Integration of Multi-Objective Genetic Algorithm and neural networks in linac optimization

KIM Chanmi 1, PARK Chong Shik *1, KIM Eun San

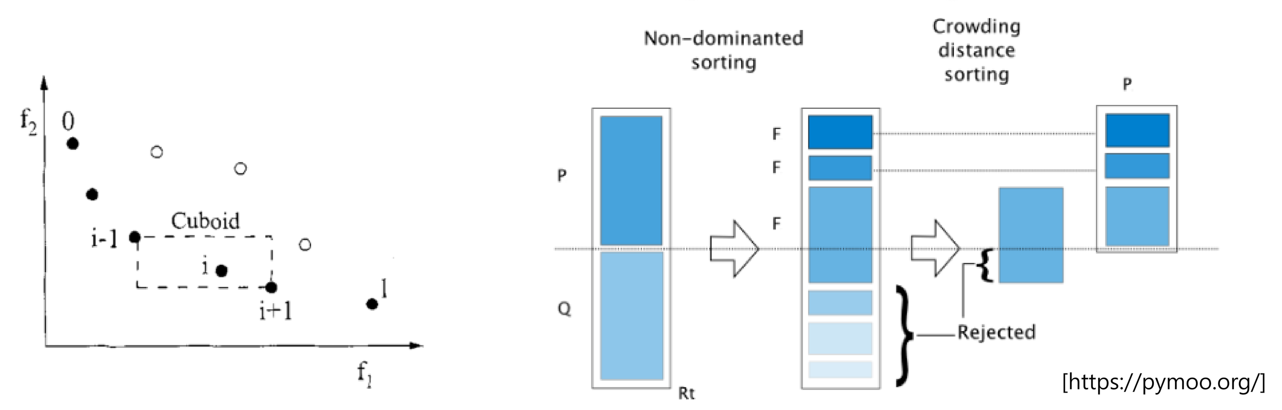
Dept. of Accelerator Science, Korea University, Sejong, South Korea

Abstract

Multi-Objective Genetic Algorithm(MOGA) is one of promising approach for optimizing nonlinear beam dynamics in accelerators. For explorative problems that have many variables and local optima, however the performance of MOGA is not always satisfactory. To improve the efficiency of optimization in linac beam line, we propose a novel integration of MOGA and neural networks. The neural network is trained with the data produced in the evolution of the MOGA. The objective values of the offspring are estimated with the trained neural network. Based on the estimated results, those offspring are ranked with the nondominated sorting method. We therefore propose a novel Machine Learning technique in which nonlinear tracking is replaced by two well-trained neural networks to beam line lattice.

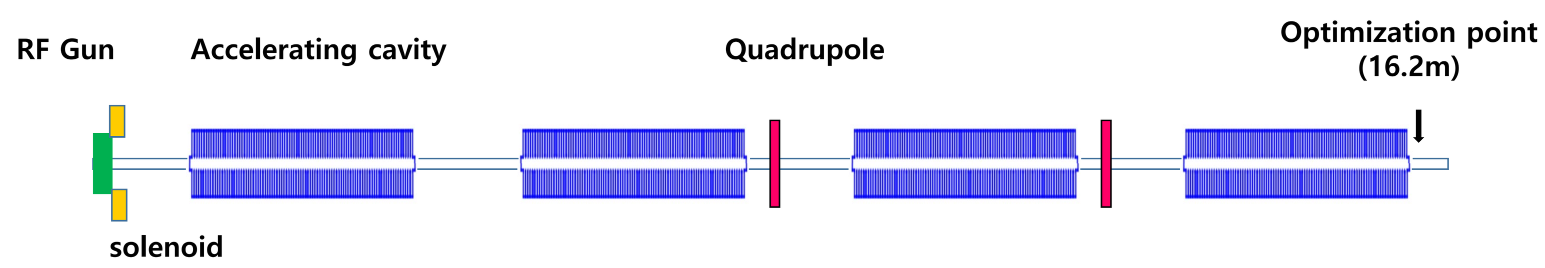
▶ Multi-Objective Genetic Algorithm (NSGA-II algorithm)

NSGA-II: Non-dominated Sorting Genetic Algorithm



The algorithm follows the general outline of a genetic algorithm with a modified mating and survival selection. In NSGA-II, first, individuals are selected frontwise. By doing so, there will be the situation where a front needs to be split because not all individuals are allowed to survive. In this splitting front, solutions are selected based on crowding distance. The crowding distance is the Manhatten Distance in the objective space. However, the extreme points are desired to be kept every generation and, therefore, get assigned a crowding distance of infinity. Furthermore, to increase some selection pressure, NSGA-II uses a binary tournament mating selection. Each individual is first compared by rank and then crowding distance.

▶ layout of the electron linac



- RF photocathode gun
- 4 accelerating structures
- One solenoid & 2 quadrupole

▶ Optimization requirement (problem functions and variables)

Objectives (3)		Unit
Normalized rms emittance X&Y	Minimization	mm-mrad
rms energy spread	Minimization	keV

Constraint (7)		Unit
Beam size X &Y (rms)	< 0.3	mm
Divergence X&Y	< 0.2663	mrad
Bunch length	< 1.0	mm
Transmission rate	>99.99	%
Average energy	>200	MeV

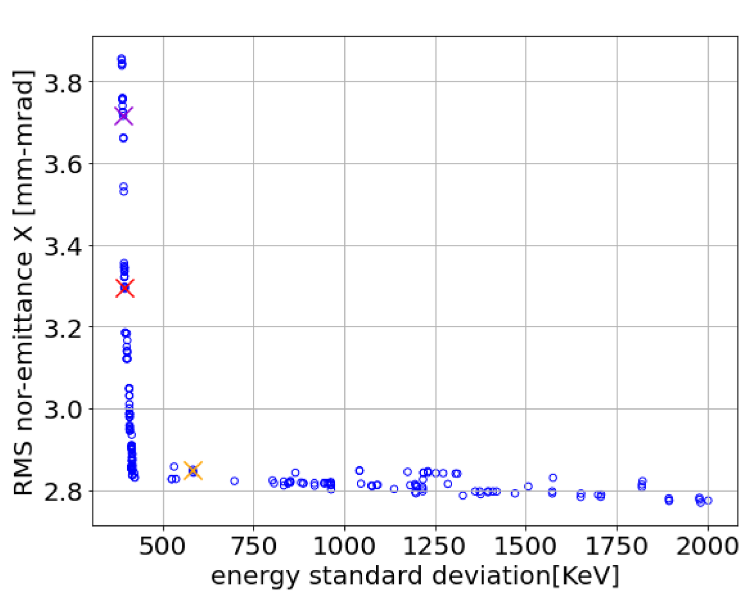
Variables (6)	range	unit
RF gun phase	0~360	degree
Acc1&2 phase	0~360	degree
Acc3&4 phase	0~360	degree
Solenoid	0.15~0.21	T
Quad 1&2	1~3(-4~-1)	T/m

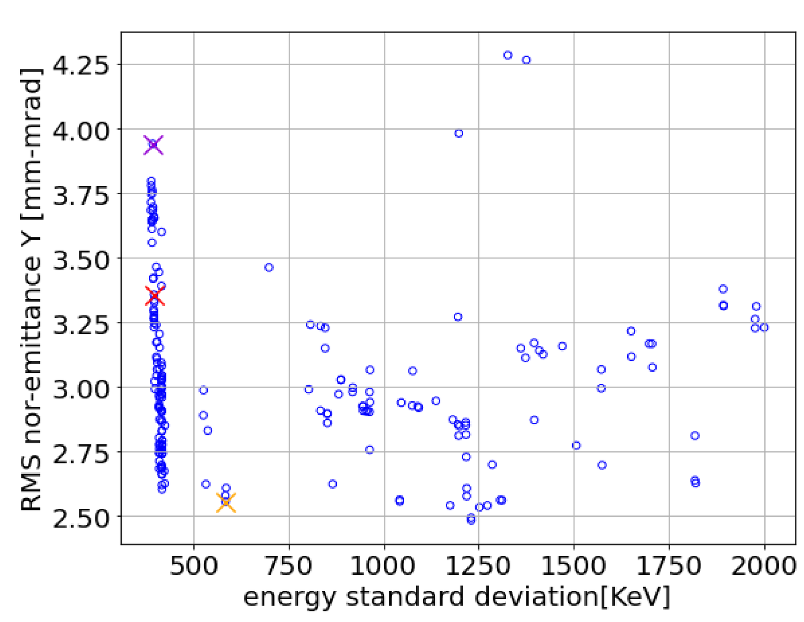
Beam tracking code
: ASTRA code
Optimization algorithm
: Multi-Objective Genetic Algorithm (MOGA)
Tool
: pymoo (base : python)

Population : 500 / offspring : 250/ generation : 300

▶ Result of optimization

Objective space

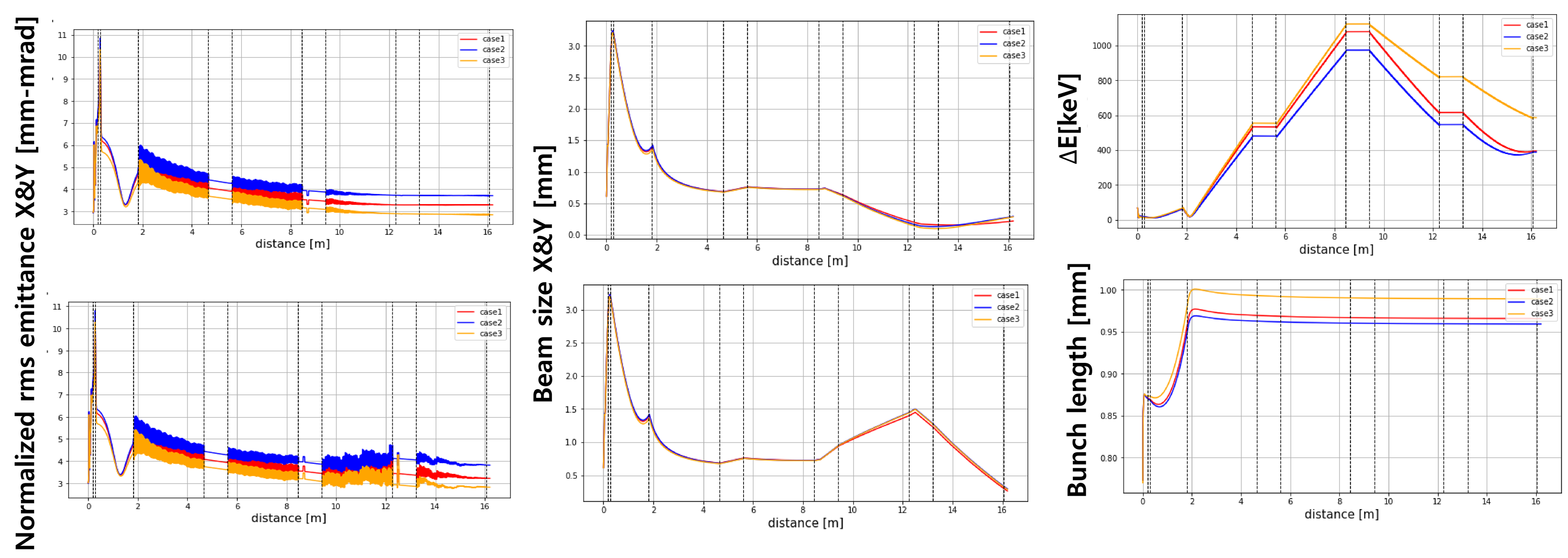




	longitudinal	Transverse	
	Energy spread	Emittance X	Emittance Y
Case1	0.5	0.25	0.25
Case2	0.8	0.1	0.1
case3	0.3	0.35	0.35

The optimized **Pareto front** data in objective space
The three cases selected to compare beam tracking simulation based on the importance of objectives
The objective weights are determined by the ratios of the three objectives

▶ beam tracking simulation



mode	Energy spread [keV]	emittanceX [mm-mrad]	geometric emittance X [nm]	emittanceY [mm-mrad]	geometric emittance Y [nm]	Xrms [mm]	Yrms [mm]	L_bunch [mm]	Transmission rate [%]	average E [MeV]
Case1	381.50	3.2990	8.44	3.2145	8.18	0.21760	0.26149	0.96584	100	200.16
Case2	372.17	3.7122	9.52	3.8144	9.71	0.29162	0.29447	0.95932	100	200.61
Case3	584.24	2.8520	7.37	2.8147	7.21	0.28593	0.28566	0.98924	100	200.92
	<1MeV(0.5%)		<10nm		<10nm			< 1mm		

▶ machine learning using MOGA data

Input data : RF gun phase, acc1&2 phase, acc3&4 phase : 3
Output data : rms energy spread : 1
Total data(train/test) : 10000 (7000/3000)
Moga generation : 200~220
Layer : 6

Case 1 :
train data transform type : normalize
Activation function : ReLU (Rectified Linear Unit)

Case 2 :
train data transform type : raw data
Activation function : Tanh (Hyperbolic Tangent)

Output : energy spread
RF gun phase : 120~210 (step : 10)
Acc1&2 phase and acc 3&4 phase fix

case1	case2
2553.077881	353.7229309
2602.841309	353.7229309
2652.60498	353.7229309
2702.368164	353.7229309
2752.131836	353.7230835
2801.89502	353.7247009
2851.658203	353.724823
2901.421875	353.724823
2951.185547	353.724823
3000.948486	353.724823

Input data : RF gun phase, acc1&2 phase, acc3&4 phase : 3
Output data : rms energy spread : 1
Total data(train/test) : 10000 (7000/3000)
layer : 6
Activation function : Tanh

Training data range

generation	14	200	281
RF gun	67.26912	0.049265	0.049265
acc1&2	96.30259	1.124652	0.896699
acc3&4	349.5431	8.961445	10.19618
solenoid	0.075308	1.84E-05	1.84E-05
quad 1	4.732247	0.035038	0.041793
quad 2	4.905732	0.029668	0.026938

Moga generation (step :20)
RF gun phase (120~210)

14	200	281
1871.229492	353.7229309	398.4751587
1871.229492	353.7229309	398.4751587
1871.229492	353.7229309	398.4751587
1871.229492	353.7229309	398.4751587
1871.229492	353.7230835	398.4752197
1871.229492	353.7247009	398.4762573
1871.229492	353.724823	398.4768066
1871.229492	353.724823	398.4768066
1871.229492	353.724823	398.4768066
1871.229492	353.724823	398.4768066

Moga generation (281)
Acc1&2 phase(0~100)

281
398.4760132
398.4760132
398.4760132
398.4760132
398.4760132
398.4761047
398.4768066
398.4768066
398.4768066
398.4768066

Input data : RF gun phase, acc1&2 phase, acc3&4 phase : 6
Output data : rms energy spread (or emittance) : 1
Total data(train/test) : 10000 (7000/3000)
Activation function : Tanh

RF gun phase (120~210)
Train output data
Energy spread : 398.80032
Emittance : 3.7936825

ASTRA tracking simulation output

RF gun phase	energy spread	emittance x	transmission rate
210	781.71	6.0387	100
200	568.66	7.1091	100
190	1882.2	10.614	85.95
180	2791.1	2.3695	28.67

The output values for input variables reflecting the train data are different from the results of beam tracking simulation.

▶ Future plan

- We need to find the reason why the train results and beam tracking simulation results are different.
  : Increasing the number of train data
  : Obtain data suitable for machine learning from moga
  : Adjusting other hyper-parameters used in machine learning
  (number of layers, Hidden_size, Epochs, Learning_rate, Data_rate, etc.)

KOREA UNIVERSITY