Project II presentation

Víctor Sánchez Anguix (vicsana1@upv.es)

DEPARTAMENTO DE ESTADÍSTICA E INV. OPERAT. APLICADAS Y CALIDAD

Portfolio investment

$$\max Z: \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_i w_j d_{i,j}$$

$s.t.$

$$[Number\ of\ investments] \sum_{i=1}^{n} x_i = k;$$

$$[Invest\ all\ budget] \sum_{i=1}^{n} w_i = 1;$$

$$[Expected\ return] \sum_{i=1}^{n} r_i w_i \geq R;$$

$$[Linking\ constraint\ asset\ i]\ \epsilon x_i \leq w_i \leq x_i; i = 1 \ldots n$$

General overview

- Review of relevant articles and papers (encouraged)
- Design and implementation of one or several metaheuristics
- Hyperparameter optimization
- Project report
- Submission of your best metaheuristic to the competition

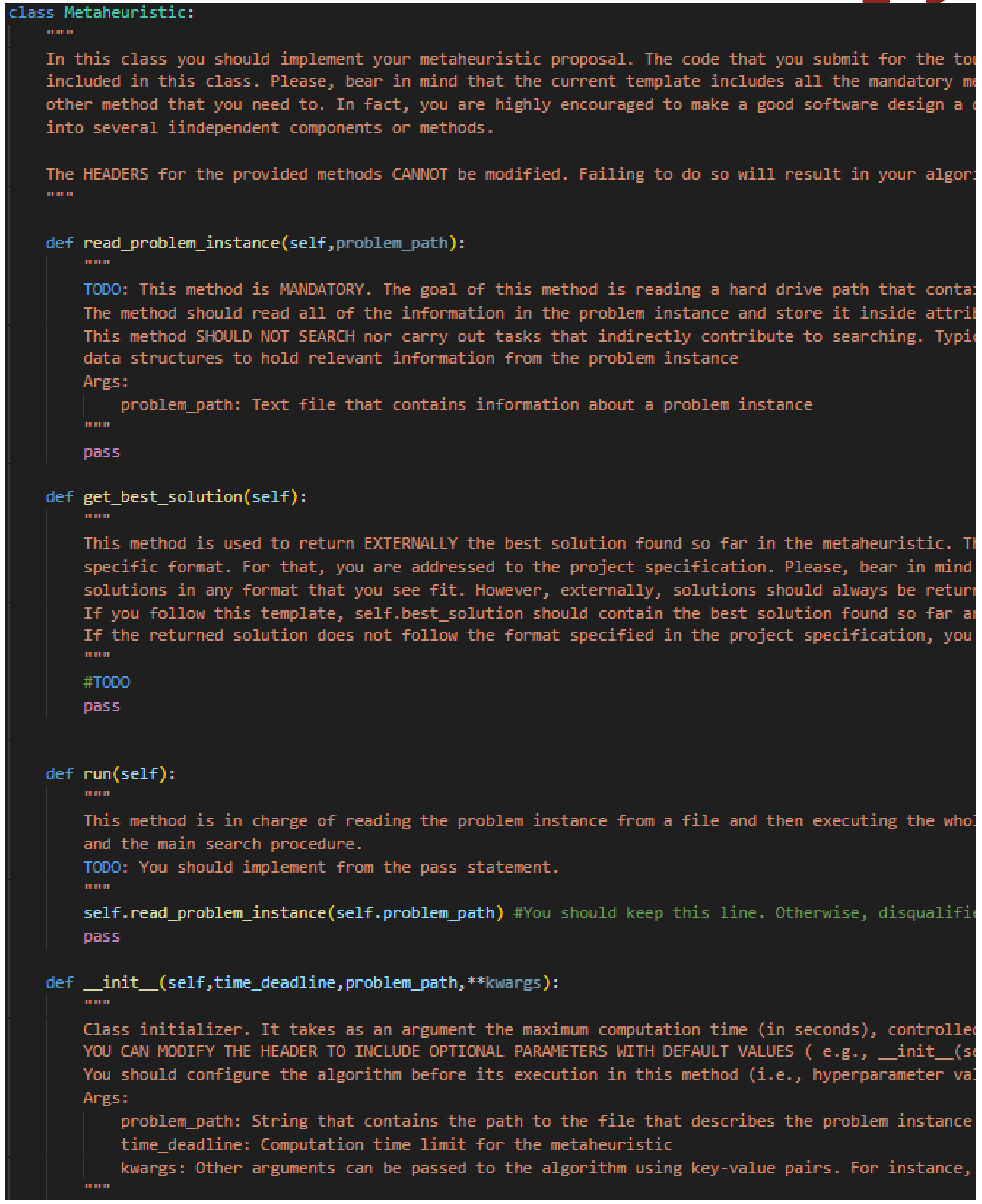
Problem instances

```
{
        "n": 4,
        "k": 2,
        "r": [1.1, 1.05, 1.2, 1.4],
        "R": 1.11,
        "dij": [ [0, 0.55, 0.89, 0.34],
                 [0.55, 0, 0.33, 0.11],
                 [0.89, 0.33, 0, 0.67],
                 [0.34, 0.11, 0.67, 0]]

}
```

Submission of metaheuristic.py

- The constructor of the metaheuristic will take two compulsory arguments: the path and the maximum computation time.
- The constructor can take other optional arguments to configure the algorithm. However, the default values for these parameters should be the best values found in the experimental optimization of the algorithm. This may be useful to reuse code and logic of the metaheuristic.
- The class will implement the method *get_best_solution*, that will return the best solution found up to this point in the metaheuristic (the run may have not finished!). The method will return a list of n float numbers. **Otherwise, your submission will not participate in the tournament**.
- The method *run* will implement the logic of the metaheuristic (i.e., population initialization, evolutionary loop, etc.).
- The class will implement the *read_problem_instance* method, that will read the information of the problem to be solved. **No precomputations or search are allowed in this method**.



```
class Metaheuristic:
    """
    In this class you should implement your metaheuristic proposal. The code that you submit for the to
    included in this class. Please, bear in mind that the current template includes all the mandatory me
    other method that you need to. In fact, you are highly encouraged to make a good software design a
    into several iindependent components or methods.

    The HEADERS for the provided methods CANNOT be modified. Failing to do so will result in your algor
    """

    def read_problem_instance(self,problem_path):
        """
        TODO: This method is MANDATORY. The goal of this method is reading a hard drive path that conta
        The method should read all of the information in the problem instance and store it inside attri
        This method SHOULD NOT SEARCH nor carry out tasks that indirectly contribute to searching. Typi
        data structures to hold relevant information from the problem instance
        Args:
            problem_path: Text file that contains information about a problem instance
        """
        pass

    def get_best_solution(self):
        """
        This method is used to return EXTERNALLY the best solution found so far in the metaheuristic. T
        specific format. For that, you are addressed to the project specification. Please, bear in mind
        solutions in any format that you see fit. However, externally, solutions should always be retur
        If you follow this template, self.best_solution should contain the best solution found so far a
        If the returned solution does not follow the format specified in the project specification, you
        """
        #TODO
        pass


    def run(self):
        """
        This method is in charge of reading the problem instance from a file and then executing the who
        and the main search procedure.
        TODO: You should implement from the pass statement.
        """
        self.read_problem_instance(self.problem_path) #You should keep this line. Otherwise, disqualifi
        pass

    def __init__(self,time_deadline,problem_path,**kwargs):
        """
        Class initializer. It takes as an argument the maximum computation time (in seconds), controlle
        YOU CAN MODIFY THE HEADER TO INCLUDE OPTIONAL PARAMETERS WITH DEFAULT VALUES ( e.g., __init__(se
        You should configure the algorithm before its execution in this method (i.e., hyperparameter va
        Args:
            problem_path: String that contains the path to the file that describes the problem instance
            time_deadline: Computation time limit for the metaheuristic
            kwargs: Other arguments can be passed to the algorithm using key-value pairs. For instance,
        """
```

Submission of experiments.py

- 60 seconds per run of the metaheuristic.
- Metric: best fitness, in case of a tie, computation time.
- Each configuration tested should be tested n times.
- Use the same populations across different configurations
- Obtained results will be attached and summarized as part of the project report.
- The experiments' code should be executable and replicable.
- To optimize the hyperparameters and configuration of the genetic algorithm you are encouraged to use grid search, random search, or bayesian optimization. Code should be provided for those.

Tournament baselines: Random Search

```
While time is available:
  Select k random assets to be employed
  Generate k uniform numbers between 0 and 1, normalize to sum exac
tly 1
  Assign normalized weights to the k assets, others are 0.
  Check if generated solution improves the current best solution.
```

Tournament baselines: Greedy Randomized Heuristic

```
For each asset i:
  Compute the average difference D_i to other assets.
  Normalize D_i values to a 0-1 scale.
  Normalize r_i values to a 0-1 scale.
  Compute r_i * D_i
While time is available:
  Select the k assets using fitness proportional selection according to the
 previous metric
  Generate k uniform numbers between 0 and 1, normalize to sum exactly 1.
  Assign normalized weights to the k assets, others are 0.
  Check if generated solution improves the current best solution.
```

Tournament baselines: GA I

- A genetic algorithm that:
  - Represents a solution as a list of n real values between 0 and 1. The decoding of a solution consists of selecting the k assets with the highest weight, and normalizing weights to sum 1.
  - Generates an initial random population of 100 individuals.
  - Employs binary tournament selection to select 30% of the population as parents.
  - Applies one-point crossover with a probability of 80% to selected parents. Otherwise, parents are copied.
  - Children are mutated with a 20% probability using swap mutation.
  - The top individuals from the population and generated children are used as the next generation.

Tournament comparison

Algorithm	Test instance	Avg. fitness	Avg. time
alg1	test01	12.45	167.5
alg1	test02	562.12	179.87
alg1	test03	2444.11	179.23
alg1	test04	850.34	120.24
alg1	test05	40.15	112.3
alg2	test01	24.56	180.0
alg2	test02	456.24	175.16
alg2	test03	3200.1	169.83
alg2	test04	700.12	170.24
alg2	test05	35.23	115.67
alg3	test01	34.45	162.9
alg3	test02	500.88	177.17
alg3	test03	4000.25	179.13
alg3	test04	400.15	128.24
alg3	test05	67.6	146.23

Test instance	Rank alg1	Rank alg2	Rank alg3
test01	1	2	3
test02	2	1	3
test03	1	2	3
test04	3	1	2
test05	2	1	3

Algorithm	Average rank
alg2	1.4
alg1	1.8
alg3	2.8

Report

- 6 pages maximum using IEEE template
  - Abstract
  - Introduction
  - Design of submitted GA
  - Experiments
  - Conclusions
  - References

Assessment

Aspect	<50%	50-69%	70-89%	90-100%
Style, organization and clarity (1 mark)	The document is not organized in the proposed sections There is no coherence in ideas and explanations References are not present.	There is a certain coherence in ideas and explanations Missing sections or inadequate organization	There is a certain coherence in ideas and explanation No missing sections Adequate use of figures and tables	Ideas and explanations are coherent No missing sections Adequate use of figures and tables
Metaheuristic design and implementation (2 marks)	Non working implementation no merit or plagiarized	Basic design Basic description	Basic design Detailed description	Original design Detailed description Documented code
Tournament (5 marks)	Improves baselines	Q4 ranking	Q3 and Q2 ranking	Q1 ranking
Experiments (2 marks)	No experiments, no merit, or non-functional	Provided functional code for experiments	Provided functional code for experiments Experimental results explained in report	Provided functional code for experiments Experimental results explained in report Correct methodology for analysis

Thank you!

Víctor Sánchez Anguix - vicsana1@upv.es