**Complete Code : FW-NSGA-II**

import random  
import openpyxl  
  
# Step 1: Initialize the algorithm parameters in NSGA-II.  
population\_size = 500  
max\_iterations = 1000  
crossover\_probability = 0.5  
mutation\_probability = 0.5  
  
# Define the chromosome coding method and other necessary parameters  
  
class Individual:  
 def \_\_init\_\_(self, chromosome):  
 self.chromosome = chromosome  
 self.fitness = None  
  
def initialize\_population():  
 # Generate the initial parent population  
 population = []  
 for \_ in range(population\_size):  
 # Create a new individual  
 individual = create\_individual()  
 population.append(individual)  
 return population  
  
def create\_individual():  
 # Create a new individual (chromosome) based on the coding method  
 # Return the individual  
  
def evaluate\_fitness(individual):  
 # Calculate the fitness value for an individual based on the problem domain  
 # Return the fitness value  
  
def crossover(parent1, parent2):  
 # Perform crossover between two parents to generate offspring  
 # Return the offspring  
  
def mutate(individual):  
 # Perform mutation on an individual  
 # Return the mutated individual  
  
def selection(population):  
 # Perform selection to create the mating pool for reproduction  
 # Return the selected individuals  
  
def non\_dominated\_sorting(population):  
 # Perform non-dominated sorting on the population  
 # Assign non-dominance ordinal values to individuals  
 # Return the sorted population  
  
def calculate\_crowding\_distance(front):  
 # Calculate crowding distance for individuals in a front  
 # Update crowding distance values for the individuals in the front  
  
def elite\_retention(population, offspring):  
 # Perform elite retention strategy to select individuals for the next generation  
 # Return the updated population  
  
# Read parameters from Excel  
def read\_parameters\_from\_excel(file\_path):  
 workbook = openpyxl.load\_workbook(file\_path)  
 # Read and assign the parameters from the Excel file  
  
# Save results to Excel  
def save\_results\_to\_excel(results, file\_path):  
 workbook = openpyxl.Workbook()  
 sheet = workbook.active  
 # Write the results to the Excel file  
 workbook.save(file\_path)  
  
# Main code:  
if \_\_name\_\_ == "\_\_main\_\_":  
 # Read parameters from Excel  
 parameters\_file = r'C:\Users\junxi\OneDrive - UNSW\Desktop\modeling RTNDP\20230617\parameters.xlsx'  
 read\_parameters\_from\_excel(parameters\_file)  
  
 # Step 1: Initialize the algorithm parameters in NSGA-II.  
 population = initialize\_population()  
  
 # Step 2: S-ETNDP road traffic flow assignment and fitness value calculation.  
 for individual in population:  
 fitness = evaluate\_fitness(individual)  
 individual.fitness = fitness  
  
 # Step 3: Population update using NSGA-II algorithm.  
 for iteration in range(max\_iterations):  
 offspring = []  
 while len(offspring) < population\_size:  
 # Perform selection, crossover, and mutation to generate offspring  
 parent1, parent2 = selection(population)  
 child = crossover(parent1, parent2)  
 mutated\_child = mutate(child)  
 offspring.append(mutated\_child)  
  
 # Step 4: Execute operations using NSGA-II algorithm.  
 merged\_population = population + offspring  
 non\_dominated\_population = non\_dominated\_sorting(merged\_population)  
 for front in non\_dominated\_population:  
 calculate\_crowding\_distance(front)  
  
 # Step 4.3: Elite retention strategy.  
 population = elite\_retention(population, offspring)  
  
 # Step 5: Determine if the maximum number of iterations has been reached.  
 if iteration == max\_iterations - 1:  
 # Step 6: The Pareto optimal solutions are output and the FW-NSGA-II terminates.  
 pareto\_front = non\_dominated\_population[0]  
 # Output the Pareto optimal solutions  
  
 # Save results to Excel  
 results\_file = r'C:\Users\junxi\OneDrive - UNSW\Desktop\modeling RTNDP\20230617\results.xlsx'  
 save\_results\_to\_excel(results, results\_file)