of not getting stuck in local optima. Additionally, by applying the concepts of the variable tabu list and the aspiration criterion, cycling was avoided, which results in a diversification of the search. The characteristic behavior of this technique was clearly observed in the evolution of the best OF value which is not strictly increasing since worse solutions are allowed. As a result, a better solution was found using this technique with respect to the GA in a reasonable computation time.

The main drawback of the TS was its dependence on the neighborhood size to be explored. Although a small fraction was explored in each iteration, the execution time was high. Furthermore, it should be noted that the running time in both codes depends on the LPs evaluation in the ERANOS.

Recomendations

Although the primary objective of the research was met, the following recommendations are offered based on the work accomplished:

1. The ALLEGRO reactor core model could be improved if more information on the latest design specifications were available. For example, more precise details on the geometry of the core structural elements and control rods. In addition, specifications on reactor operation and fuel cycle would considerably enhance the study.
2. Parallelization of metaheuristic techniques could have considerably reduced the execution time. Both techniques, GA and TS, are suitable for parallel computation. Each candidate solution can be evaluated independently.
3. A more advanced study would be to solve the true multi-objective problem without using the constraint optimization with penalty function approach. Although this method is widely used, it is a simplification of the problem. Both metaheuristic techniques can handle multi-objective problems (e.g. Multi-Objective Genetic Aalgorithm).

4. It would be interesting to implement hybridization, which is a trend in the field of metaheuristics. Not only by combining these techniques with other available metaheuristics, but also by integrating classical optimization methods. The latter are efficient in the local search, so they could reinforce the intensification process.

Publications in Scientific Journals and Conferences Proceedings

Lima-Reinaldo, Y., & François, J.-L. (2022). Fuel loading pattern optimization of ALLEGRO fast reactor using genetic algorithms. *Annals of Nuclear Energy*, *180*, 109451. https://doi.org/10.1016/J.ANUCENE.2022.109451

Lima-Reinaldo, Y., & François, J.-L. (2022). Fuel Loading Pattern Optimization of Allegro Fast Reactor Using Genetic Algorithms. *Proceedings of the International Conference on Physics of Reactors 2022 (PHYSOR 2022).* Pittsburgh, PA, USA.

Lima-Reinaldo, Y., & François, J.-L. (2021). Neutronic analysis of the ALLEGRO fast reactor core with deterministic ERANOS code and Monte Carlo Serpent code. *Annals of Nuclear Energy*, *163*, 108567. https://doi.org/https://doi.org/10.1016/j.anucene.2021.108567

Lima-Reinaldo, Y., & François, J.-L. (2021). *Gestión de combustible dentro del núcleo del reactor ALLEGRO*. *Proceedings of the Mexican Nuclear Society XXXII Annual Congress / 2021 The Latin American Section of the American Nuclear Society (LAS/ANS) Symposium*. Cancun, Quintana Roo, Mexico.

References

Adams, B. M., Bohnhoff, W. J., Dalbey, K. R., Ebeida, M. S., Eddy, J. P., Eldred, M. S., Hooper, R. W., Hough, P. D., Hu, K. T., Jakeman, J. D., Khalil, M., Maupin, K. A., Monschke, J. A., Ridgway, E. M., Rushdi, A. A., Thomas, D., Stephens, J. A., Swiler, L. P., Tran, A., & Winokur, J. G. (2022). *Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.16 User's Manual*.

Arora, J. S. (2017). Introduction to Optimum Design. In J. S. Arora (Ed.), *Introduction to Optimum Design (Fourth Edition)* (Fourth Edi). Academic Press. https://doi.org/https://doi.org/10.1016/C2013-0-15344-5

Arora, R. K. (2015). Optimization: Algorithms and applications. In *Optimization: Algorithms and Applications*. https://doi.org/10.1201/b18469

Bandaru, S., & Deb, K. (2016). Metaheuristic techniques. *Decision Sciences: Theory and Practice*, 693–749. https://doi.org/10.1201/9781315183176

Bianchi, L., Dorigo, M., Gambardella, L. M., & Gutjahr, W. J. (2009). A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, *8*(2), 239–287. https://doi.org/10.1007/s11047-008-9098-4

Castillo, A., Alonso, G., Morales, L. B., Martín-del-Campo, C., François, J. L., & Del Valle, E. (2004). BWR fuel reloads design using a Tabu search technique. *Annals of Nuclear Energy*, *31*(2), 151–161. https://doi.org/10.1016/S0306-4549(03)00214-7

Castillo, A., Ortiz, J. J., Alonso, G., Morales, L., & Del Valle, E. (2005). BWR control rod design using tabu search. *Annals of Nuclear Energy*, *32*(7), 741–754. https://doi.org/10.1016/J.ANUCENE.2004.12.004

Castillo, A., Ortiz, J. J., Montes, J. L., & Perusquía, R. (2007). Fuel loading and control rod patterns optimization in a BWR using tabu search. *Annals of Nuclear Energy*, *34*(3), 207–212. https://doi.org/10.1016/J.ANUCENE.2006.12.006

Čerba, Š., Vrban, B., Lüley, J., Dařílek, P., Zajac, R., Nečas, V., & Haščík, J. (2014). Verification of spectral burn-up codes on 2D fuel assemblies of the GFR demonstrator ALLEGRO reactor. *Nuclear Engineering and Design*, *267*, 148–153.

https://doi.org/10.1016/j.nucengdes.2013.11.065

Čerba, Š., Vrban, B., Lüley, J., Nečas, V., & Haščík, J. (2017). Optimization of the heterogeneous GFR 2400 control rod design. *Progress in Nuclear Energy*, *97*, 170–181. https://doi.org/10.1016/J.PNUCENE.2017.01.009

Cetnar, J., Stanisz, P., & Oettingen, M. (2021). Linear Chain Method for Numerical Modelling of Burnup Systems. *Energies*, *14*(6), 1520. https://doi.org/10.3390/en14061520

de Lima, A. M. M., Schirru, R., da Silva, F. C., & Medeiros, J. A. C. C. (2008). A nuclear reactor core fuel reload optimization using artificial ant colony connective networks. *Annals of Nuclear Energy*, *35*(9), 1606–1612. https://doi.org/10.1016/J.ANUCENE.2008.03.002

DeChaine, M. D., & Feltus, M. A. (1995). Nuclear Fuel Management Optimization Using Genetic Algorithms. *Nuclear Technology*, *111*(1), 109–114. https://doi.org/10.13182/NT95-A35149

Deep, K., Singh, K. P., Kansal, M. L., & Mohan, C. (2009). A real coded genetic algorithm for solving integer and mixed integer optimization problems. *Applied Mathematics and Computation*, *212*(2), 505–518. https://doi.org/10.1016/j.amc.2009.02.044

Domingos, R. P., Schirru, R., & Pereira, C. M. N. A. (2006). Particle Swarm Optimization in Reactor Core Design. *Nuclear Science and Engineering*, *152*(2), 197–203. https://doi.org/10.13182/NSE06-A2575

Dorigo, M, & Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, *1*(1), 53–66. https://doi.org/10.1109/4235.585892

Dorigo, M, Maniezzo, V., & Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *26*(1), 29–41. https://doi.org/10.1109/3477.484436

Dorigo, Marco, Birattari, M., & Stutzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine*, *1*(4), 28–39. https://doi.org/10.1109/MCI.2006.329691

Dréo, J., Pétrowski, A., Siarry, P., & Taillard, E. (2006). *Metaheuristics for Hard Optimization: Methods and Case Studies* (1st ed.). Springer Berlin, Heidelberg. https://doi.org/https://doi.org/10.1007/3-540-30966-7

Du, K.-L., & Swamy, M. N. S. (2016). *Search and Optimization by Metaheuristics: Techniques and Algorithms Inspired by Nature*. Springer International Publishing. https://doi.org/10.1007/978-3-319-41192-7

Eddy, J., & Lewis, K. (2001). Effective Generation of Pareto Sets Using Genetic Programming. *Proceedings of ASME 2001 Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Volume 2B: 27th Design Automation Conference*, 783–791. https://doi.org/https://doi.org/10.1115/DETC2001/DAC-21094

Esquivel-Estrada, J., Ortiz, J. J., Castillo, A., & Perusquía, R. (2011). Azcaxalli: A system based on Ant Colony Optimization algorithms, applied to fuel reloads design in a Boiling Water Reactor. *Annals of Nuclear Energy*, *38*(1), 103–111. https://doi.org/10.1016/j.anucene.2010.08.011

Fogel, L. J., Owens, A. J., & Walsh, M. J. (1966). Artificial intelligence through simulated evolution. In *Artificial intelligence through simulated evolution.* John Wiley & Sons.

François, J. L., Martín-del-Campo, C., François, R., & Morales, L. B. (2003). A practical optimization procedure for radial BWR fuel lattice design using tabu search with a multiobjective function. *Annals of Nuclear Energy*, *30*(12), 1213–1229. https://doi.org/10.1016/S0306-4549(03)00055-0

François, J. L., Ortiz, J. J., Martín-del-Campo, C., Castillo, A., & Esquivel-Estrada, J. (2013). Comparison of metaheuristic optimization techniques for BWR fuel reloads pattern design. *Annals of Nuclear Energy*, *51*, 189–195. https://doi.org/10.1016/J.ANUCENE.2012.08.014

Generation IV International Forum (GIF). (2018). *GIF R&D Outlook for Generation IV Nuclear Energy Systems: 2018 Update*.

Generation IV International Forum (GIF). (2021). *Annual Report*.

Gharari, R., Poursalehi, N., Abbasi, M., & Aghaie, M. (2016). Implementation of Strength Pareto Evolutionary Algorithm II in the Multiobjective Burnable Poison Placement Optimization of KWU Pressurized Water Reactor. *Nuclear Engineering and Technology*, *48*(5), 1126–1139. https://doi.org/10.1016/j.net.2016.04.004

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, *13*(5), 533–549. https://doi.org/10.1016/0305-0548(86)90048-1

Glover, F. (1989). Tabu Search - Part I. *ORSA Journal on Computing*, *1*(3), 190–206. https://doi.org/10.1287/ijoc.1.3.190

Glover, F. (1990). Tabu Search - Part II. *ORSA Journal on Computing*, *2*(1), 4–32. https://doi.org/10.1287/ijoc.2.1.4

Glover, F., & Laguna, M. (1997). *Tabu Search*. Springer New York, NY. https://doi.org/https://doi.org/10.1007/978-1-4615-6089-0

Goldberg, D. E. (1989). *Genetic Algorithm in Search, Optimization, and Machine Learning*. Addison-Wesley.

Grasso, G., Alemberti, A., Doderlein, C., & Tucek, K. (2013). *Definition of the LFR core and neutronic characterization*.

Haupt, R. L., & Haupt, S. E. (2004). Practical genetic algorithms. In *Practical Genetic Algorithms*. Wiley-Blackwell. https://doi.org/10.1002/0471671746

Hill, N. J., & Parks, G. T. (2015). Pressurized water reactor in-core nuclear fuel management by tabu search. *Annals of Nuclear Energy*, *75*, 64–71. https://doi.org/10.1016/j.anucene.2014.07.051

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press.

Hussain, A., Muhammad, Y. S., Nauman Sajid, M., Hussain, I., Mohamd Shoukry, A., & Gani, S. (2017). Genetic Algorithm for Traveling Salesman Problem with Modified Cycle Crossover Operator. *Computational Intelligence and Neuroscience*, *2017*, 1–7. https://doi.org/10.1155/2017/7430125

Inc., T. M. (2019). *MATLAB R2019b*. The MathWorks Inc. https://www.mathworks.com

Jagawa, S., Yoshii, T., & Fukao, A. (2001). Boiling Water Reactor Loading Pattern Optimization Using Simple Linear Perturbation and Modified Tabu Search Methods. *Nuclear Science and Engineering*, *138*(1), 67–77. https://doi.org/10.13182/NSE00-44

Jayalal, M. L., Murty, S. A. V. S., & Baba, M. S. (2014). A Survey of Genetic Algorithm Applications in Nuclear Fuel Management. *Journal of Nuclear Engineering & Technology*, *4*(1), 45–62.

Kaltiaisenaho, T. (2014). Statistical Tests and the Underestimation of Variance in Serpent 2. In *VTT-R-00371-14*. http://montecarlo.vtt.fi/download/VTT-R-00371-14.pdf

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, *39*(3), 459–471. https://doi.org/10.1007/s10898-007-9149-x

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*, *4*, 1942–1948 vol.4.

https://doi.org/10.1109/ICNN.1995.488968

Khoshahval, F., Zolfaghari, A., Minuchehr, H., Sadighi, M., & Norouzi, A. (2010). PWR fuel management optimization using continuous particle swarm intelligence. *Annals of Nuclear Energy*, *37*(10), 1263–1271. https://doi.org/10.1016/j.anucene.2010.05.023

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, *220*(4598), 671–680. https://doi.org/10.1126/science.220.4598.671

Kochenderfer, M. J., & Wheeler, T. A. (2019). *Algorithms for Optimization*. The MIT Press.

Krepel, J., Pelloni, S., Mikityuk, K., & Coddington, P. (2010). GFR equilibrium cycle analysis with the EQL3D procedure. *Nuclear Engineering and Design*, *240*(4), 905–917. https://doi.org/10.1016/j.nucengdes.2009.12.007

Kropaczek, D. J., & Turinsky, P. J. (1991). In-Core Nuclear Fuel Management Optimization for Pressurized Water Reactors Utilizing Simulated Annealing. *Nuclear Technology*, *95*(1), 9–32. https://doi.org/10.13182/NT95-1-9

Larrañaga, P., Kuijpers, C. M. H., Murga, R. H., Inza, I., & Dizdarevic, S. (1999). Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators. *Artificial Intelligence Review*, *13*(2), 129–170. https://doi.org/10.1023/A:1006529012972

Leppänen, J. (2007). Development of a new Monte Carlo reactor physics code [Helsinki University of Technology]. In *VTT Publications* (Issue 640). http://montecarlo.vtt.fi/download/P640.pdf

Leppänen, J. (2015). *Serpent: a Continuous - energy Monte Carlo Reactor Physics Burnup Calculation Code*. http://montecarlo.vtt.fi/download/Serpent_manual.pdf

Leppänen, J., Pusa, M., Viitanen, T., Valtavirta, V., & Kaltiaisenaho, T. (2015). The Serpent Monte Carlo code: Status, development and applications in 2013. *Annals of Nuclear Energy*, *82*, 142–150. https://doi.org/10.1016/j.anucene.2014.08.024

Lima-Reinaldo, Y., & François, J.-L. (2021). Neutronic analysis of the ALLEGRO fast reactor core with deterministic ERANOS code and Monte Carlo Serpent code. *Annals of Nuclear Energy*, *163*, 108567. https://doi.org/https://doi.org/10.1016/j.anucene.2021.108567

Lima-Reinaldo, Y., & François, J.-L. (2022). Fuel loading pattern optimization of ALLEGRO fast reactor using genetic algorithms. *Annals of Nuclear Energy*, *180*, 109451. https://doi.org/10.1016/J.ANUCENE.2022.109451

Lima-Reinaldo, Y., François, J.-L., & Martín-del-Campo, C. (2019). Analysis of the use of thorium in the GFR2400 gas-cooled fast reactor. *Nuclear Engineering and Design*, *343*, 11–21. https://doi.org/https://doi.org/10.1016/j.nucengdes.2018.12.016

Martín-del-Campo, C., Palomera-Pérez, M. Á., & François, J. L. (2009). Advanced and flexible genetic algorithms for BWR fuel loading pattern optimization. *Annals of Nuclear Energy*, *36*(10), 1553–1559. https://doi.org/10.1016/J.ANUCENE.2009.07.013

Meneses, A. A. de M., Gambardella, L. M., & Schirru, R. (2010). A new approach for heuristics-guided search in the In-Core Fuel Management Optimization. *Progress in Nuclear Energy*, *52*(4), 339–351. https://doi.org/10.1016/J.PNUCENE.2009.07.007

Meneses, A. A. de M., Machado, M. D., & Schirru, R. (2009). Particle Swarm Optimization applied to the nuclear reload problem of a Pressurized Water Reactor. *Progress in Nuclear Energy*, *51*(2), 319–326. https://doi.org/10.1016/j.pnucene.2008.07.002

Nissan, E. (2019). An Overview of AI Methods for in-Core Fuel Management: Tools for the Automatic Design of Nuclear Reactor Core Configurations for Fuel Reload, (Re)arranging New and Partly Spent Fuel. *Designs*, *3*(3), 37. https://doi.org/10.3390/designs3030037

Oettingen, M. (2021). Assessment of the Radiotoxicity of Spent Nuclear Fuel from a Fleet of PWR Reactors. *Energies*, *14*(11). https://doi.org/10.3390/en14113094

Ortiz, J. J., Castillo, A., Montes, J. L., & Perusquía, R. (2007). A New System to Fuel Loading and Control Rod Pattern Optimization in Boiling Water Reactors. *Nuclear Science and Engineering*, *157*(2), 236–244. https://doi.org/10.13182/NSE07-A2725

Ortiz, J. J., & Requena, I. (2004). An Order Coding Genetic Algorithm to Optimize Fuel Reloads in a Nuclear Boiling Water Reactor. *Nuclear Science and Engineering*, *146*(1), 88–98. https://doi.org/10.13182/NSE04-A2395

Palmiotti, G., Assawaroongruengchot, M., Salvatores, M., Herman, M., Oblozinsky, P., Mattoon, C., & Pigni, M. (2011). Nuclear Data Target Accuracies for Generation-IV Systems Based on the Use of New Covariance Data. *Journal of the Korean Physical Society*, *59*(2(3)), 1264–1267. https://doi.org/10.3938/jkps.59.1264

Palmiotti, G., Lewis, E. E., & Carrico, C. B. (1995). *VARIANT: VARIational Anisotropic Nodal Transport for Multidimensional Certesian and Hexagonal Geometry Calculation*.

Parisot, J. F. (2015). *Neutronics*. CEA.

Park, T. K., Joo, H. G., Kim, C. H., & Lee, H. C. (2009). Multiobjective Loading Pattern Optimization by Simulated Annealing Employing Discontinuous Penalty Function and

Screening Technique. *Nuclear Science and Engineering*, *162*(2), 134–147. https://doi.org/10.13182/NSE162-134

Perkó, Z. (2012). *GFR2400 core neutronics and thermal-hydraulics characterization*. Delft University of Technology.

Perkó, Z., Pelloni, S., Mikityuk, K., Křepel, J., Szieberth, M., Gaëtan, G., Vrban, B., Lüley, J., Čerba, Š., Halász, M., Fehér, S., Reiss, T., Leen Kloosterman, J., Stainsby, R., & Poette, C. (2015). Core neutronics characterization of the GFR2400 Gas Cooled Fast Reactor. *Progress in Nuclear Energy*, *83*, 460–481. https://doi.org/10.1016/j.pnucene.2014.09.016

Pónya, P., & Czifrus, S. (2017). Core optimisation issues of MOX fueled ALLEGRO reactor. *Annals of Nuclear Energy*, *108*, 188–197. https://doi.org/https://doi.org/10.1016/j.anucene.2017.05.001

Poursalehi, N., Zolfaghari, A., & Minuchehr, A. (2013). PWR loading pattern optimization using Harmony Search algorithm. *Annals of Nuclear Energy*, *53*, 288–298. https://doi.org/10.1016/J.ANUCENE.2012.06.037

Rimpault, G. (1997). *ERANOS 2.3 Documentation: Physics documentation of ERANOS, the ECCO cell code*.

Rudolph, G. (2012). Evolutionary Strategies. In G. Rozenberg, T. Bäck, & J. N. Kok (Eds.), *Handbook of Natural Computing* (pp. 673–698). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-92910-9_22

Ruggieri, J. M., Tommasi, J., Lebrat, J. F., Suteau, C., Plisson-Rieunier, D., De Saint Jean, C., Rimpault, G., & Sublet, J. C. (2006). ERANOS 2.1: International code system for GEN IV fast reactor analysis. *Proceedings of the 2006 International Congress on Advances in Nuclear Power Plants*, 2432–2439.

So, C., Ho, I. M., Chae, J. S., & Hong, K. H. (2021). PWR core loading pattern optimization with adaptive genetic algorithm. *Annals of Nuclear Energy*, *159*, 108331. https://doi.org/10.1016/J.ANUCENE.2021.108331

Stacey, W. M. (2018). *Nuclear Reactor Physics* (3rd ed.). John Wiley & Sons Ltd.

Stainsby, R., Peers, K., Mitchell, C., Poette, C., Mikityuk, K., & Somers, J. (2011). Gas cooled fast reactor research in Europe. *Nuclear Engineering and Design*, *241*(9), 3481–3489. https://doi.org/10.1016/j.nucengdes.2011.08.005

Stanisz, P., Oettingen, M., & Cetnar, J. (2016). Monte Carlo modeling of Lead-Cooled Fast

Reactor in adiabatic equilibrium state. *Nuclear Engineering and Design*, *301*, 341–352. https://doi.org/https://doi.org/10.1016/j.nucengdes.2016.02.025

Stevens, J. G., Smith, K. S., Rempe, K. R., & Downar, T. J. (1995). Optimization of Pressurized Water Reactor Shuffling by Simulated Annealing with Heuristics. *Nuclear Science and Engineering*, *121*(1), 67–88. https://doi.org/10.13182/NSE121-67

Toshinsky, V. G., Sekimoto, H., & Toshinsky, G. I. (1999). Multiobjective fuel management optimization for self-fuel-providing LMFBR using genetic algorithms. *Annals of Nuclear Energy*, *26*(9), 783–802. https://doi.org/10.1016/S0306-4549(98)00092-9

Tran, V.-P., Phan, G. T. T., Hoang, V.-K., Ha, P. N. V., Yamamoto, A., & Tran, H.-N. (2021). Evolutionary simulated annealing for fuel loading optimization of VVER-1000 reactor. *Annals of Nuclear Energy*, *151*, 107938. https://doi.org/https://doi.org/10.1016/j.anucene.2020.107938

Turinsky, P. J. (2005). Nuclear Fuel Management Optimization: A Work in Progress. *Nuclear Technology*, *151*(1), 3–8. https://doi.org/10.13182/NT05-A3626

U.S. DOE. (2002). *A Technology Roadmap for Generation IV Nuclear Energy Systems*.

Waltar, A. E., Todd, D. R., & Tsvetkov, P. V. (2012). *Fast Spectrum Reactors*. Springer US. https://doi.org/10.1007/978-1-4419-9572-8

Wu, S. C., Chan, T. H., Hsieh, M. S., & Lin, C. (2016). Quantum evolutionary algorithm and tabu search in pressurized water reactor loading pattern design. *Annals of Nuclear Energy*, *94*, 773–782. https://doi.org/10.1016/J.ANUCENE.2016.04.039

Yadav, R. D. S., & Gupta, H. P. (2011). Optimization studies of fuel loading pattern for a typical Pressurized Water Reactor (PWR) using particle swarm method. *Annals of Nuclear Energy*, *38*(9), 2086–2095. https://doi.org/10.1016/J.ANUCENE.2011.05.019

Yamamoto, A. (1998). *Study on Advanced In-Core Fuel Management for Pressurized Water Reactors Using Loading Pattern Optimization Methods* [Kyoto University]. http://hdl.handle.net/2433/156982

Yang, X.-S. (2009). Firefly Algorithms for Multimodal Optimization. In O. Watanabe & T. Zeugmann (Eds.), *Stochastic Algorithms: Foundations and Applications* (pp. 169–178). Springer Berlin Heidelberg.

Yang, X.-S., & Deb, S. (2009). Cuckoo Search via Lévy flights. *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, 210–214.

https://doi.org/10.1109/NABIC.2009.5393690