Development of Genetic Algorithms for Plant Reload Optimization for an Operating Pressurized Water Reactor

Eunseo So[a], Yunyeong Heo[bf], Mohammad G. Abdo[c†], Andrea Alfonsi[d], and Yong-Joon Choi[e]

[a] (Former) Idaho National Laboratory, Idaho Falls, ID, U.S., soes0820@gmail.com
[b] Idaho National Laboratory, Idaho Falls, ID, U.S., yunyeong.heo@inl.gov
[c] Idaho National Laboratory, Idaho Falls, ID, U.S., mohammad.abdo@inl.gov
[d] (Former) Idaho National Laboratory, Idaho Falls, ID, U.S., a.alfonsi@usnc-tech.com
[e] Idaho National Laboratory, Idaho Falls, ID, U.S., yong-joon.choi@inl.gov
[f] Ulsan National Institute of Science and Technology, Ulsan, South Korea, yyheo0207@unist.ac.kr
[†] *Corresponding author*

Abstract: This paper summarizes the development, results, and enhancement activity of the Artificial Intelligence (AI) based automated nuclear power plant fuel reload optimization platform under the guidance of the United States Department of Energy, Light Water Reactor Sustainability Program, under the Risk-Informed Systems Analysis Pathway. The research focuses on the optimization of the fuel arrangement to maximize fuel utilization. The AI-based Genetic Algorithms work with both convex and non-convex, constrained or unconstrained problems. This can help explain the relationship between the fuel arrangement and fuel cycle length, in particular, the surrogate models used to reconstruct the Multiphysics problem maps the features/inputs of the problem to the fuel cycle length to provide such an explanation. The Genetic Algorithm is composed of several evolutionary processes: fitness evaluation, parent selection, crossover, mutation, survivor selection, and termination. Crossover and mutation are the main steps responsible for injecting randomness/heuristics to prevent the algorithm from getting stuck in local minima. In this paper, roulette wheel parent selection, one-point crossover, swap mutation, and fitness-based survivor selection are used for demonstration to convert the fuel arrangement problem from the physical world (phenotype space) to the computational word (genotype space) via a user performed encoding/decoding step. Here, the search variables (genes) are the fuel locations in the core, whereas the values each variable takes, represent the fuel identification that will be placed in that specific location. The optimization process was demonstrated with a quarter core initial loading problem. In the core, 56 locations are loaded with five types of fuel assemblies, each type has different amount of enrichment and burnable poisons. As a result, the fuel cycle length increased to over 590 days, which is very close to the expected value. The results and enhancements in the optimization algorithm are also discussed in this paper.

---

1. INTRODUCTION

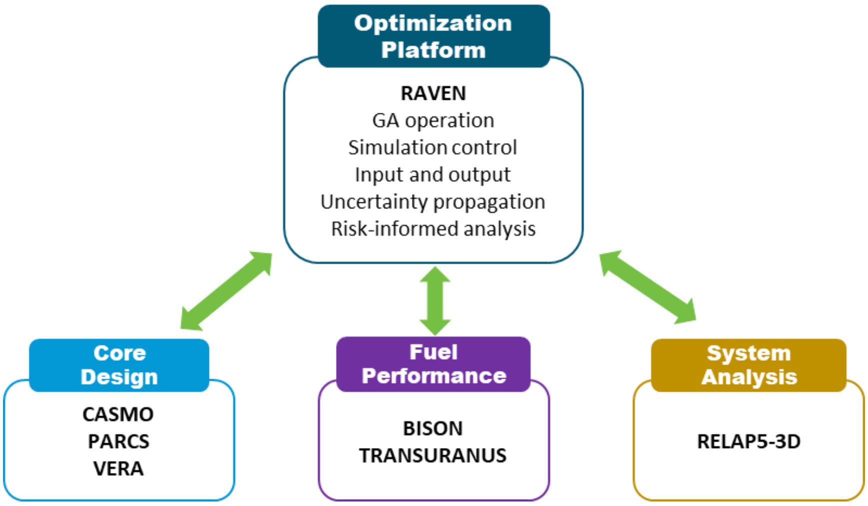
In U.S., as of 2019, the fuel represented about 20% of the total generating cost. The cost of typical fuel reload for a light water reactor (LWR) is about 50 million dollars [1]. To reduce the fuel cost, loading pattern optimization has been one of the most important considerations to reduce the amount of fuel used in the core. However, loading pattern cannot be optimized by itself. Fuel performance analysis and system analysis results also need to be considered to determine the loading pattern, which leads to long computational time. In this regard, artificial intelligence (AI) based genetic algorithm can largely reduce this burden using the physics code results as verification stage.

There were other methods that showed similar or better performance under specific cases for loading pattern optimization. However, the genetic algorithm can generate multiple solutions to the optimization problem, so that it has maintained its reputation as a reference method in the loading pattern optimization fields [2]. In a previous study on pin lattice optimization, the authors compared the performance of five heuristic optimization methods: ant colony system, artificial neural networks, genetic algorithm, greedy search, and a hybrid of path relinking and scatter search. The authors proposed that genetic algorithm and path relinking coupled to scatter search shows the best results in terms of global cost [3].

In this regard, a modular optimization framework (MOF) was proposed in a previous study. MOF aims to facilitate the application of other optimization methods as well as the genetic algorithm using object-oriented programming [4]. Meanwhile, in this paper, Risk Analysis and Virtual Environment (RAVEN) [5] was utilized as a main controller as well as the reloading pattern optimization platform. RAVEN's capability is not just limited to optimization, but it can also provide input decks to other physical codes and perform post-processing of the simulation results. This extensibility of RAVEN facilitates the coupling with other physical codes such as RELAP5-3D, which can lead to creating a unified framework considering physical phenomena. For example, thermal-hydraulic analysis and fuel performance analysis results can be used to identify if the optimized core meets the safety requirements or not.

Therefore, plant reload optimization project has launched as a part of U.S. Department of Energy (DOE) Light Water Reactor Sustainability (LWRS) program Risk-Informed Systems Analysis (RISA) pathway [1]. As shown in Figure 1, the framework developed in this project consists of several elements: core design, system analysis, fuel performance analysis as well as the loading pattern optimization platform implemented in RAVEN. The entire framework is explained in more detail in a separate paper [6]. This paper is more focused on the methods and demonstration of the loading pattern optimization platform.

Figure 1: Schematic Diagram of Plant Reload Optimization Platform.



2. GENETIC ALGORITHM

2.1. Overview of Genetic Algorithms

Most of the terminologies in genetic algorithms are inspired from evolutionary theory in biology. For clear explanation, terminologies of genetic algorithm are described in Table 1 and schematic diagram is shown in Figure 2. Chromosome, comprised of several genes, is a fundamental unit to reflect the object to optimize. Allele can be any type of data for each gene, binary, integer, real number, or values from discrete or continuous distributions. RAVEN [7] has capability for assigning different distributions to each allele.

In the loading pattern optimization, for example, different types of fuel rods can be represented as alleles, and they can be assigned to spatial location which can be represented as genes. The arrangement of different types of fuel rod in the core, core configuration, can be represented as chromosome. To find optimal solution/chromosomes, i.e., optimal core configuration, the chromosomes are included in the population to undergo optimization using genetic algorithm. As described in the example above, the real problem solution space, phenotype space, can be encoded to computational space, genotype space.

Table 1: Terminology of Genetic Algorithm

Terminology	Description	Loading pattern optimization
Phenotype space	The actual real problem solution space, comprising of solutions in the raw (non-computational) representations.	
Genotype space	The computational space comprising of all candidate solutions after encoding to a computational representation.	
Decoding and encoding	The optional process to convert phenotype representation (real variables) into genotype (computational) representation.	
Population	A subset of all candidate solutions in the genotype (encoded) space.	Pool of possible solution (fuel rod arrangement)
Chromosomes (Individuals)	A single possible solution of the problem at hand taken from that population.	Fuel rod arrangement
Gene	A single element in the chromosome.	Location of fuel rod
Allele	The value in the gene.	Different types of fuel
Mating/Reproduction pool	A collection of parents used to create a new generation.	
Fitness function	The function used to rank the solutions (elitism). It might or might not be the same as the objective function.	
Reproduction operation/operators	Operations that alter the composition of a certain chromosome, i.e., crossover, mutation, and selection.	

Figure 2: Schematic Diagram of Genetic Algorithm

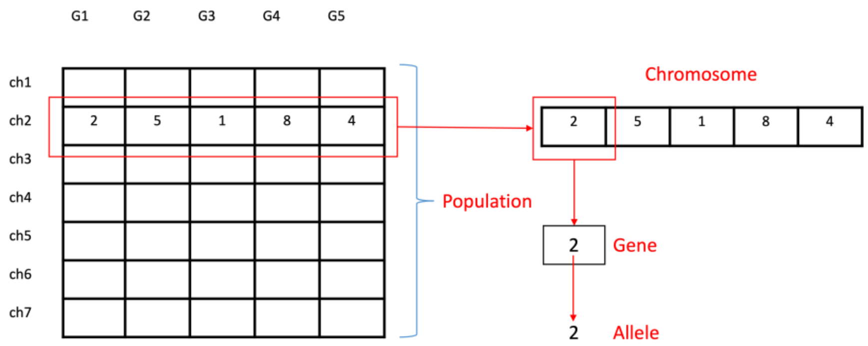
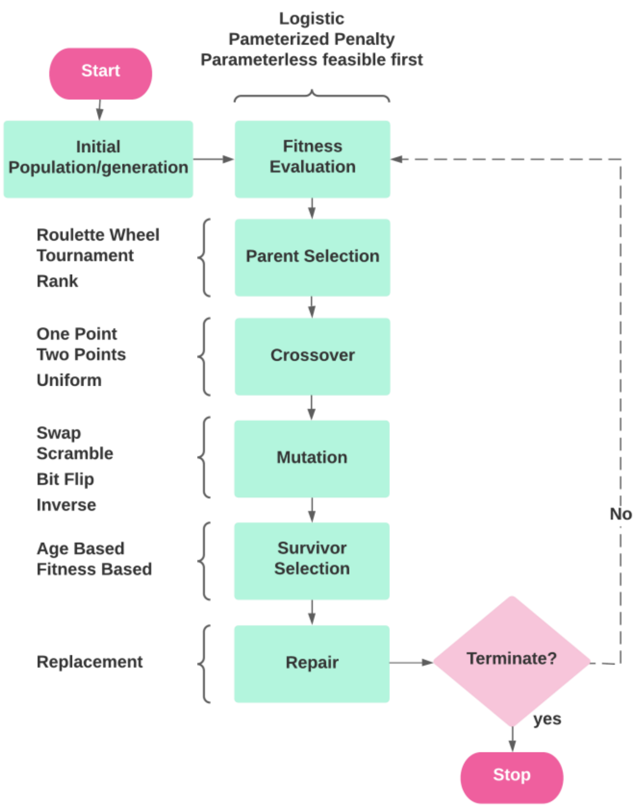


Figure 3 shows the optimization process using genetic algorithm step by step. First of all, the optimization starts with an initial population composed of several arbitrary chromosomes. Then the fitness of each chromosome is evaluated. To make new offspring, two chromosomes in the population are selected as parents and crossover and mutation are performed. Then the newly born offspring are included in the new population. Among the chromosomes in the population including the old chromosomes and new offspring from selected parents, survivors should be selected to develop the next population. If there is any constraint violated, the chromosomes can be replaced or repaired to meet the requirements. This entire process, from fitness evaluation to repair, is repeated until the final chromosome meets the termination criterion or iteration limit.

2.2. Fitness Evaluation

Fitness evaluation is an essential element of the genetic algorithm which quantifies how the different solutions/chromosomes at the current iteration/population meet the design and performance goals and hence ranks the solutions accordingly.

Figure 3: Genetic Algorithm Flow Chart



Specifically, fitness is usually a function of the objective function and constraints and is sought to be maximized as the solution reaches the optimal solution. To achieve such functionality, fitness algorithms should satisfy several requirements such as:

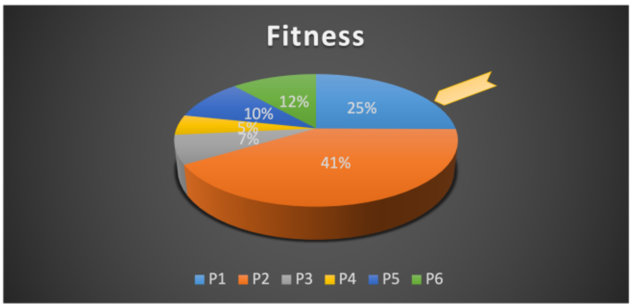
1. Evaluation function should be efficiently implemented, and fast to compute.
2. Clearly defined such that the best and worst candidate should have the best and worst fitness scores respectively.
3. Maximum fitness in the last population will be declared as the best candidate.
4. As a candidate violates a constraint its fitness should become worse (less) to make it less likely to be selected in the next generations.

In the Genetic algorithm implementation in RAVEN, several finesses have been considered including an inverse Linear fitness that is a linear combination of the objective function and the penalty in case any explicit and/or implicit constraints are violated. Another available fitness was inspired by [8] and was called `feasibleFirst`.

2.3 Parent Selection

To generate offsprings in the genetic algorithm, elitist parents should be selected before they enter the mating pool for the next generation based on specific rules. There are several algorithms implemented for parent selection including roulette wheel, rank selection, tournament selection, etc. For example, the roulette wheel algorithm assigns the probability that each chromosome is selected based on their fitness. These algorithms are already implemented in RAVEN so that analysts can pick any algorithm or build their own algorithm for their specific problem. Figure 4 illustrates the roulette wheel algorithm. The roulette wheel algorithm assigns each parent to a portion of the wheel dictated by the fitness score of that parent, i.e., the area assigned to that parent is just the fitness divided by the sum of fitness. And while that gives a larger probability to the highest fittest individuals, the pointer of the wheel is randomly rotated and can still select a different parent. This helps avoiding getting stuck in local minima and keep selecting the same parents over and over.

Figure 4: Roulette wheel parent selection.



Individual	Fitness
P1	5
P2	8.2
P3	1.4
P4	0.98
P5	2
P6	2.3

2.4. Crossover

After the parents are selected, crossover operation generates the offspring from the parents. For instance, as a chromosome is made up of a series of genes, two parent chromosomes can exchange their genes at a specific location(s) to form the offsprings. Again, several algorithms are implementd including One point cross over, two points crossover, and uniform crossover. Figure 5 shows the one-point crossover as an example of the crossover evolutionary operation.

Figure 5: One-Point crossover

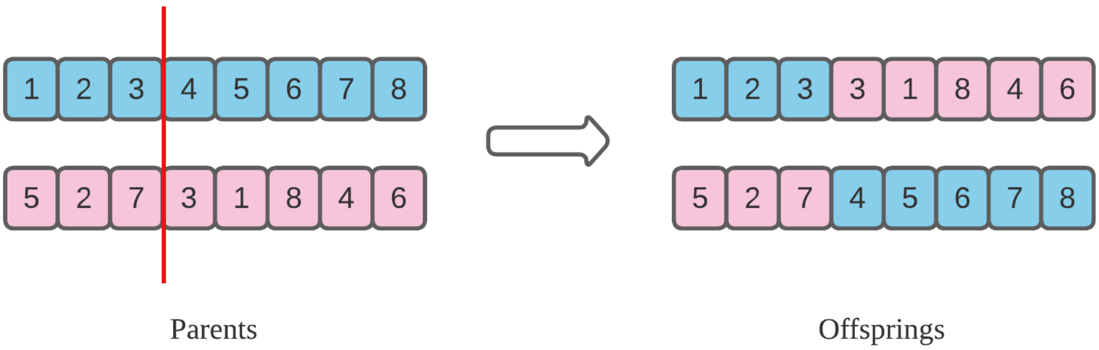
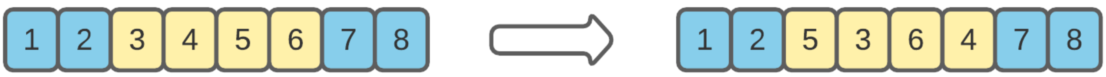


Figure 6: Scramble Mutation



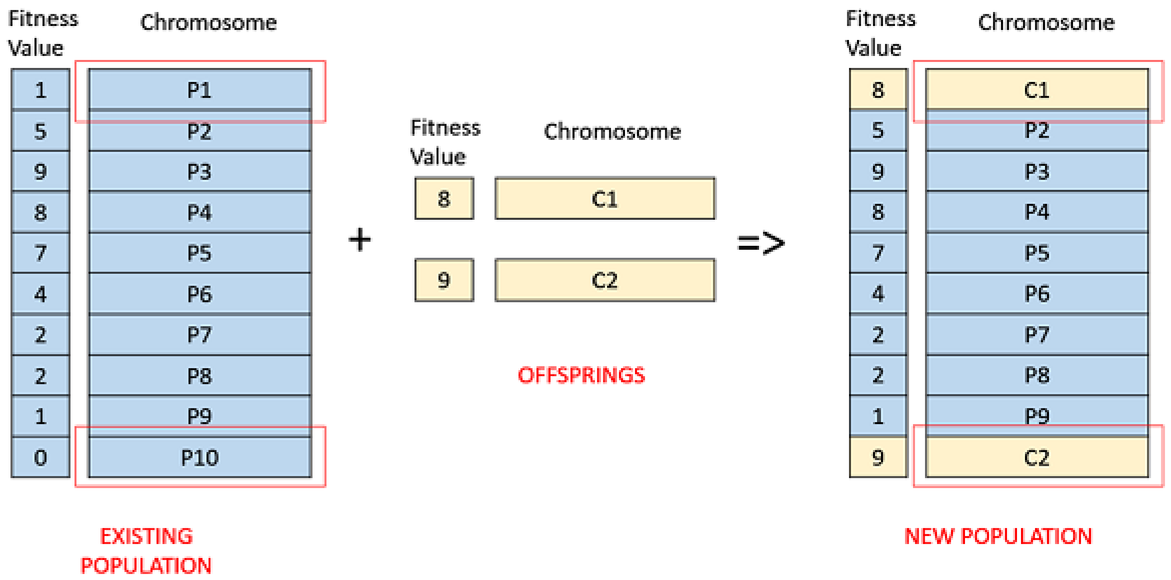
2.5. Mutation

After the generation of offspring, the genes in each chromosome can also be randomly altered. Several mutation algorithms are implements to offer the user a vast spectrum of options to perform mutations such as swap mutation, scramble mutation, inversion mutation, and bit flip mutation. The randomness induced by these algorithms helps escape from valleys and local minima in nonconvex problems. Figure 6 depicts one of these mutation mechanisms namely the scramble mutation.

2.6. Survivor selection and repair

To fill the population with the most elite chromosomes, the algorithm selects which chromosomes will survive in the next generation. For example, in fitness-based parent selection, a few chromosomes with lower fitness can be removed from the population and replaced with fitter children, so that the next generation has higher overall fitness. Once survivors are identified, a reoair process is carried out to make sure the crossover and/or mutation did not cause any violations, for instance if the problem should not allow repeated values or the fuel inventory allows for limited number of a certain fuel id, but the crossover lead to violating these design specs, a repair process should fix these issues by replacing the violating gene by a random value withdrawn from the associate discrete distribution [1]. Figure 7 outlines the fitness based survivor selection mechanism.

Figure 5: Fitness-based survivor selection



2.7. Convergence Criterion

After going through numerous iterations to find the optimum chromosome, specific criteria should be determined to terminate the iterations. The genetic algorithm can have multiple termination criteria. For example, the iteration can be stopped (1) if it reaches the maximum number of iterations or (2) if it meets a specific termination criterion such as p-averaged Hausdorff distance between two consecutive generations [1].

2.8. Optimization Methodology Enhancement

Unlike gradient-based methods, metaheuristic methods such as Genetic algorithms exploit randomness to pick the next candidates. On one hand this reduces the likeliness of getting stuck in local minima, but on the other hand will cause slower convergence. This problem is magnified for Multiphysics problems such as the problem at hand. Multiple enhancements are suggested to mitigate this issue. In this subsection, enhancements are outlined to demonstrate the upcoming directions of this work.

Ideas to enhance the optimization framework include - but are not limited to – exploring active subspaces hosting search directions that increase the fitness and then contain the evolutionary operations (i.e., crossover, mutation, etc.) to pick the offspring from those subspaces [9]. Another potential enhancement is to accelerate the genetic algorithm using Markov Chains [10]. Moreover, hybridization with other metaheuristic optimizers such as Particle Swarm, Ant Colony, and Bee algorithms has proven to offer deep enhancement to the optimization framework [11]. Finally, recasting the problem as a reinforcement learning problem can be used to compare and benchmark the results from the genetic algorithms [12].

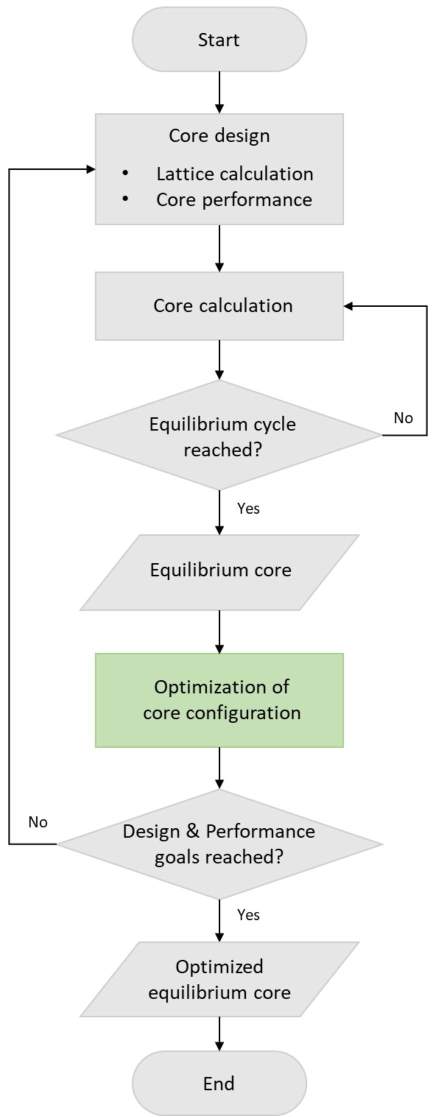
3. DEMONSTRATION OF PLANT RELOAD OPTIMIZATION FRAMEWORK

3.1. Core Design Optimization Process

The genetic algorithm is the optimization method involved in the plant reload optimization framework. The framework is composed of system analysis, core design, fuel performance analysis, and optimization platform as shown in Figure 1. The components are connected closely to each other, and exchange information required for calculation. More detailed explanation of the plant reload optimization framework is presented in a separate paper [6] as mentioned above.

This paper is more focused on the optimization platform using RAVEN [5]. The core design process is shown in the flowchart in Figure 8. The process starts with initial guess core configuration. In this project, a nodal code SIMULATE-3 [13] and a lattice code CASMO-4E [14] are used for reactor core simulation and cross section calculation.

Figure 8: Flowchart of Core Design



Based on the initial core configuration, equilibrium core or cycle can be computed after multi-cycle analysis for 3 refueling cycles. If the fuel reloading pattern (i.e., same composition and spatial loading of the fuel batches) remains almost constant, it can be considered as the equilibrium cycle [1]. In this paper, it is assumed that the equilibrium cycle is reached after 8th reload.

Then this equilibrium cycle can be used as a starting point for the optimization of the core configuration. The optimization is performed using the genetic algorithm described in Section 2, and more detailed information about the optimization condition and results are explained in Section 3.2 with a toy problem for demonstration. After the optimization, the optimized core configuration is analyzed if it satisfies design and performance goals. For example, the design goal can be minimization of fuel load and the performance goal can be desired cycle length.

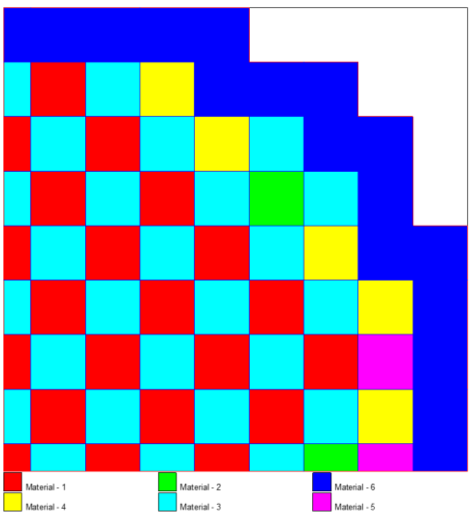
If the optimized core configuration is required to be adjusted to meet the design and performance goals, the equilibrium core should also be computed again owing to the design differences. Even though the design and performance goals were satisfied with the equilibrium core before optimization, the layout of the assemblies will change during the optimization process. For example, the assemblies will face different neighbors, different enrichments, etc. Therefore, the cross section associated with those assemblies will also change, which is required to be recalculated. In other words, equilibrium cycle requires (1) core design which meets the design and performance goals and (2) cross section for the core design. If the optimization of core configuration makes any difference to meet these requirements, then it should be recalculated until it converges as shown in the flowchart in Figure 8.

3.2. Demonstration Results

In this paper, demonstration of optimization method using genetic algorithm is performed on an ¼ core initial loading problem. As shown in Figure 9, there are 5 types of fuel assemblies which will be loaded to 56 locations. In terms of genetic algorithm, a core configuration (i.e., chromosome) is loaded with fuel assemblies located at each location (i.e., genes). The composition of five types of material is presented as follows:

- Material 1 – Enrichment 2.2% in U-235, no burnable poisons
- Material 2 – Enrichment 2.5% in U-235, no burnable poisons
- Material 3 – Enrichment 2.5% in U-235, burnable poisons (8.0E-6 #/cm·barn)
- Material 4 – Enrichment 3.5% in U-235, no burnable poisons
- Material 5 – Enrichment 3.5% in U-235, burnable poisons (8.0E-6 #/cm·barn)

Figure 9: Layout of possible materials in quarter core



The design goal of the optimization is minimization of the fuel load. The performance goals are (1) desired cycle length and (2) minimizing the radial and axial power peaking to make it less than 1.3. Additional performance goals can be set as constraints by safety bases from fuel performance analysis results such as maximum oxidation or soluble boron concentration, however the former two performance goals are only considered for the demonstration in this paper.

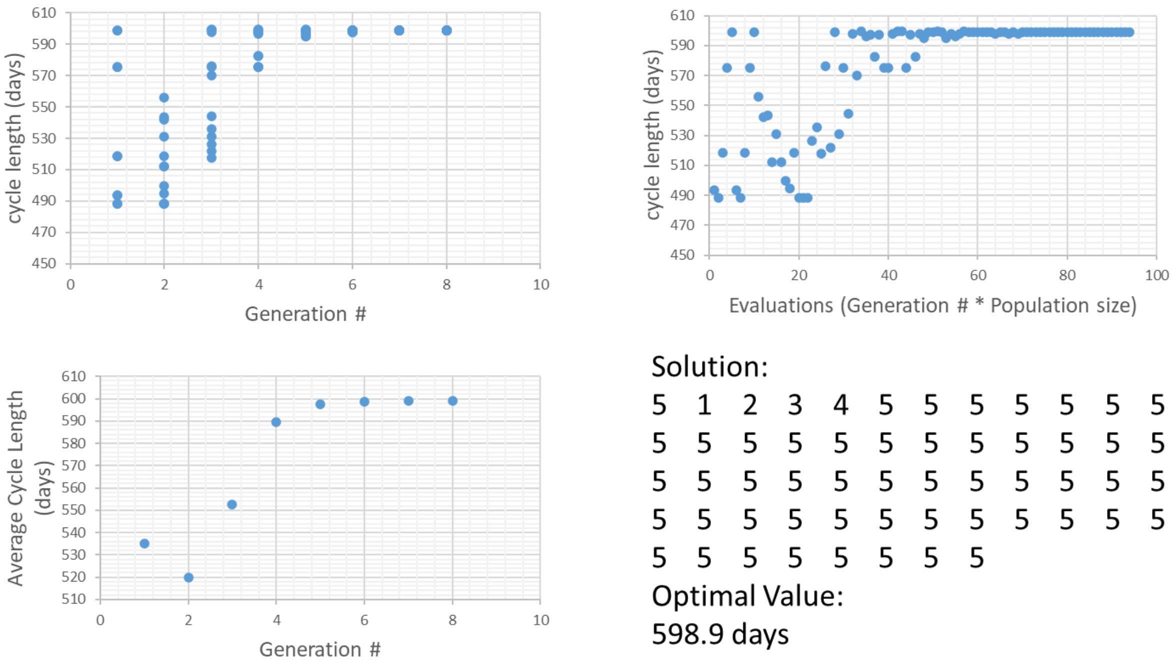
In this paper, an objective function is set to maximization of the cycle length to meet the design goal which is minimizing the amount of fuel required to be used to meet the performance goals. To find an optimum core configuration (chromosome), population size is set to be 100 and 40 parents are included in the mating pool. More detailed condition is presented in Table 2 shown below.

Table 2: Genetic Algorithm Used for Plant Reload Optimization

Procedure for genetic algorithm	Plant reload optimization
Fitness evaluation	invLinear function
Parent selection	Roulette wheel
Crossover	One-point crossover (80% crossover probability with randomly chosen crossover location)
Mutation	Swap mutation (90% mutation probability with randomly chosen mutation location)
Survivor selection	Fitness-based survivor selection
Termination criteria	1) Maximum number of iterations 2) P-averaged Hausdroff distance between two consecutive generations

Figure 6 shows the convergence of cycle length computed by the genetic algorithm. The results showed that the cycle length converged to an optimal value, around 600 days. Even though the material composition is not realistic enough, these results shown that the genetic algorithm can be used as a part of the entire plant reload framework.

Figure 10: Optimized Cycle Length Using Genetic Algorithm



4. CONCLUSION

As a part of RISA pathway in the LWRS program, plant reload optimization project has launched to develop a plant reload optimization framework. In the framework, RAVEN takes a role as a main controller of the entire framework as well as the optimization platform. This paper proposed the optimization platform using genetic algorithm and explained the optimization procedure step by step. In section 3, the genetic algorithm is demonstrated using a simple quarter core problem. The results showed that it converges reasonably to the expected optimum value. However, the optimization conditions such as material composition and constraints are not realistic enough to represent the physical phenomena. Therefore, the genetic algorithm will be improved using more realistic conditions in the future.

Acknowledgements

This work was supported by DOE's LWRS Program.

References

[1] Y-J. Choi et al., "Demonstration of the plant fuel reload process optimization for an operating PWR," INL/EXT-21-64549, Rev. 0, 2021.
[2] B. D. Andersen, "A Machine Learning Based Approach to Minimize Crud Induced Effects in Pressurized Water Reactors," Dissertation, North Carolina State University, 2021.
[3] A. Castillo et al., "Comparison of Heuristic Optimization Techniques for the Enrichment and Gadolinia Distribution in BWR Fuel Lattices and Decision Analysis," Annals of Nuclear Energy, Vol. 63, pp. 556-564, 2014.
[4] B. Andersen et al., "MOF: A Modular Framework for Rapid Application of Optimization Methodologies to General Engineering Design Problems," arXiv preprint arXiv:2204.00141, 2022.
[5] A. Alfonsi et al., "RAVEN Theory Manual," INL/EXT-16-38178, Rev. 4, Idaho National Laboratory, 2021.
[6] Y. Heo et al., "Demonstration of the Plant Fuel Reload Process Optimization for an Operating PWR," Hawaii, U.S., PSAM16, June 26th – July 1st, 2022.
[7] A. Alfonsi et al., "RAVEN Theory Manual," INL/EXT-16/38178, Rev. 4, 2021.
[8] Deb, Kalyanmoy, et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II." IEEE transactions on evolutionary computation 6.2 (2002): 182-197.
[9] Demo, Nicola, Marco Tezzele, and Gianluigi Rozza. "A supervised learning approach involving active subspaces for an efficient genetic algorithm in high-dimensional optimization problems." SIAM Journal on Scientific Computing 43.3 (2021): B831-B853.
[10] Wang, Guan, Chen Chen, and Kwok Yip Szeto. "Accelerated genetic algorithms with Markov Chains." Nature Inspired Cooperative Strategies for Optimization (NICSO 2010). Springer, Berlin, Heidelberg, 2010. 245-254.
[11] Kao, Yi-Tung, and Erwie Zahara. "A hybrid genetic algorithm and particle swarm optimization for multimodal functions." Applied soft computing 8.2 (2008): 849-857.
[12] Maleki, Isa, Ali Ghaffari, and Mohammad Masdari. "A new approach for software cost estimation with hybrid genetic algorithm and ant colony optimization." International Journal of Innovation and Applied Studies 5.1 (2014): 72.
[13] K. Smith et al., "SIMULATE-3 Methodology," Studsvik/SOA-95/18, 1995.
[14] J. Rhodes et al., "CASMO-4E: Extended Capability of CASMO-4, User's Manual," SSP-01/401 Rev. 2, Studsvik Scandpower, 2004.