INL/RPT-23-71667

Light Water Reactor Sustainability Program

Development of Genetic Algorithm Based Multi-Objective Plant Reload Optimization Platform

March 2023

U.S. Department of Energy

Office of Nuclear Energy

INL/RPT-23-71667

Development of Genetic Algorithm Based Multi-Objective Plant Reload Optimization Platform

Junyung Kim
Mohammad Abdo
Congjian Wang
Yong-Joon Choi

Idaho National Laboratory

March 2023

Prepared for the
U.S. Department of Energy
Office of Nuclear Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

Page intentionally left blank

EXECUTIVE SUMMARY

The U.S. nuclear industry is facing a challenge in maintaining required levels of safety while ensuring economic competitiveness to stay in business. Safety remains a key parameter for all aspects of light-water reactor nuclear power plant operations. Safety can become more economical by using a risk-informed ecosystem, such as the one being developed in the Risk-Informed Systems Analysis Pathway under the U.S. Department of Energy Light Water Reactor Sustainability Program. The Light Water Reactor Sustainability Program promotes a wide range of research and development activities to maximize both the safety and economic efficiency of nuclear power plants through improved scientific understanding, especially given that many plants are now considering second license renewals.

The Risk-Informed Systems Analysis Pathway has two main goals:

- Deploy methodologies and technologies that better represent safety margins and cost and safety factors
- Develop advanced applications that enable cost-effective plant operations.

The Plant Reload Optimization Platform development project aims to build a reactor core design tool that includes reactor safety and fuel performance analyses and uses artificial intelligence to support the optimization of core design solutions.

This report summarizes genetic-algorithm-based multi-objective fuel reload optimization activities, specifically:

- Developing the non-dominated sorting genetic algorithm II optimizer in the Risk Analysis and Virtual ENviroment (RAVEN)
- Demonstrating and validating the developed non-dominated sorting genetic algorithm II optimizer using benchmark optimization problems.

Page intentionally left blank

CONTENTS

FIGURES

TABLES

ACRONYMS

DOE	U.S. Department of Energy
EA	evolutionary algorithm
FY	fiscal year
GA	genetic algorithm
INL	Idaho National Laboratory
LWR	light-water reactor
LWRS	Light Water Reactor Sustainability
MOEA	multi-objective evolutionary algorithms
MOOP	multi-objective optimization problem
NSGA-II	non-dominated sorting genetic algorithm
PWR	pressurized-water reactor
RAVEN	Risk Analysis and Virtual ENvironment
RISA	Risk-Informed Systems Analysis
U.S.	United States

1. INTRODUCTION

The United States (U.S.) Department of Energy (DOE) Light Water Reactor Sustainability (LWRS) Program Risk-Informed Systems Analysis (RISA) Pathway Plant Reload Optimization Project aims to develop an integrated, comprehensive platform offering an all-in-one solution for reload evaluations with a special focus on core design optimization [1]. Fuel reload optimization is a multi-physics problem that needs to consider core design, fuel performance, and system safety. The genetic algorithm (GA) was selected as the foundation for the optimization platform. In Fiscal Year (FY) 2023, researchers used NSGA-II (non-dominated sorting genetic algorithm II), a GA variant, for the multi-objective optimization problems (MOOPs) to optimize multiple objectives, such as fuel cycle length, enrichment, and burnable poisons with multiple constraints, including core design limits and system safety parameters.

This project used Idaho National Laboratory's Risk Analysis and Virtual Environment (RAVEN) [2] as the workflow manager and a fuel reload optimization platform. RAVEN controls the perturbations of input decks to all the physics codes in neutronics, fuel performance, and safety analyses via generic and specialized built-in code interfaces, parses inputs and outputs, and performs post-processing of the simulation results.

The reactor core design problem is a complex problem involving high-dimensional input-output streams with nuclear power plant level feedback, rendering the interdependencies between nonlinear parameters of interest. It is challenging to create a global model that can translate multi-physics inputs into safety requirements or economic metrics. The reactor core design usually involves considering constraints of numerous parameters and multiple objectives, including neutronics, fuel performance, safety requirements, and economic objectives.

Many studies have been dedicated to optimizing the fuel loading pattern and leading the development of artificial-intelligence-based approaches [3]. One of the metaheuristic techniques used for optimizing the fuel loading pattern was the GA, which was inspired by natural evolution [4]. This method has proven to be more effective than traditional optimization techniques. These studies developed several in-core fuel management strategies using GA to optimize multiple parameters simultaneously [5]–[7].

The GA optimization platform development progressed in FY-2022 by implementing more evolutionary operations mechanisms (i.e., fitness functions, parent selection, crossover, mutation, and survivor selection) [8]. In FY-2023 the development focused on the improving the GA method for multi-objective optimization and risk-informed methodology applicability to the Plant Reload Optimization Platform.

2. DEVELOPMENT STATUS OF MULTI-OBJECTIVE OPTIMIZATION PLATFORM

2.1 Background of Multi-Objective Optimization

When a problem involves multiple objectives, it results in a set of optimal solutions known as Pareto-optimal solutions instead of a single optimal solution. In the absence of additional information, the solutions on the Pareto curve (or Pareto front) are assumed to be the optimal solutions, thus Pareto-optimal solutions. Traditional optimization methods, including multi-criteria decision-making techniques, recommend transforming the multi-objective optimization problem (MOOP) into a single-objective optimization problem by emphasizing one Pareto-optimal solution during single simulation. However, for a problem with multiple solutions, this approach needs to be applied multiple times, with each simulation expected to yield a different solution.

A MOOP includes a set of *n* decision variables, *k* objective functions, and a set of (*m* inequality and *p* equality) constraints. The optimization goal is:

$$\text{Min/Max } \boldsymbol{y}(\boldsymbol{x}) = \big(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \dots, f_k(\boldsymbol{x})\big), k \geq 2 \tag{1}$$

$$\text{Subject to } g_i(\boldsymbol{x}) \leq 0, i = 1, 2, \dots, m \tag{2}$$

$$h_j(\boldsymbol{x}) = 0, i = 1, 2, \dots, p \tag{3}$$

where $\boldsymbol{x} = (x_1, \dots, x_n)$ is an *n*-dimensional decision vector in $\boldsymbol{x} \in \mathbb{R}^n$ ($\mathbb{R}$ is the set of real numbers), $\boldsymbol{y}$ is a *k*-dimensional objective vector in $\mathbb{R}^k$, *f* defines the mapping function, g_i is the i^{th} inequality constraint, and h_j is the j^{th} equality constraint.

If the following conditions are satisfied, $\boldsymbol{x}_1$ can be considered as superior to $\boldsymbol{x}_2$, where $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are the two feasible solution vectors of the multi minimization problem.

$$f_j(\boldsymbol{x_1}) \leq f_j(\boldsymbol{x}_2) \text{ for all } j = \{1,2, \dots, k\}, \text{ and } f_j(\boldsymbol{x}_1) < f_j(\boldsymbol{x}_2) \text{ for at least one } j = \{1,2, \dots, k\}, \tag{4}$$

where *k* is the number of objective functions and $f_j(\boldsymbol{x})$ is j^{th} value of an objective function for decision vector $\boldsymbol{x}$.

Here, the vector value $\boldsymbol{x}$ is the Pareto-optimal solution when it is not dominated by any other feasible solutions. The collection of all Pareto-optimal solutions is a Pareto set, and the objective vectors that correspond to the Pareto set are called a Pareto front, as illustrated in Figure 1.

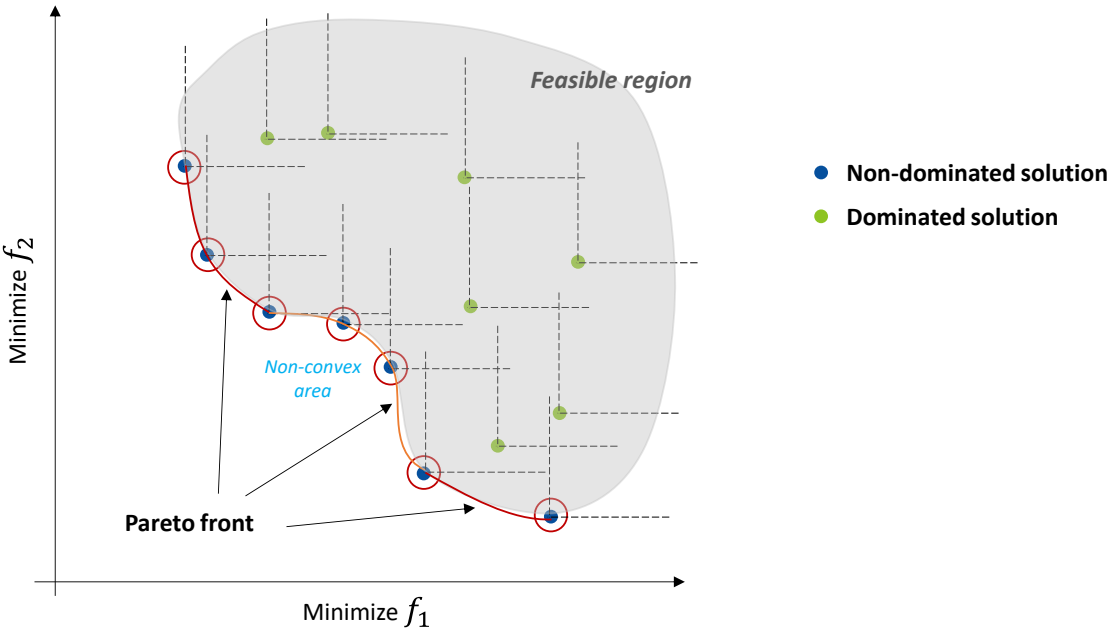

Figure 1. Pareto dominance.

Several multi-objective evolutionary algorithms (MOEAs) have been proposed with different purposes and applicability. Table 1 shows a summary of the different MOEAs. For the plant fuel reload optimization, the NSGA-II was selected for various reasons. Firstly, after testing it on multiple testing problems, NSGA-II showed an advantage in finding a wide range of solutions and converging characteristics compared to the other contemporary MOEAs [9]. NSGA-II, initially proposed by Deb et al. in 2000 [9], is a powerful GA-based method for solving MOOPs and problems with continuous and discrete variables. Furthermore, NSGA-II has shown its efficiency in managing many engineering optimization problems [10].

Table 1. Comparison summary among multi evolutionary algorithms [11].

Algorithm	Fitness Assignment	Diversity Mechanism	Elitism	External Population	Advantages	Disadvantages
VEGA [12]	Each subpopulation is evaluated with respect to a different objective	No	No	No	First MOGA straightforward implementation	Tend to converge to the extreme of each objective
MOGA [13]	Pareto ranking	Fitness sharing by niching	No	No	Simple extension of single-objective GA	Usually slow convergence
WBGA [14]	Weighted average of normalized objectives	Niching predefined wights	No	No	Simple extension of single-objective GA	Difficulties in non-convex objective function space
NPGA [15]	No fitness assignment Tournament selection	Niche count as tiebreaker in tournament selection	No	No	Very simple tournament selection	Problems related to niche size parameters
RWGA [16]	Weighted average of normalized objectives	Randomly assigned weights	Yes	Yes	Efficient and easy implement	Difficulties in non-convex objective function space
PESA [17]	No fitness assignment	Cell-based density	Pure elitist	Yes	Easy to implement and computationally efficient	Performance depends on cell sizes
PAES [18]	Pareto dominance is used to replace a parent if offspring dominates	Cell-based density as tiebreaker between offspring and parent	Yes	Yes	Random mutation hill-climbing strategy that is easy to implement and computationally efficient	Prior information needed about objective space, not a population-based approach, and performance depends on cell sizes
NSGA [19]	Ranking based on non-domination sorting	Fitness sharing by niching	No	No	Fast convergence	Problems related to niche size parameter
NSGA-II [20]	Ranking based on non-domination sorting	Crowding distance	Yes	No	Single parameter, well tested, and efficient	Crowding distance works in objective space only

SPEA [21]	Ranking based on the external archive of non-dominated solutions	Clustering to truncate external population	Yes	Yes	Well tested, with no parameter for clustering	Complex clustering algorithm
SPEA-2 [22]	Strength of dominators	Density based on the k^{th} nearest neighbor	Yes	Yes	Improved SPEA and made sure extreme points are preserved	Computationally expensive fitness and density calculation
RDGA [23]	The problem reduced to bi-objective problem with solution rank and density as objectives	Forbidden region cell-based density	Yes	Yes	Dynamic cell update that was robust with respect to the number of objectives	More difficult to implement than others
DMOEA [24]	Cell-based ranking	Adaptive cell-based density	Yes (implicitly)	Yes	Includes efficient techniques to update cell densities and adaptive approaches to set GA parameters	More difficult to implement than others

2.2 Non-Dominated Sorting Genetic Algorithm II

The NSGA-II optimization inherits definitions used in the GA method. For instance, the initial set of solutions—called a population—is composed of a chromosome, which is a vector of variables (called genes in NSGA-II). Figure 2 shows a schematic diagram of the population and its element.

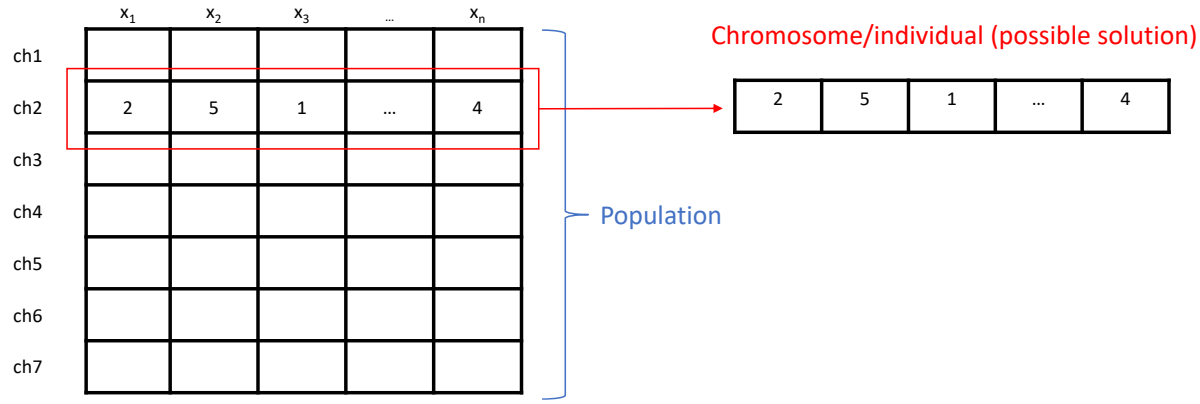

Figure 2. The set of potential solutions (population) and their elements.

2.2.1 Basic Structure of Non-dominated Sorting Genetic Algorithm II

NSGA-II has three basic concepts of optimization: dominance depth method, elitism, and crowding distance.

2.2.1.1 Dominance Depth Method

The dominance depth method sorts non-dominated solutions using the Pareto dominance concept. The non-dominated sorting procedure commences by allocating the initial population's non-dominated members to the first front (or so-called “rank” in NSGA-II). These members are then categorized into the first front and are removed from the initial population. The remaining population members undergo the dominance depth method. The non-dominated members of the residual population are then designated the second rank and added to the second front. This process is reiterated until all population members are grouped into different fronts based on their respective ranks. Figure 3 shows an example of the dominance depth method. The solutions are scattered and non-dominated in the left figure and sorted with four different Pareto fronts in the right figure.

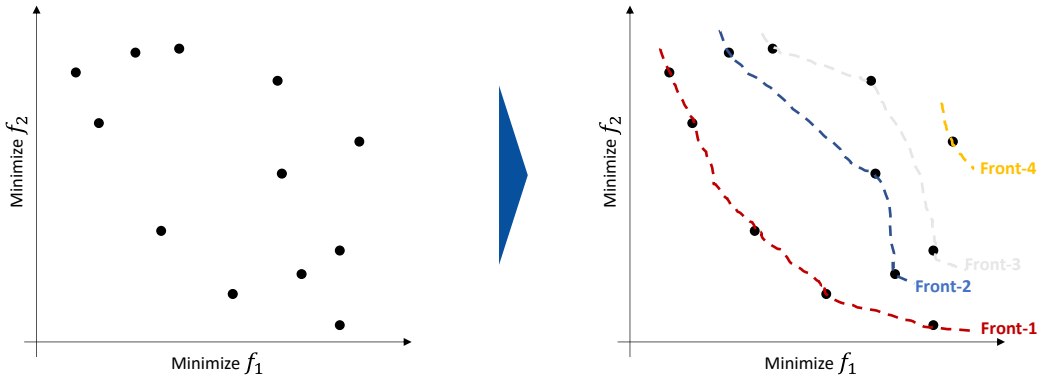

Figure 3. The dominance depth method.

2.2.1.2 Elitism

Elitism, also known as the elite preserving strategy, is an essential concept that NSGA-II emphasizes. It conserves a population's elite solutions by directly transferring them to the succeeding generation. Put differently, the non-dominated solutions discovered in each generation proceed to the next generations until some solutions dominate them.

2.2.1.3 Crowding Distance

To assess the density of solutions surrounding a specific solution, the crowding distance is computed. It represents the average distance between two solutions on each side of the solution along each objective. When comparing two solutions that have different crowding distances, the one with the greater crowding distance is believed to exist in a less congested area. The i^{th} solution's crowding distance is the average side length of the cuboid, as depicted in Figure 4. If f_j^i is the j^{th} value of an objective function for the i^{th} solution and f_j^{max} and f_j^{min} are the maximum and minimum values, respectively, of j^{th} objective function among all the solutions, the crowding distance of i^{th} solution is defined as the average distance of the two nearest solutions on either side, as given in Equation (4).

$$cd(i) = \sum_{i=1}^{k} \frac{f_j^{i+1} - f_j^{i-1}}{f_j^{max} - f_j^{min}} \tag{5}$$

where k is the number of objective functions.

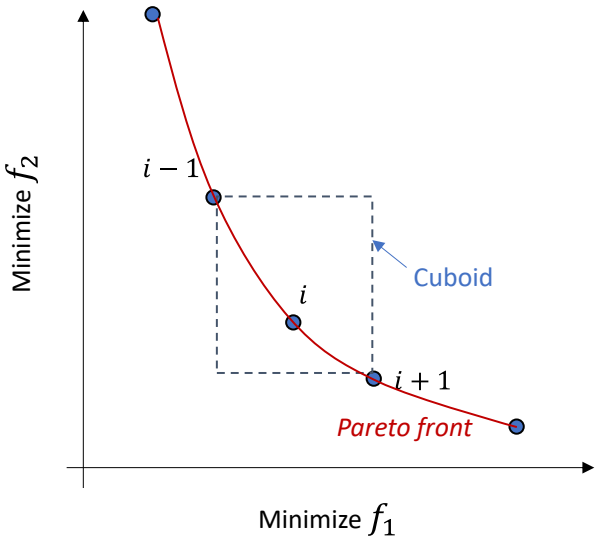

Figure 4. Cuboid with neighboring solutions for calculating crowding distance.

2.2.2 Survivor Selection of Non-Dominated Sorting Genetic Algorithm NSGA-II

The population for the next generation was selected using a tournament selection operator, which uses the rank of chromosomes and their crowding distances for selecting ones out of chromosomes for the next generation. The survivor selection process are:

[1] Select chromosomes that do not violate any constraints

[2] If both the chromosomes have different ranks, the one with the better rank is selected for the next generation

[3] If both the chromosomes are of the same ranks, the one with the higher crowding distance is selected for the next generation.

2.2.3 Procedures

The NSGA-II procedure begins with generating an initial population P(t=0) of size N, where t represents the number of iterations. Then a new population Q(t=0) (offspring) is created after performing crossover and mutation operations on the population P(t=0). After that, the population P(t=0) and Q(t=0) are combined to form a new population R(t=0) (which is the size of 2 × N), and the non-dominated sorting procedure is performed on R(t=0). Then the population members of R(t=0) are ranked into different fronts according to their non-domination levels.

The next process is to select N members from R(t=0) to create the next population P(t=1). If the size of the first front is greater than or equal to N, only N members are selected from the least crowded region of the first front to form P(t=1). On the contrary, if the size of the first front is less than N, the chromosomes of first front are directly transferred to the next generation, and the remaining members are taken from the least crowded region of the second front and added to P(t=1). If the size of P(t=1) is still less than N, the same procedure is followed for the next consecutive fronts until the size of P(t=1) becomes equal to N. The populations of P(t=2), P(t=3), …, are constructed following same procedure until the stopping criteria are satisfied. The NSGA-II procedure is shown in Figure 5.

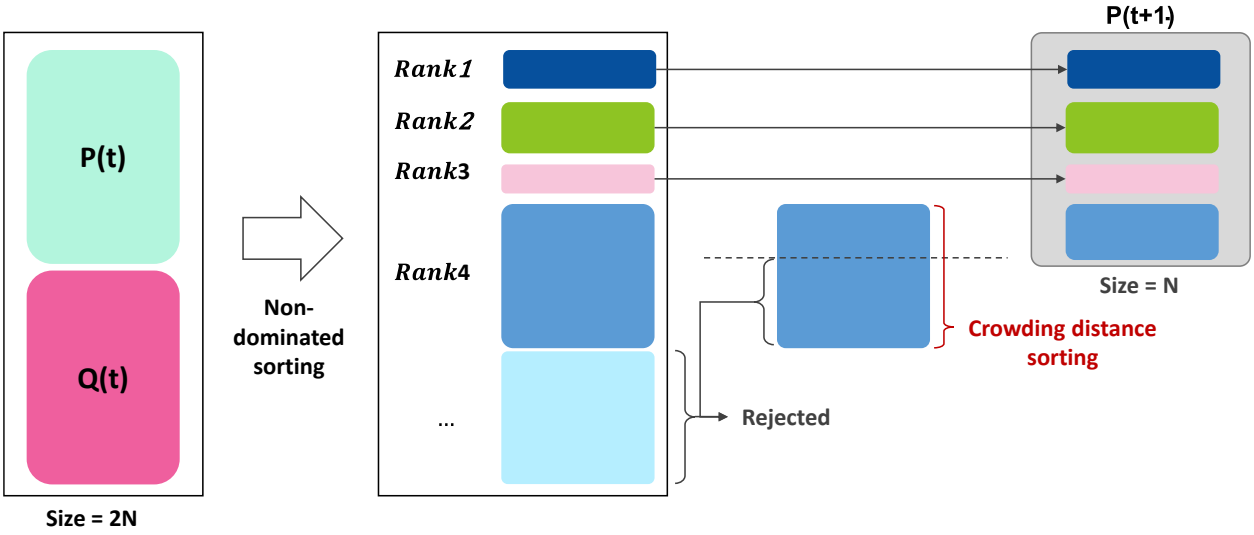

Figure 5. Procedure of NSGA-II.

2.3 Implementation of the Multi-Objective Optimization in RAVEN

The multi-objective optimization of NSGA-II was developed in the RAVEN framework. Constraint handling, mutation, and crossover processes were implemented in the RAVEN optimizer. The dominance depth and crowding distance methods were added in the survivor selection model. NSGA-II's multi-objective optimization RAVEN source code is not currently publicly available. The two demonstration cases were a minimization of local sums problem and a ZDT (Zitzler-Deb-Thiele) benchmark problem.

2.3.1 Minimizations of Local Sums with Constraints

The demonstration involved a simple MOOP, the minimizations of local sums. The general form of the problem can be expressed as:

- Minimize $f_1(\boldsymbol{x}) = \sum_{i=1}^{6} i \times x_i$
- Minimize $f_2(\boldsymbol{x}) = \sum_{i=1}^{2} i \times x_i$
- Explicit constraint: $x_3 + x_4 < 10$

- Implicit constraint: $f_1(\boldsymbol{x}) < 100$.

where $x_i \sim \mathcal{U}^d[2,7]$ is sampled from a discrete uniform distribution. Figure 6 shows examples of accepted and eliminated solutions. For instance, solution $\boldsymbol{x} = (7,4,6,3,2,5)$ is an accepted solution that satisfies all constraints. However, solution $\boldsymbol{x} = (4,3,6,7,2,5)$ violates an explicit constraint, so this solution will be eliminated during the survivor selection process. This problem is the multi-sum optimization problem.

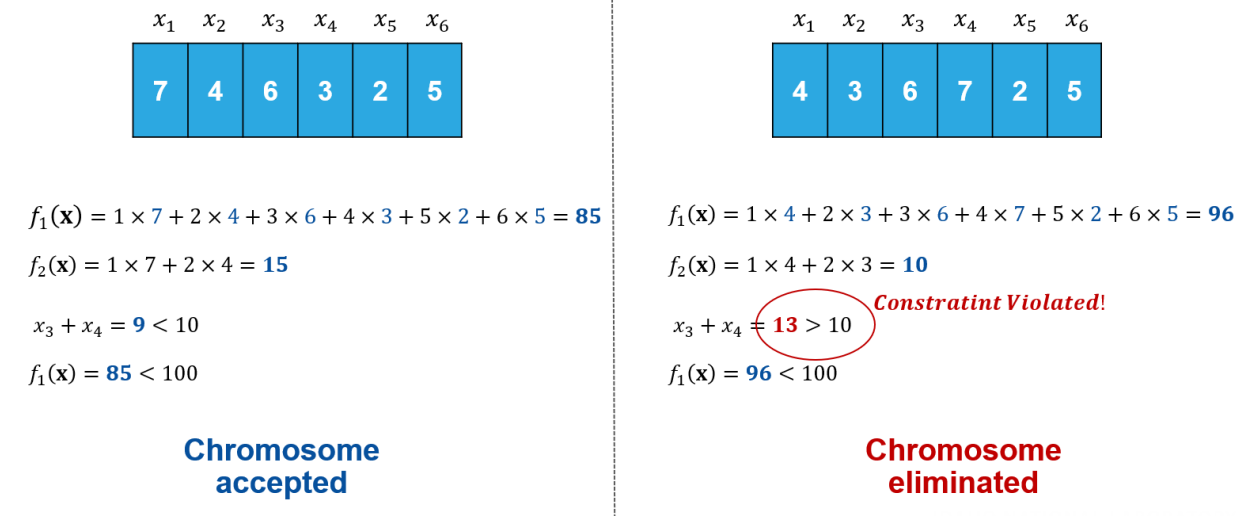

Figure 6. Multi-Sum problems with explicit and implicit constraints.

Figure 7 and Figure 8 show the multi-sum model and implicit and explicit constraint models for RAVEN, respectively. RAVEN input for the multi-sum optimization problem is shown in Appendix A.

```
def evaluate(Inputs):
  Sum = 0
  LocalSum1 = 0
  for ind,var in enumerate(Inputs.keys()):
    Sum += (ind + 1) * Inputs[var]
    if (ind == 1):
        LocalSum1 = Sum
  return Sum[:], LocalSum1[:]

def run(self,Inputs):
  """
    RAVEN API
    @ In, self, object, RAVEN container
    @ In, Inputs, dict, additional inputs
    @ Out, None
  """
  self.obj1,self.obj2 = evaluate(Inputs)
```

Figure 7. RAVEN multis-sum model.

```python
import numpy as np

def constrain(Input): ···

def implicitConstraint(Input): ···

def expConstr1(Input): ···

def expConstr2(Input): ···

def expConstr3(Input):
  g = 10 - Input.x3 - Input.x4
  return g

def impConstr1(Input): ···

def impConstr2(Input): ···

def impConstr3(Input):
  g = 100 - Input.obj1
  return g
```

Figure 8. Implicit and explicit constraint modeling of RAVEN.

Figure 9 shows the multi-sum test results. The solutions converge to the theoretic optimal solutions after 50 iterations. Table 2 shows parameters used for implementing NSGA-II in RAVEN for multi-sum problem solving. A two points crossover and random mutator gave enough perturbation to the offspring populations. The probabilities of crossover and mutation were set to 0.7 from the maximum value of 1.0. This led to an increased level of diversity in the offspring populations.

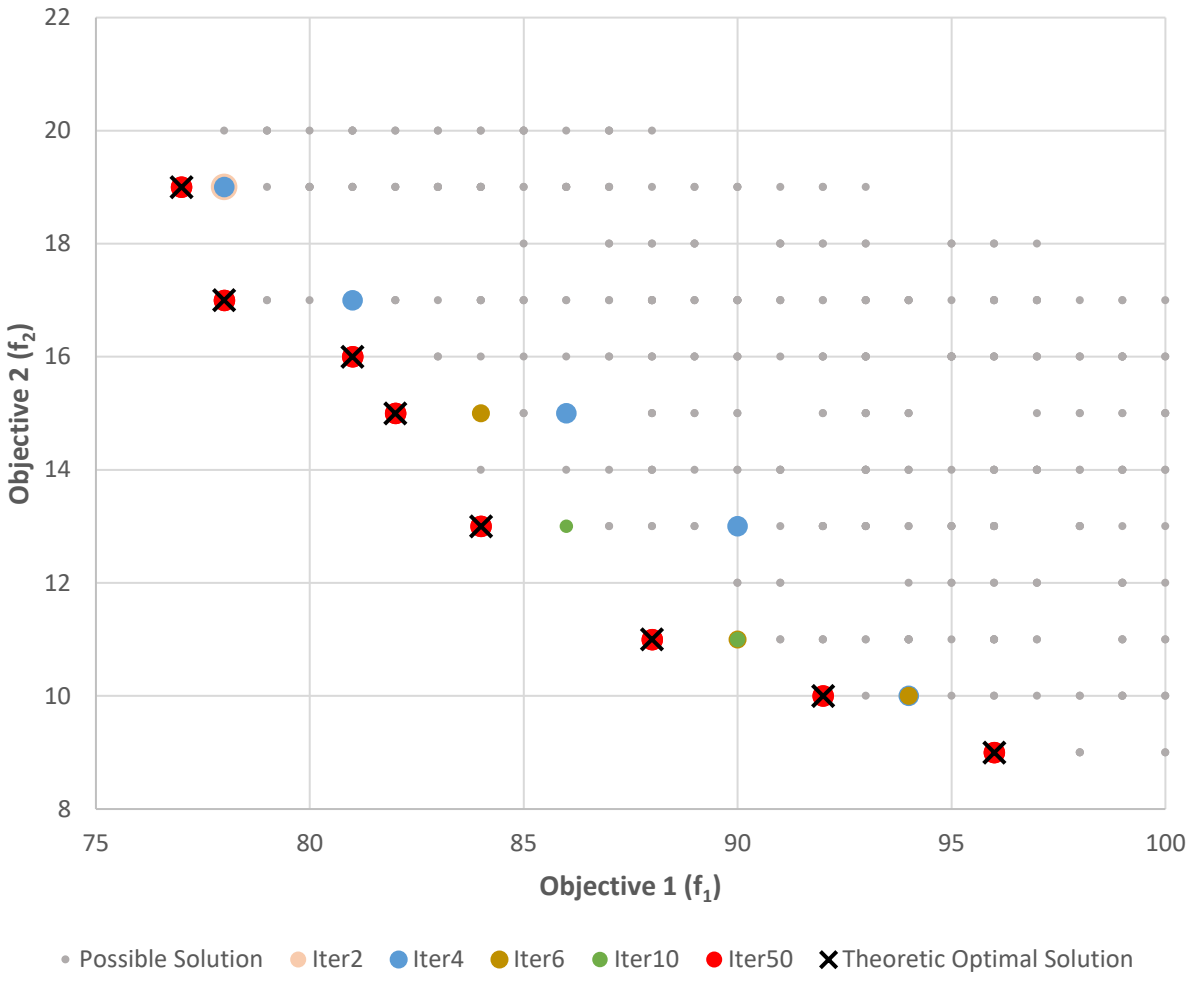

Figure 9. Multi-sum test results in RAVEN.

Table 2. NSGA-II parameters for the multi-sum problem in RAVEN.

Population size	30	Generation	50
Crossover type	Two points crossover	Mutation type	Random mutator
Probability of crossover	0.7	Probability of mutation	0.7

2.3.2 Benchmark with the Zitzler-Deb-Thiele Problem

The ZDT problem is a MOOP that is commonly used as a benchmark in evolutionary algorithms and other optimization techniques [25]. The ZDT problem consists of a set of test functions, each with a different number of decision variables and objective functions. The general form of the ZDT problem can be expressed as:

- Minimize $f_1(\boldsymbol{x})$
- Minimize $f_2(\boldsymbol{x}) = g(\boldsymbol{x}) \cdot h\big(f_1(\boldsymbol{x}), g(\boldsymbol{x})\big)$.

where $0 \leq x_i \leq 1 \; and \; i = 1, \dots, n$.

The ZDT problem is particularly challenging because the objective functions are non-linear and have conflicting objectives, so improving one objective function may result in a degradation of others.

There are several variants of the ZDT problem, each with different numbers of decision variables and objective functions. These variants are designed to test the ability of optimization algorithms to handle problems with different levels of complexity. The ZDT1 problem functions are expressed as below:

$$f_1(\boldsymbol{x}) = x_1 \tag{5}$$

$$g(\boldsymbol{x}) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i \tag{6}$$

$$h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} \tag{7}$$

with 30 variables (n = 30).

Figure 10 shows the ZDT1 model communicating with RAVEN. The RAVEN input for the ZDT1 optimization problem is shown in Appendix B.

```python
import math

def evaluate(Inputs):
  Sum = 0
  obj1 = 0

  for ind,var in enumerate(Inputs.keys()):
    # write the objective function here
    if (ind == 0) :
      obj1 += Inputs[var]
    if (ind != 0):
      Sum += Inputs[var]
  g = 1 + (9/len(Inputs.keys())*Sum )
  h = 1 - math.sqrt(obj1/g)
  obj2 = g*h
  return obj1[:], obj2[:]

def run(self,Inputs):
  """
    RAVEN API
    @ In, self, object, RAVEN container
    @ In, Inputs, dict, additional inputs
    @ Out, None
  """
  self.obj1,self.obj2 = evaluate(Inputs)
```

Figure 10. RAVEN input for the ZDT1 model.

Figure 11 shows the ZDT1 test results. After 350 simulation iterations, the solutions are converged to the theoretical optimal Pareto front. Table 3 shows the parameters used for implementing NSGA-II in RAVEN for ZDT1 problem solving. The population and generation were set large enough to apply a large input space (n=30). The probabilities of crossover and mutation were set to 0.8 and 0.9, respectively, which gave the highest level of diversity in the offspring populations.

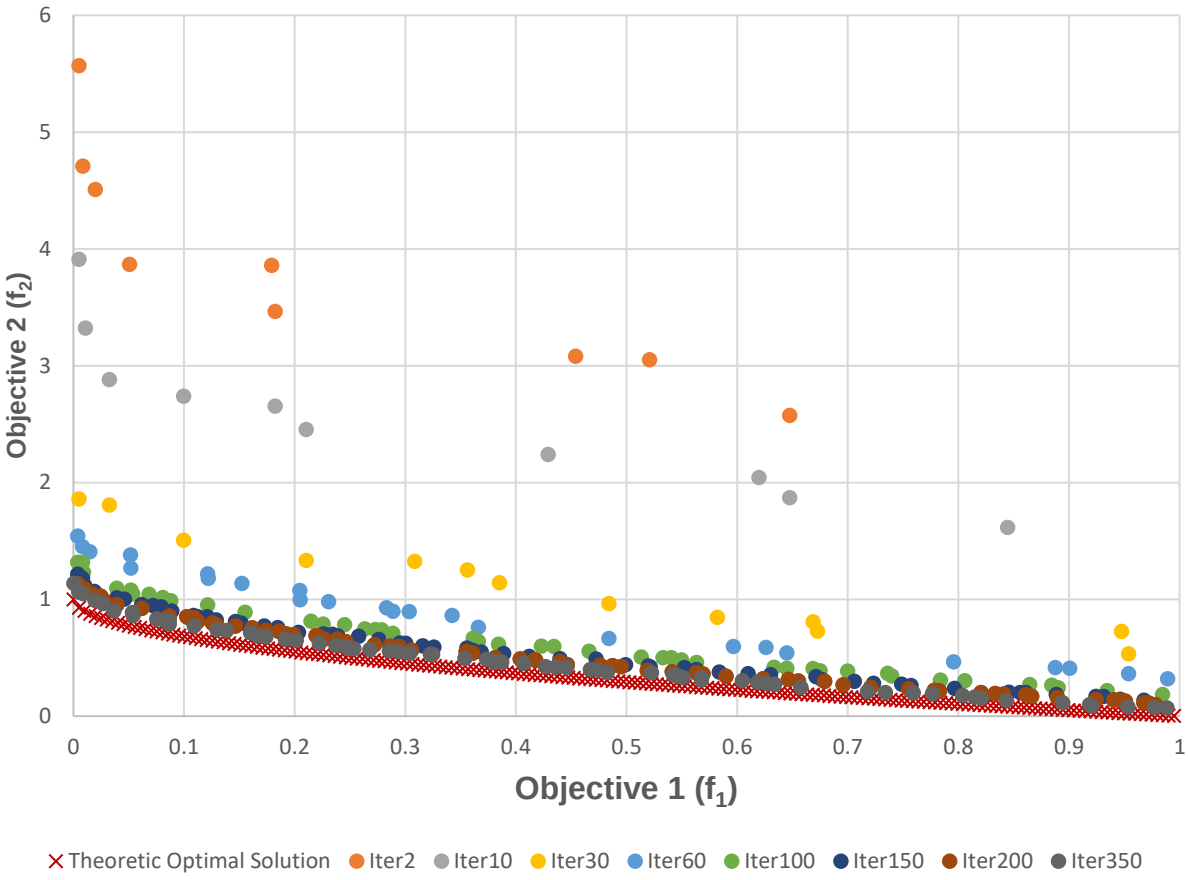

Figure 11. ZDT1 test results in RAVEN.

Table 3. NSGA-II parameters for the ZDT1 problem in RAVEN.

Population size	100	Generation	200
Crossover type	Two points crossover	Mutation type	Random mutator
Probability of crossover	0.8	Probability of mutation	0.9

3. SUMMARY AND FUTURE WORKS

The NSGA-II optimizer was implemented for the Plant Reload Optimization Platform and demonstrated with a minimization with local sum and ZDT benchmark problems. The results showed the NSGA-II optimizer in RAVEN generates Pareto-optimal solutions well and converged to the target theoretical solution. However, for the realistic plant reload optimization problems, the following models will be added:

- Implementation of the NSGA-II acceleration models:
  - Convergence acceleration scheme to improve the Pareto-optimal solutions' searching capability
  - Space exploration model to enhance the survival selection procedure in GA optimizer
- Development of the adaptive mutation and crossover algorithms and additional convergence criteria
- Investigation of any other hybrid meta-heuristic method, which may have better convergence behavior (e.g., GA coupled with the Particle Swarm method)
- Demonstration of the multi-objective optimization platform with an actual reactor core design problem.

REFERENCES

[1] Y.-J. Choi, M. Abdo, D. Mandelli, A. Epiney, J. Valeri, C. Gosdin, C. Frepoli and A. Alfonsi. 2021. “Demonstration of the Plant Fuel Reload Process Optimization for an Operating PWR.” INL/EXT-21-64549, Idaho National Laboratory.

[2] C. Rabiti, A. Alfonsi, J. Cogliati, D. Mandelli, R. Kinoshita, S. Sen, C. Wang, P. W. Talbot, D. P. Maljovec and M. G. Abdo. 2022. “RAVEN User Manual.” INL/EXT-15-34123, Idaho National Laboratory. https://doi.org/10.2172/1784874.

[3] N. Shaukat, A. Ahmad, B. Mohsin, R. Khan, S. U.-D. Khan, and S. U.-D. Khan. 2021. “Multiobjective core reloading pattern optimization of PARR-1 using modified genetic algorithm coupled with Monte Carlo methods.” Science and Technology of Nuclear Installations 2021: 1–13. https://doi.org/10.1155/2021/1802492.

[4] J. H. Holland. 1973. “Genetic algorithms and the optimal allocation of trials.” SIAM Journal on Computing 2(2): 88–105. https://doi.org/10.1137/0202009.

[5] A. Ahmad, B. Mohsin, R. Khan, S. U.-D. Khan, and S. U.-D. Khan. 2021. “Multiobjective Core Reloading Pattern Optimization of PARR-1 Using Modified Genetic Algorithm Coupled with Monte Carlo Methods.” Science and Technology of Nuclear Installations 2021: 1–13. https://doi.org/10.1155/2021/1802492.

[6] G. T. Parks. 2017. “Multiobjective Pressurized Water Reactor Reload Core Design by Nondominated Genetic Algorithm Search.” Nuclear Science and Engineering 124(1): 178–187. https://doi.org/10.13182/NSE96-A24233.

[7] J. L. C. Chapot, F. C. Da Silva, and R. Schirru. 1999. “A new approach to the use of genetic algorithms to solve the pressurized water reactor’s fuel management optimization problem,” Annals of Nuclear Energy 26(7): 641-655. https://doi.org/10.1016/S0306-4549(98)00078-4.

[8] Y.-J. Choi, M. Abdo, G. Palamone, S. Heagy, C. Frepoli, K. Ogujiuba, N. Rollins, G. Deplipei, and J. Hou. 2022. “Development and Demonstration of a Risk-Informed Approach to the Regulatory Required Fuel Reload Safety Analysis.” INL/RPT-22-68628, Idaho National Laboratory. https://doi.org/10.2172/1885790.

[9] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2022. “A fast and elitist multiobjective genetic algorithm: NSGA-II.” IEEE Transactions on Evolutionary Computation 6(20): 182–197. https://doi.org/10.1109/4235.996017.

[10] P. Wang, K. Ye, X. Hao, and J. Wang. 2023. “Combining multi-objective genetic algorithm and neural network dynamically for the complex optimization problems in physics.” Scientific Reports 13(1): 880. https://doi.org/10.1038/s41598-023-27478-7.

[11] A. Konak, D. W. Coit, and A. E. Smith. 2006. “Multi-objective optimization using genetic algorithms: A tutorial.” Reliability Engineering & System Safety 91(9): 992–1007. https://doi.org/10.1016/j.ress.2005.11.018.

[12] J. D. Schaffer. 1985. “Multiple objective optimization with vector evaluated genetic algorithms.” in International Conference on Genetic Algorithm and Their Applications. https://doi.org/10.4324/9781315799674.

[13] C. M. Fonseca and P. J. Fleming. 1993. “Multiobjective genetic algorithms.” In IEEE Colloquium on Genetic Algorithms for Control Systems Engineering, Digest No. 1993/130, IEEE, London, UK.

[14] P. Hajela and C. Y. Lin. 1992. “Genetic search strategies in multicriterion optimal design.” Struct Optimization 4(2): 99–107. https://doi.org/10.1007/BF01759923.

[15] J. Horn, N. Nafpliotis and D. E. Goldberg, “A niched Pareto genetic algorithm for multiobjective optimization.” in Proceedings of IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence, Orlando, FL. https://doi.org/10.1109/ICEC.1994.350037.

[16] T. Murata and H. Ishibuchi. 1995. “MOGA: multi-objective genetic algorithms.” In IEEE International Conference on Evolutionary Computation, Perth, WA, Australia. https://doi.org/10.1109/ICEC.1995.489161.

[17] D. W. Corne, J. Knowles, and M. Oates. 2000. “The Pareto envelope-based selection algorithm for multiobjective optimization.” in 6th International Conference on Parallel Problem Solving, Paris, France. https://doi.org/10.1007/3-540-45356-3_82.

[18] J. Knowles and. D. Corne. 1999. “The Pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimization.” In Proceedings of the 1999 Congress on Evolutionary Computation, Washington, DC. https://doi.org/10.1109/CEC.1999.781913.

[19] N. Srinivas and K. Deb. 1994. “Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms.” Evolutionary Computation 2(3): 221–248. https://doi.org/10.1162/evco.1994.2.3.221.

[20] K. Deb, S. Agrawal, A. Pratap, and. T. Meyarivan. 2000. “A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II.” In 6th International Conference on Parallel Problem Solving from Nature, Paris, France. https://doi.org/10.1007/3-540-45356-3_83.

[21] E. Zitzler and L. Thiele. 1999. “Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach.” IEEE Transactions on Evolutionary Computation 3(4): 257–271. https://doi.org/10.1109/4235.797969.

[22] E. Zitzler, M. Laumanns, and L. Thiele. 2001. “SPEA2: improving the strength Pareto evolutionary algorithm.” Swiss Federal Institute Techonology, Zurich, Switzerland. https://doi.org/10.3929/ethz-a-004284029.

[23] H. Lu and G. Yen. 2003. “Rank-density-based multiobjective genetic algorithm and benchmark test function study.” IEEE Transactions on Evolutionary Computation 7(4): 325–343. https://doi.org/10.1109/TEVC.2003.812220.

[24] G. Yen and H. Lu. 2003. “Dynamic multiobjective evolutionary algorithm: adaptive cell-based rank and density estimation.” IEEE Transactions on Evolutionary Computation 7(3): 253–274. https://doi.org/10.1109/TEVC.2003.810068.

[25] E. Zitzler, K. Deb, and L. Thiele. 2000. “Comparison of Multiobjective Evolutionary Algorithms: Empirical Results.” Evolutionary Computation 8(2): 173–195. https://doi.org/10.1162/106365600568202.

Appendix A
RAVEN Input for Multi-sum Optimization Problem

```
<?xml version="1.0"?>
<Simulation verbosity="debug" profile="jobs">
  <TestInfo>

<name>\raven\tests\framework\Optimizers\NSGAII\di
screte\constrained\</name>
  <author>Junyung Kim, Mohammad Abdo </author>
  <created>2022-12-21</created>
  <classesTested/>
  <description>NSGA-II min-min test
</description>
  </TestInfo>

  <RunInfo>

<WorkingDir>Multi_MinwoReplacement_Figure</Workin
gDir>
  <Sequence>optimize,print</Sequence>
  <batchSize>2</batchSize>
  </RunInfo>

  <Steps>
  <MultiRun name="optimize" re-seeding="2286">
  <Input class="DataObjects"
type="PointSet">placeholder</Input>
  <Model class="Models"
type="ExternalModel">myLocalSum</Model>
  <Optimizer class="Optimizers"
type="GeneticAlgorithm">GAopt</Optimizer>
  <SolutionExport class="DataObjects"
type="PointSet">opt_export</SolutionExport>
  <Output class="DataObjects"
type="PointSet">optOut</Output>
  <Output class="OutStreams"
type="Print">opt_export</Output>
  </MultiRun>
  <IOStep name="print">
  <Input class="DataObjects"
type="PointSet">opt_export</Input>
  <Input class="DataObjects"
type="PointSet">optOut</Input>
  <Output class="OutStreams"
type="Print">opt_export</Output>
  <Output class="OutStreams"
type="Print">optOut</Output>
  </IOStep>
  </Steps>

  <Models>
  <ExternalModel
ModuleToLoad="../myLocalSum_multi.py"
name="myLocalSum" subType="">

<variables>x1,x2,x3,x4,x5,x6,obj1,obj2</variables
>
  </ExternalModel>
  </Models>

  <Functions>
  <External file="../myConstraints.py"
name="expConstr3">
  <variables>x1,x2,x3,x4,x5,x6</variables>
  </External>
  <External file="../myConstraints.py"
name="impConstr3">
  <variables>x1,x2,x3,x4,x5,x6,obj1</variables>
  </External>
  </Functions>

  <Distributions>
  <UniformDiscrete name='woRep_dist'>
  <lowerBound>2</lowerBound>
  <upperBound>7</upperBound>
  <strategy>withoutReplacement</strategy>
  </UniformDiscrete>
  </Distributions>

  <Optimizers>
  <GeneticAlgorithm name="GAopt">
  <samplerInit>
  <limit>50</limit>
  <initialSeed>42</initialSeed>
  <writeSteps>every</writeSteps>
  <type>min</type>
  </samplerInit>

  <GAparams>
  <populationSize>30</populationSize>

<parentSelection>tournamentSelection</parentSelec
tion>
  <reproduction>
  <crossover type="twoPointsCrossover">
  <crossoverProb>0.7</crossoverProb>
  </crossover>
  <mutation type="randomMutator">
  <mutationProb>0.7</mutationProb>
  </mutation>
  </reproduction>
  <fitness type="rank_crowding">
  </fitness>

<survivorSelection>rankNcrowdingBased</survivorSe
lection>
  </GAparams>
  <convergence>
  <AHDp>0.0</AHDp>
  </convergence>
  <variable name="x1">
  <distribution>woRep_dist</distribution>
  </variable>
  <variable name="x2">
  <distribution>woRep_dist</distribution>
  </variable>
  <variable name="x3">
  <distribution>woRep_dist</distribution>
  </variable>
  <variable name="x4">
  <distribution>woRep_dist</distribution>
  </variable>
  <variable name="x5">
  <distribution>woRep_dist</distribution>
  </variable>
  <variable name="x6">
  <distribution>woRep_dist</distribution>
  </variable>
```

```
  <objective>obj1, obj2 </objective>
  <TargetEvaluation class="DataObjects"
type="PointSet">optOut</TargetEvaluation>
  <Sampler class="Samplers"
type="MonteCarlo">MC_samp</Sampler>
  <Constraint class='Functions'
type='External'>expConstr3</Constraint>
  <ImplicitConstraint class='Functions'
type='External'>impConstr3</ImplicitConstraint>
  </GeneticAlgorithm>
  </Optimizers>

  <Samplers>
  <MonteCarlo name="MC_samp">
  <samplerInit>
  <limit>30</limit>
  <initialSeed>050877</initialSeed>
  </samplerInit>
  <variable name="x1">
  <distribution>woRep_dist</distribution>
  </variable>
  <variable name="x2">
  <distribution>woRep_dist</distribution>
  </variable>
  <variable name="x3">
  <distribution>woRep_dist</distribution>
  </variable>
  <variable name="x4">
  <distribution>woRep_dist</distribution>
  </variable>
  <variable name="x5">
  <distribution>woRep_dist</distribution>
  </variable>
  <variable name="x6">
  <distribution>woRep_dist</distribution>
  </variable>
  </MonteCarlo>
  </Samplers>

  <DataObjects>
  <PointSet name="placeholder"/>
  <PointSet name="optOut">
  <Input>x1,x2,x3,x4,x5,x6</Input>
  <Output>obj1,obj2</Output>
  </PointSet>
  <PointSet name="opt_export">
  <Input>trajID</Input>

<Output>x1,x2,x3,x4,x5,x6,obj1,obj2,age,batchId,r
ank,CD,ConstraintEvaluation_expConstr3,
ConstraintEvaluation_impConstr3,fitness,accepted
</Output>
  </PointSet>
  </DataObjects>

  <OutStreams>
  <Print name="optOut">
  <type>csv</type>
  <source>optOut</source>
  </Print>
  <Print name="opt_export">
  <type>csv</type>
  <source>opt_export</source>
  <clusterLabel>trajID</clusterLabel>
  </Print>
  </OutStreams>
    </Simulation>
```

Appendix B
RAVEN input for ZDT1 optimization problem

```
<?xml version="1.0"?>
<Simulation verbosity="debug" profile="jobs">
 <TestInfo>

<name>raven\tests\framework\Optimizers\NSGAII\dis
crete\constrained\</name>
  <author>Junyung Kim, Mohammad Abdo </author>
  <created>2023-02-21</created>
  <classesTested/>
  <description>ZDT1 test using NSGA-
II</description>
  </TestInfo>

  <RunInfo>
  <WorkingDir>ZDT1_result</WorkingDir>
  <Sequence>optimize,print</Sequence>
  <batchSize>1</batchSize>
  </RunInfo>

  <Steps>
  <MultiRun name="optimize" re-seeding="2286">
  <Input class="DataObjects"
type="PointSet">placeholder</Input>
  <Model class="Models"
type="ExternalModel">ZDT</Model>
  <Optimizer class="Optimizers"
type="GeneticAlgorithm">GAopt</Optimizer>
  <SolutionExport class="DataObjects"
type="PointSet">opt_export</SolutionExport>
  <Output class="DataObjects"
type="PointSet">optOut</Output>
  <Output class="OutStreams"
type="Print">opt_export</Output>
  </MultiRun>
  <IOStep name="print">
  <Input class="DataObjects"
type="PointSet">opt_export</Input>
  <Input class="DataObjects"
type="PointSet">optOut</Input>
  <Output class="OutStreams"
type="Print">opt_export</Output>
  <Output class="OutStreams"
type="Print">optOut</Output>
  </IOStep>
  </Steps>

  <Models>
  <ExternalModel ModuleToLoad="../ZDT_model.py"
name="ZDT" subType="">

      <variables>x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,
      x11,x12,x13,x14,x15,x16,x17,x18,x19,x20,x2
      1,x22,x23,x24,x25,x26,x27,x28,x29,x30,obj1
      ,obj2</variables>
  </ExternalModel>
  </Models>

  <Distributions>
  <Uniform name='unifDist'>
  <lowerBound>0</lowerBound>
  <upperBound>1</upperBound>
  </Uniform>
  </Distributions>

  <Optimizers>
  <GeneticAlgorithm name="GAopt">
  <samplerInit>
  <limit>200</limit>
  <initialSeed>42</initialSeed>
  <writeSteps>every</writeSteps>
  <type>min</type>
  </samplerInit>

  <GAparams>
  <populationSize>100</populationSize>

<parentSelection>tournamentSelection</parentSelec
tion>
  <reproduction>
  <crossover type="twoPointsCrossover">
  <crossoverProb>0.8</crossoverProb>
  </crossover>
  <mutation type="randomMutator">
  <mutationProb>0.9</mutationProb>
  </mutation>
  </reproduction>
  <fitness type="rank_crowding">
  </fitness>

<survivorSelection>rankNcrowdingBased</survivorSe
lection>
  </GAparams>

  <convergence>
  <AHDp>0.0</AHDp>
  </convergence>

  <variable name="x1">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x2">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x3">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x4">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x5">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x6">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x7">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x8">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x9">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x10">
  <distribution>unifDist</distribution>
```

```
  </variable>

  <variable name="x11">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x12">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x13">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x14">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x15">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x16">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x17">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x18">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x19">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x20">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x21">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x22">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x23">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x24">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x25">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x26">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x27">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x28">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x29">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x30">
  <distribution>unifDist</distribution>
  </variable>
  <objective>obj1, obj2 </objective>
  <TargetEvaluation class="DataObjects"
type="PointSet">optOut</TargetEvaluation>
  <Sampler class="Samplers"
type="MonteCarlo">MC_samp</Sampler>
  </GeneticAlgorithm>
  </Optimizers>
```

```
<Samplers>
<MonteCarlo name="MC_samp">
<samplerInit>
<limit>100</limit>
<initialSeed>050877</initialSeed>
</samplerInit>
<variable name="x1">
<distribution>unifDist</distribution>
</variable>
<variable name="x2">
<distribution>unifDist</distribution>
</variable>
<variable name="x3">
<distribution>unifDist</distribution>
</variable>
<variable name="x4">
<distribution>unifDist</distribution>
</variable>
<variable name="x5">
<distribution>unifDist</distribution>
</variable>
<variable name="x6">
<distribution>unifDist</distribution>
</variable>
<variable name="x7">
<distribution>unifDist</distribution>
</variable>
<variable name="x8">
<distribution>unifDist</distribution>
</variable>
<variable name="x9">
<distribution>unifDist</distribution>
</variable>
<variable name="x10">
<distribution>unifDist</distribution>
</variable>
<variable name="x11">
<distribution>unifDist</distribution>
</variable>
<variable name="x12">
<distribution>unifDist</distribution>
</variable>
<variable name="x13">
<distribution>unifDist</distribution>
</variable>
<variable name="x14">
<distribution>unifDist</distribution>
</variable>
<variable name="x15">
<distribution>unifDist</distribution>
</variable>
<variable name="x16">
<distribution>unifDist</distribution>
</variable>
<variable name="x17">
<distribution>unifDist</distribution>
</variable>
<variable name="x18">
<distribution>unifDist</distribution>
</variable>
<variable name="x19">
<distribution>unifDist</distribution>
</variable>
<variable name="x20">
<distribution>unifDist</distribution>
</variable>
<variable name="x21">
<distribution>unifDist</distribution>
```

```
  </variable>
  <variable name="x22">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x23">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x24">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x25">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x26">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x27">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x28">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x29">
  <distribution>unifDist</distribution>
  </variable>
  <variable name="x30">
  <distribution>unifDist</distribution>
  </variable>
  </MonteCarlo>
  </Samplers>

  <DataObjects>
  <PointSet name="placeholder"/>
  <PointSet name="optOut">

      <Input>x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,
      x12,x13,x14,x15,x16,x17,x18,x19,x20,x21,x2
      2,x23,x24,x25,x26,x27,x28,x29,x30</Input>
  <Output>obj1,obj2</Output>
  </PointSet>
  <PointSet name="opt_export">
  <Input>trajID</Input>

      <Output>x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11
      ,x12,x13,x14,x15,x16,x17,x18,x19,x20,x21,x
      22,x23,x24,x25,x26,x27,x28,x29,x30,obj1,ob
      j2,age,batchId,rank,CD,fitness,accepted
      </Output><!--Modify if necessary
      CD,iteration,accepted,conv_AHDp-->
  </PointSet>
  </DataObjects>

  <OutStreams>
  <Print name="optOut">
  <type>csv</type>
  <source>optOut</source>
  </Print>
  <Print name="opt_export">
  <type>csv</type>
  <source>opt_export</source>
  <clusterLabel>trajID</clusterLabel>
  </Print>
  </OutStreams>
</Simulation>
```