Chapter 5

REALM Demonstration

This chapter demonstrates using REALM to apply genetic algorithms to maximize k_{eff} in a single AHTR fuel slab. The `dissertation-results` Github repository contains all the scripts, results, and plots shown in this chapter [**?**].

5.1 Straightened AHTR Fuel Slab

5.1.1 Problem Definition

This demonstration explores how inhomogeneous fuel distributions impact k_{eff} compared with homogenous fuel distributions customary in most reactor designs. The reactor core explored is a straightened slab from the FHR benchmark's AHTR design. Figure 5.1 illustrates the straightened fuel slab. The slab has $27.1 \times 3.25 \times 1.85$ cm^3 dimensions with periodic boundary conditions in the x- and y-directions and reflective boundary conditions in the z-direction. The materials are the same as in the FHR benchmark, except for the

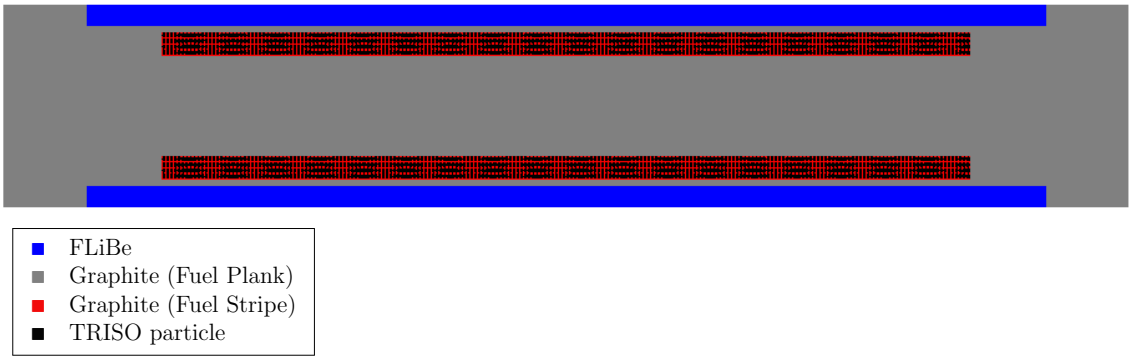


Figure 5.1: Straightened Advanced High Temperature Reactor (AHTR) fuel slab.

homogenization of each TRISO particle's four outer layers: porous carbon buffer, inner pyrolytic carbon, silicon carbide layer, and the outer pyrolitic carbon. The TRISO particles' dimensions remain the same. The k_{eff} for this original straightened AHTR configuration is 1.38625 ± 0.00109.

The REALM optimization problem's objective is to maximize the slab's k_{eff} by varying the TRISO particle packing fraction across the slab while keeping total TRISO particle the total packing fraction constant at 0.0979. This total packing fraction is consistent with the original straightened slab with TRISO particles in fuel stripes (Figure 5.1). The slab is divided into ten slices along the x-axis between the FLiBe and graphite buffers resulting in ten $2.31 \times 2.55 \times 1.85\ cm^3$ slices. A sine distribution governs the TRISO particle packing fraction's distribution across slices:

$$PF(x) = (a\ sin(bx + c) + 2) \times NF \tag{5.1}$$

where

PF = packing fraction [−]

a = amplitude, peak deviation of the function from zero [−]

b = angular frequency, rate of change of the function argument $[\frac{radians}{cm}]$

c = phase, the position in its cycle the oscillation is at t = 0 $[radians]$

x = midpoint value for each slice $[cm]$

NF = Normalization factor [−]

The sine distribution's value at each of the ten x-slices' midpoints is collected and normalized by the total packing fraction to ensure a consistent number of TRISO particles in the slab. Thus, for a packing fraction distribution of $PF(x) = (0.5\ sin(\frac{\pi}{3}x + \pi) + 2) \times NF$, The packing fraction for the ten slices are 0.103, 0.120, 0.049, 0.138, 0.076, 0.081, 0.136, 0.048,

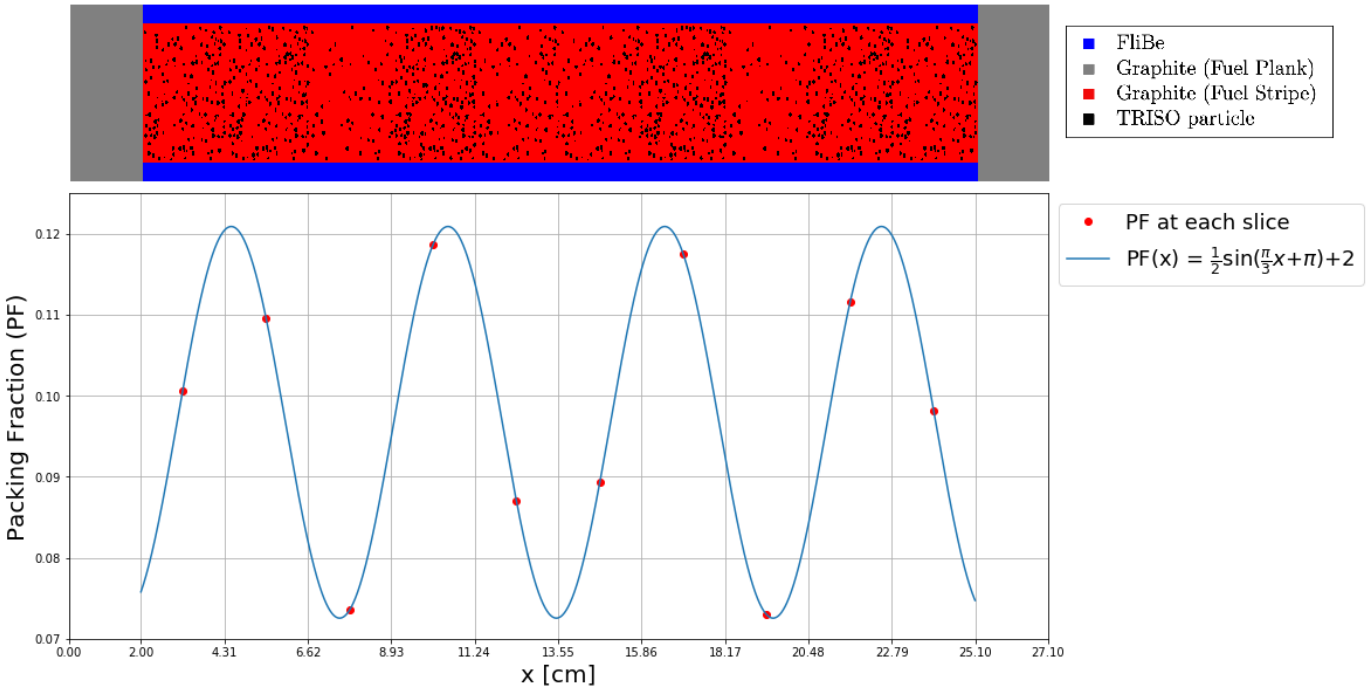


Figure 5.2: Below: $PF(x) = (0.5\ sin(\frac{\pi}{3}x + \pi) + 2) \times NF$ sine distribution with red points indicating the packing fraction at each slice. Above: Straightened Advanced High Temperature Reactor (AHTR) fuel slab with varying TRISO particle distribution across ten slices based on the sine distribution.

0.125, 0.098, respectively. Figure 5.2 shows this sine distribution, highlights the packing fraction at the respective midpoints, and the slab's XY-view with packing fraction varying based on this sine distribution.

In REALM, a genetic algorithm varies the a, b, and c variables to find a combination which produces a packing fraction distribution that maximizes the slab's k_{eff}. The upper and lower bounds of a, b and c ~~are~~:

- $0 < a < 2$
- $0 < b < \frac{\pi}{2}$
- $0 < c < 2\pi$

The a variable's bounds ~~were selected~~ to keep the sine distribution from falling below zero. b and c variable bounds ~~are~~ wide enough to allow the genetic algorithm to explore various

sine distributions. The OpenMC evaluator calculates k_{eff}. Each OpenMC simulation is run with 80 active cycles, 20 inactive cycles, and 8000 particles to reach ~130pcm uncertainty. Figure 5.3 shows the REALM input file for this genetic algorithm optimization problem. `ahtr_slab_openmc.py` is the template OpenMC straightened AHTR slab script that accepts a, b and c from REALM, calculates packing fraction distribution, and assigns packing fraction values to each fuel slice. Subsequently, REALM runs the templated OpenMC script to generate k_{eff}.

5.1.2 Hyperparameter Search

In REALM's input file, the user defines the genetic algorithm's hyperparameters. A good hyperparameter set guides the optimization process by balancing exploitation and exploration to find an optimal solution quickly and accurately. Finding a good set of hyperparameters requires a trial and error process.

I performed the hyperparameter search with a coarse-to-fine random sampling scheme, whose advantages were previously discussed in Section 2.4.2. The hyperparameters varied include population size, number of generations, mutation probability, mating probability, selection operator, selection operator's number of individuals, selection operator's tournament size, mutation operator, and mating operator. I started with 25 coarse experiments and fine-tuned the hyperparameters with 15 more experiments. For each genetic algorithm experiment, I hold the number of OpenMC evaluations constant at 600. The number of evaluations correlates the population size and number of generations. I randomly sample population size and use the following equation to calculate the number of generations:

$$\text{no. of generations} = \frac{\text{no. of evaluations}}{\text{population size}} \tag{5.2}$$

Table 5.1 shows the lower and upper bounds used for each hyperparameter's random sampling.

```
{
    "control_variables": {
        "a": {"min": 0.0, "max": 2.0},
        "b": {"min": 0.0, "max": 1.57},
        "c": {"min": 0.0, "max": 6.28},
    },
    "evaluators": {
        "openmc": {
            "input_script": "ahtr_slab_openmc.py",
            "inputs": ["a", "b", "c"],
            "outputs": ["keff"],
            "keep_files": false,
        }
    },
    "constraints": {"keff": {"operator": [">="], "constrained_val": [1.0]}},
    "algorithm": {
        "objective": "max",
        "optimized_variable": "keff",
        "pop_size": 60,
        "generations": 10,
        "mutation_probability": 0.23,
        "mating_probability": 0.46,
        "selection_operator": {"operator": "selTournament", "k": 15, "tournsize": 5},
        "mutation_operator": {
            "operator": "mutPolynomialBounded",
            "eta": 0.23,
            "indpb": 0.23,
        },
        "mating_operator": {"operator": "cxBlend", "alpha": 0.46},
    },
}
```

Figure 5.3: Reactor Evolutionary Algorithm Optimizer (REALM) JSON input file to maximize k_{eff} in the straightened Advanced High Temperature Reactor (AHTR) fuel slab by varying packing fraction distribution with control variables, a, b, c.

Table 5.1: Hyperparameter search is conducted in three phases: *Coarse Search*, *Fine Search 1*, *Fine Search 2*. Each hyperparameter's lower and upper bounds for each search phase is listed.

Hyperparameter	**Type**	**Coarse Search Bounds**	**Fine Search 1 Bounds**	**Fine Search 2 Bounds**
Experiments	-	0 to 24	24 to 34	35 to 39
Population size (pop)	Continuous	$10 < x < 100$	$20 < x < 60$	60
Mutation probability	Continuous	$0.1 < x < 0.4$	$0.2 < x < 0.4$	$0.2 < x < 0.3$
Mating probability	Continuous	$0.1 < x < 0.6$	$0.1 < x < 0.3$	$0.45 < x < 0.6$
Selection operator	Discrete	`SelTournament`, `SelBest`, `SelNSGA2`	`SelTournament`, `SelBest`, `SelNSGA2`	`SelTournament`
Selection individuals	Continuous	$\frac{1}{3}pop < x < \frac{2}{3}pop$	$\frac{1}{3}pop < x < \frac{2}{3}pop$	15
Selection tournament size (only for SelTournament)	Continuous	$2 < x < 8$	$2 < x < 8$	5
Mutation Operator	Discrete	`mutPolynomialBounded`	`mutPolynomialBounded`	`mutPolynomialBounded`
Mating Operator	Discrete	`cxOnePoint`, `cxUniform`, `cxBlend`	`cxOnePoint`, `cxUniform`, `cxBlend`	`cxOnePoint`, `cxBlend`

The initial 25 coarse experiments' objective is to narrow down the hyperparameters to find a smaller set of optimal hyperparameter bounds that produce higher final generation k_{eff} values. Figure 5.4 shows the hyperparameters' plotted against each other with a third color dimension representing the average k_{eff} value in each experiment's final generation. The lighter the scatter point is, the higher the final population's average k_{eff} value is, the better the hyperparameter set. The hyperparameters are plotted against each other to visualize the interdependence between hyperparameters. From the coarse hyperparameter search, the trends that stood out were:

- Mutation probability has a higher k_{effave} between 0.2 and 0.4.
- Mating probability has a higher k_{effave} between 0.1 and 0.3.
- Population size has a higher k_{effave} between 20 and 60.

There is also no interdependence between hyperparameters.

Next, I proceeded to the fine search. From Figure 5.4, I narrowed down population size, mutation probability, and mating probability bounds, as shown in Table 5.1's *Fine Search*

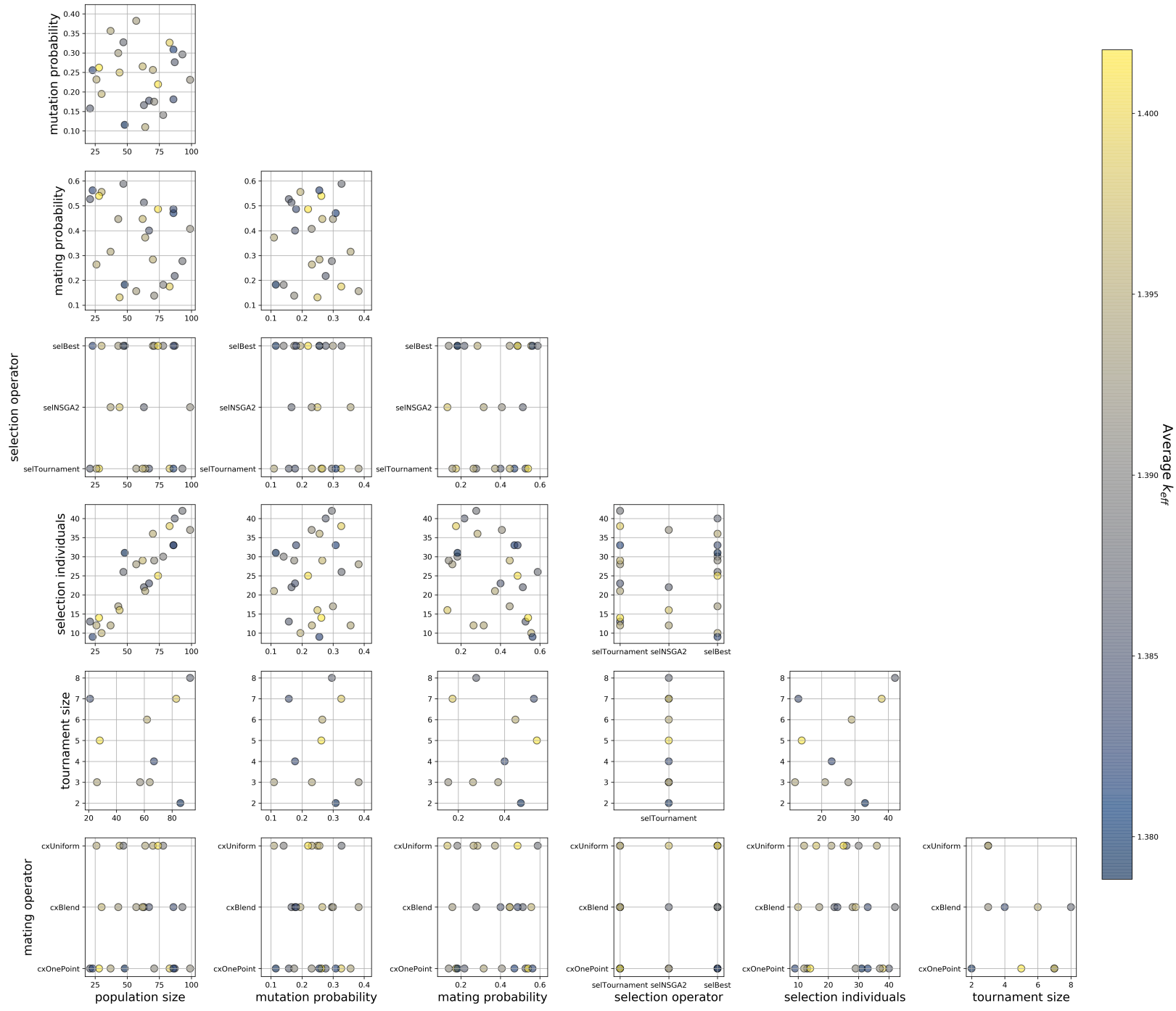


Figure 5.4: Coarse hyperparameters search's results. Hyperparameter values are plotted against each other with a third color dimension representing each experiment's final population's average k_{eff}.

1 Bounds column. There were no significant trends in the other hyperparameters, so I left them as is. I ran ten more experiments (25 to 34) with sampling hyperparameters from the *Fine Search 1 Bounds*. From these results, I conducted a second fine search with five experiments (35 to 39) with further tuned hyperparameter bounds as shown in Table 5.1's *Fine Search 2 Bounds* column. These new hyperparameter bounds were determined based on these reasons:

- Mutation probability has a higher k_{effave} between 0.2 and 0.3.
- I overlooked k_{effave} peaking at mating probability between 0.45 and 0.6 in the previous *Fine Search 1*, thus shifted the bounds.
- The highest k_{effave} occurred for `selTournament`.
- I narrowed down mating operator options to `cxBlend` and `cxOnePoint` since they had higher k_{effave}.
- I decided to select an arbitrary number for population size, selection individuals, and tournament size since they did not correlate with k_{effave} values.

Figure 5.5 shows the relationship between hyperparameter values and a,b,c control parameters, final generation k_{effmax}, and final generation k_{effave}. The coarse experiments' scatter points are 50% transparent, while the fine experiments' scatter points are opaque. In Figure 5.5, on average, the fine experiments (opaque scatter points) have higher k_{effave}, which indicates that the hyperparameter search process met its objective of finding hyperparameter bounds that enable quicker and more accurate optimization.

Table 5.2 shows the hyperparameters for the five experiments with the highest final generation k_{effave}. Figure 5.6 shows the packing fraction distributions that produced the k_{effmax} from the top five experiments. Four experiments had similar packing fraction distributions peaking at approximately 0.23 in the slab's, while one experiment had an exponential-looking distribution with peak packing fraction of 0.31 at the slab's side. The similar final packing

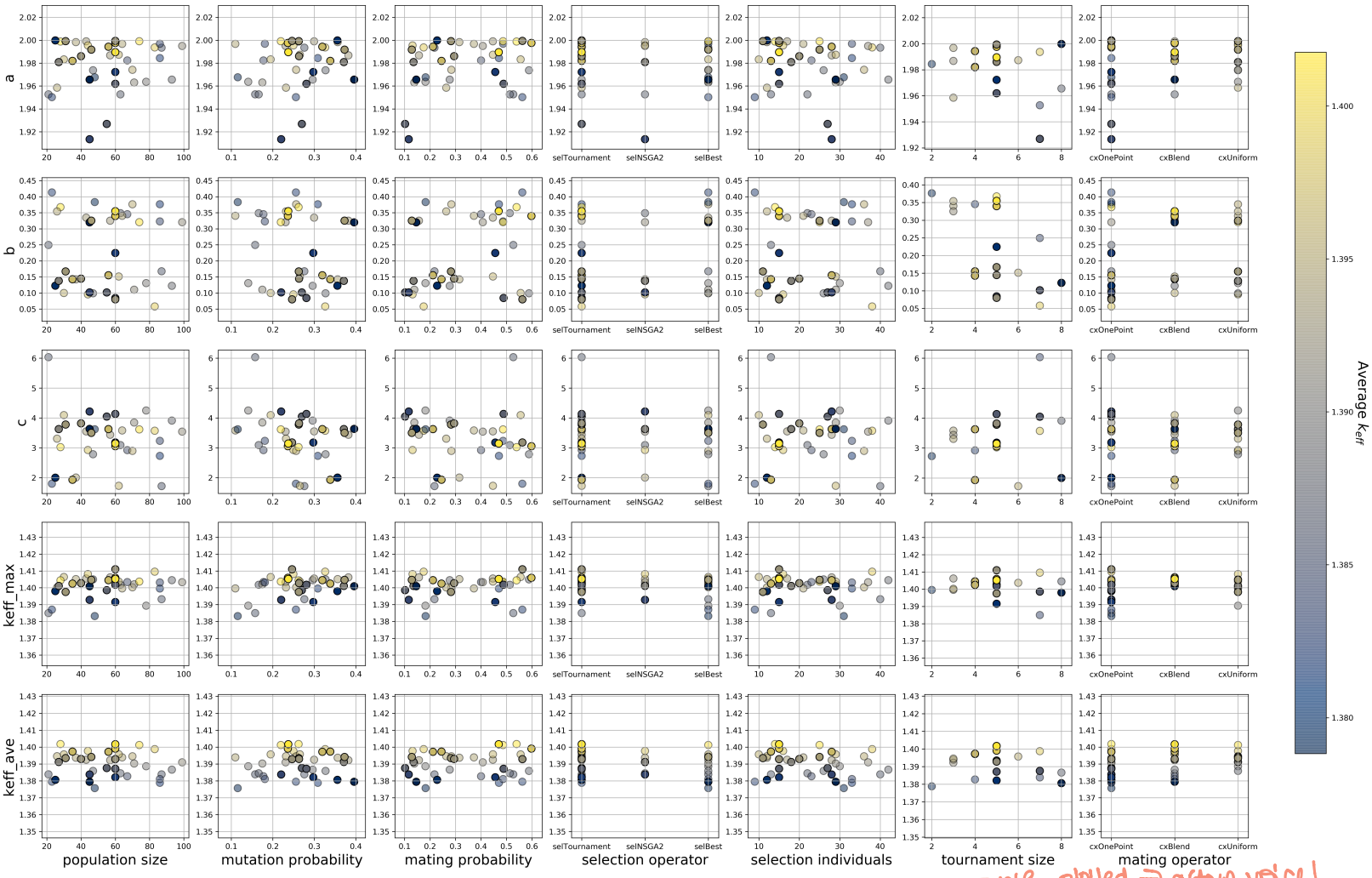


Figure 5.5: Hyperparameters search's results. Hyperparameters are plotted against a,b,c control parameters, each experiment's final generation k_{effmax}, and final generation k_{effave} with a third dimension representing each experiment's final population's average k_{eff}.

Table 5.2: Control Parameters, k_{eff} results, and hyperparameter values for the five hyperparameter search experiments with the highest final generation k_{effave}.

Control/Output Parameters	**Experiment 6**	**Experiment 15**	**Experiment 24**	**Experiment 36**	**Experiment 39**
k_{effave}	1.39876	1.40175	1.40118	1.39906	1.40165
k_{effmax}	1.40954	1.40440	1.40365	1.40590	1.40519
a	1.993	1.998	1.999	1.997	1.989
b	0.057	0.367	0.320	0.339	0.354
c	3.571	3.022	3.615	3.053	3.143
Hyperparameter					
Population size	83	28	74	60	60
Generations	8	22	9	10	10
Mutation probability	0.32	0.26	0.21	0.23	0.23
Mating probability	0.17	0.53	0.48	0.59	0.46
Selection operator	`selTournament`	`selTournament`	`selBest`	`selTournament`	`selTournament`
Selection individuals	38	14	25	15	15
Selection tournament size	7	5	-	5	5
Mutation Operator	`mutPolynomial Bounded`	`mutPolynomial Bounded`	`mutPolynomial Bounded`	`mutPolynomial Bounded`	`mutPolynomial Bounded`
Mating Operator	`cxOnePoint`	`cxOnePoint`	`cxUniform`	`cxBlend`	`cxBlend`

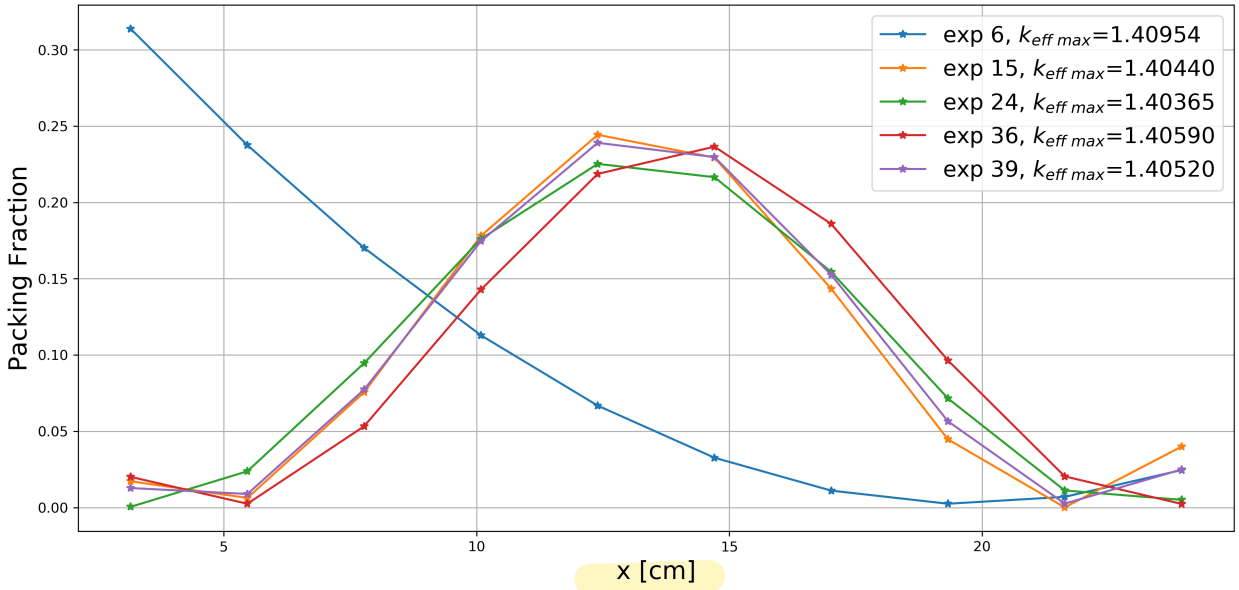


Figure 5.6

fraction distributions demonstrate genetic algorithms' robustness to find the optimal global solutions with different hyperparameters.

These simulations are run on the BlueWaters supercomputer [48]. In each REALM simulation, a population size number of individual OpenMC simulations are run at each generation. Each OpenMC simulation takes approximately 13 minutes to run on a single BlueWaters XE node. With approximately 600 OpenMC evaluations per REALM simulation, the total REALM simulation time is about 130 BlueWaters node-hours. The hyperparameter search ran 40 REALM simulations, thus using approximately 5200 node-hours.

5.1.3 Results for best hyperparameter set

I define the best-performing hyperparameter set as the experiment that produces the highest k_{effave} in its final generation. The best performing hyperparameter set is from the final *Fine Search 2*. It is experiment 39 in Table 5.2 with center-peaking packing fraction distribution with k_{effmax} = 1.40519. Experiment 39's k_{effmax} is ~ 2000pcm higher than the original straightened AHTR configuration's k_{eff}. Figures 5.7 show the packing fraction distribution that produced k_{effmax} = 1.40519.

Figure 5.8a and 5.8b show the k_{eff} evolution and packing fraction distribution through the best performing 39^{th} experiment's generations. The k_{effmax} converged quickly by generation 1; however, this is not usually the case. The genetic algorithm optimization process is stochastic, resulting in the possibility that the algorithm randomly samples a control parameter set that maximizes the objective function early in the genetic algorithm optimization process. The k_{effave} demonstrates how each generation's average k_{eff} converges towards a higher value with each generation's improvements. To demonstrate how the genetic algorithm optimization process usually goes, Figures 5.9a and 5.9b show the k_{eff} evolution and packing fraction distribution through the second-best performing 15^{th} experiment's generations. Experiment 15 demonstrates how both maximum and average k_{eff} converges towards a higher k_{eff} with improvements from each generation.

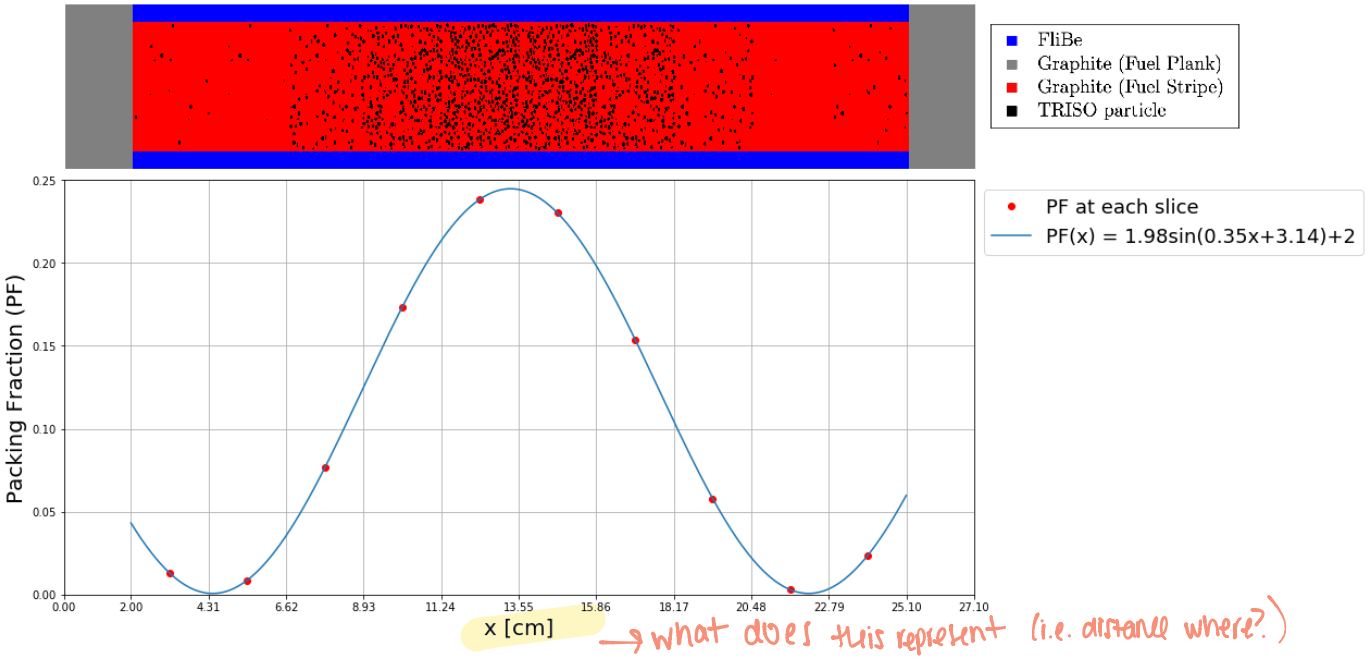
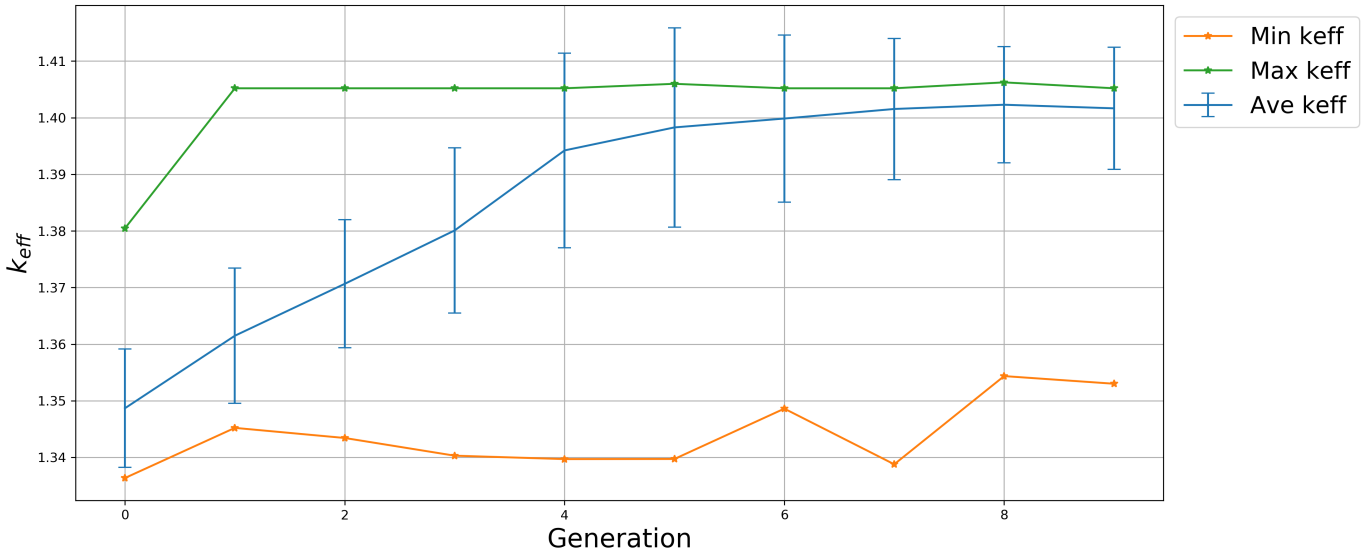
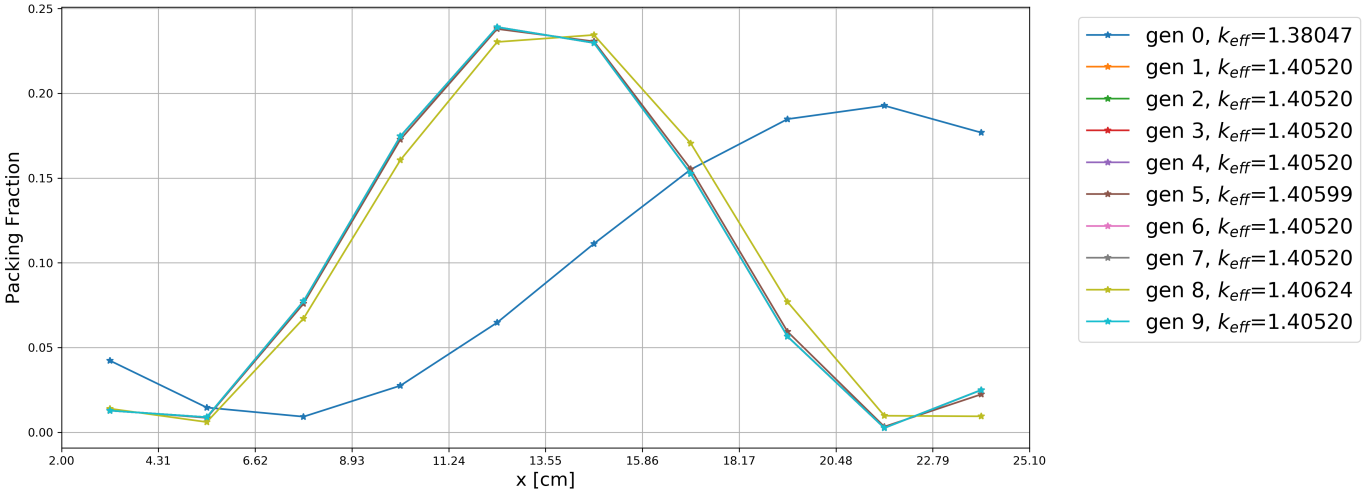


Figure 5.7: Experiment 39 packing distribution that produced $k_{effmax} = 1.40519$. Below: $PF(x) = (1.98\ sin(0.35x + 3.14) + 2) \times NF$ sine distribution with red points indicating the packing fraction at each slice. Above: Straightened Advanced High Temperature Reactor (AHTR) fuel slab with varying TRISO particle distribution across ten slices based on the sine distribution.

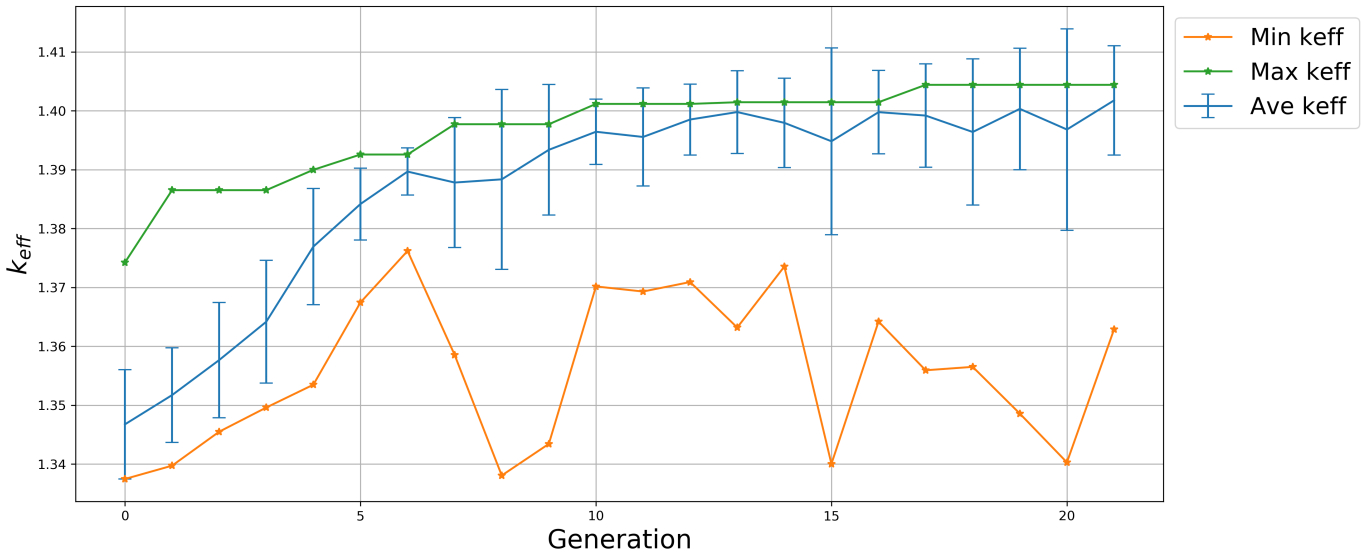


(a) Minimum, average, and maximum k_{eff} values evolution.

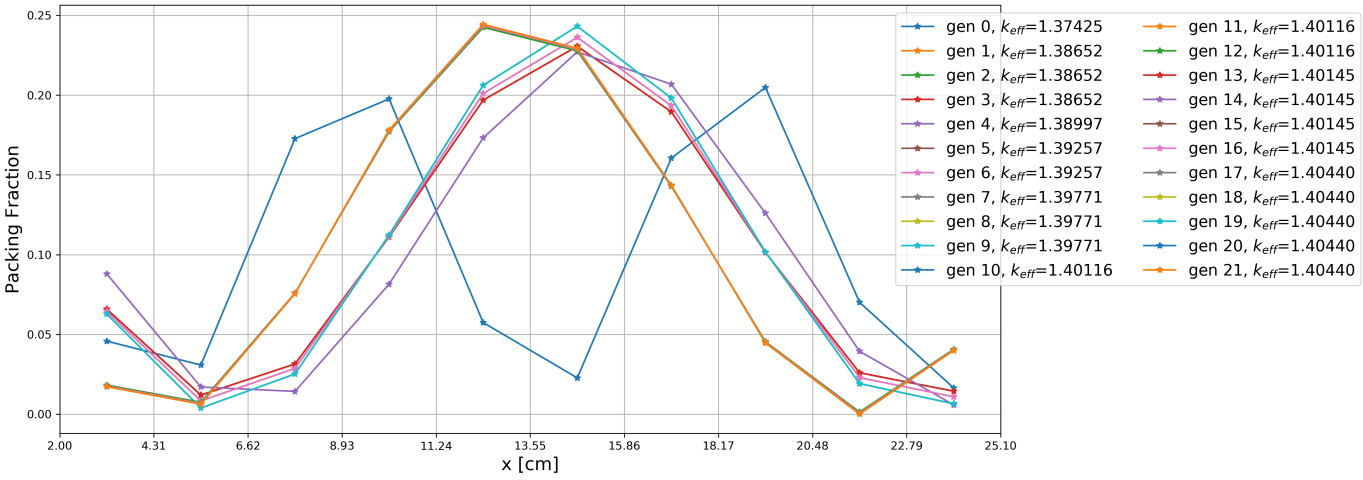


(b) Maximum k_{eff}'s packing fraction distribution evolution.

Figure 5.8: Results for each generation for REALM's genetic algorithm optimization of the Straightened Advanced High Temperature Reactor (AHTR) Fuel Slab. The REALM simulation used the 39^{th} experiment's hyperparameter set.



(a) Minimum, average, and maximum k_{eff} values evolution.



(b) Maximum k_{eff}'s packing fraction distribution evolution.

Figure 5.9: Results for each generation for REALM's genetic algorithm optimization of the Straightened Advanced High Temperature Reactor (AHTR) Fuel Slab. The REALM simulation used the 15^{th} experiment's hyperparameter set.

Both experiments 39 and 15 have packing fractions peaking at approximately 0.23 in the slab's center and decreasing to zero at the slab's sides. The amplitude, a, for the packing fraction distribution that produced k_{effmax} for experiment 39 and the other top-five experiments (Table 5.2) have settled at the upper bound of approximately 2. A higher amplitude, a, shows that a slab geometry with bigger packing fraction variations results in a higher k_{eff}. These observations about packing fraction distribution for k_{effmax} are consistent with what I understand from the original AHTR: a high k_{eff} occurs with a good balance between fuel loading and moderation space. Fission occurs at high TRISO particle concentration areas at thermal flux; however, the neutrons are born at fast-flux and require moderation to slow down to thermal ranges. Therefore, larger moderation areas ensure higher resonance escape probability for the fast neutrons resulting in higher thermal flux, leading to more fission occurring and a higher k_{eff}.

Another observation is that TRISO particle packing fraction peaks in the center of the slab, proving that if the optimization problem focuses purely on the slab's neutronics by maximizing k_{eff}, the fuel tends to want to culminate in the middle. However, this is not ideal for other important reactor core qualities such as good heat transfer and ensuring flat power across the core. With these shortcomings in mind, I proceed to the future work chapter to discuss the simulations that will be run to optimize these other parameters.

5.2 Summary

This chapter demonstrated successfully applying REALM to maximize k_{eff} in a straightened Advanced High Temperature Reactor (AHTR) fuel slab through varying the TRISO particle packing fraction distribution. I began by conducting a coarse-to-fine random sampling hyperparameter search to find the genetic algorithm hyperparameters that worked best for this optimization problem. Experiment 39 performed the best with a hyperparameter set that produced the highest final generation average k_{eff} of 1.40165. The TRISO particle packing

fraction distribution that produced the final generation's maximum k_{eff} of 1.40519 peaks at the slab's center with packing fraction distribution: $PF(x) = 1.989\ sin(0.54x + 3.143)$. This problem demonstrated how effective and robust genetic algorithms are for optimizing reactor parameters for an objective function. This demonstration problem had a single objective function which was to maximize k_{eff}. However, many other objectives should be considered, such as maximizing heat transfer and minimizing power peaking in the core. Thus, in the next chapter, I propose future simulations for optimizing these objective functions simultaneously.