Search Based Software Engineering

Exercise 04 - Multi Objective Optimization - Pareto Front

2019-06-18

Deadline: 2019-07-01 23:59

Submit to: andre.karge@uni-weimar.de

Submission details:

- **This assignment is for the students of the Digital Engineering program!**
- compress your files (.zip or .tar.gz or .rar)
- Include a text file with the names, matrikel numbers and degree program for each group member!
- **Submit a .py file for your solution (no .ipynb files!)**
- **Submit a .pdf file for theory tasks.**

Name your compressed file: <lastname>_<firstname>_<matrikelnummer>-ex<exercise-number>(.tar.gz or .rar or .zip)
or for more than one student: please use this format for all group members
example: norris_chuck_123456-schwarzenegger_arnold_121212-ex01.tar.gz

Groups: submit your solved assignment in **groups of 2**

Language: Python 3

Problem Description

Task 1. Pareto Front (7 points)

a) What are the Pareto Front and Pareto Dominance? Explain in your own words!

b) Given the following table, which individuals are dominated and which are non-dominated?

Feature	min. ↓ /max. ↑ problem	individual 1	individual 2	individual 3
Performance in s	↓	125	97	224
Memory in s	↓	80	97	50
Energy in J	↓	2150	1850	5150
Reliability in d	↑	238	138	538
ranking	↓	4	2	3

Task 2. Extended (9 points)

a) What is sparsity? Explain in your own words and give an example!

b) What is pareto strength? Explain in your own words and give an example!

c) What is wimpiness? Explain in your own words and give an example!

Hint: you can also use a visualization for your examples

Task 3. Pareto Front Algorithm (10 points)

a) Implement a python function that computes the pareto front ranks for each individual of a population of m individuals and n numeric features (where n > 10).
Create a class for individuals which holds the gene-vector and the pareto front rank of an individual and use this class when working with individuals!
Make sure that you have an appropriate initialization function and fitness assessment (keep in mind we are in multi objective optimization, so different features can have different optimization criteria)!

```
def calculate_pareto_front_rank(population):
    """
    Calculates the pareto front rank for all
    individuals in population.

    Keyword arguments:
    population -- a list holding all individuals
                  of type Individual
                  for the current population
    """
    pass

def print_pareto_ranks(population):
    """
    Iterate over the given population and
    print the pareto front rank for each
    individual

    Keyword arguments:
    population -- a list holding all individuals
                  of type Individual
                  for the current population
    """
    pass
```