Metaheuristics Project

The project is part of the *Metaheuristics* lecture. Upload your solution to LernRaumPlus containing: (1) **a typeset PDF** explaining your solution (see below) and a **.zip** containing the Python code to run your program. Do not include any instances or data. The submission is due by **Sunday (07.07.2024) 23:55**. Late submissions will be penalized by 1% of total points per minute.

Groups: You may work in groups with **exactly four students**. Exceptions will be made on a case-by-case basis to ensure everyone in the class is in a group, and only with prior permission. Furthermore, the amount of work on the project is proportionally increased or decreased depending on the group size. We reserve the right to change groups to balance skill sets between groups.

Submission document: The document you hand in should contain at least the following components:

1. Approach description
2. Citations and discussion of relevant work in the literature
3. Experimental investigation of your approach's components (**!!!**)
4. Experimental investigation of the approach's overall performance (**!!!**)

Please keep the document short, concise and to the point. **Do not put any actual code into the document!** Doing so will result in 0 points for the document. Pseudocode is, however, allowed and encouraged. Note that the experimental investigation should include analysis and will not receive full points unless analysis is included. You can use tables, plots, etc. to present the performance of your technique.

Code re-use policy: Standard libraries/modules as well as libraries/code snippets from the internet may be used as long as they fulfill all of the following conditions:

1. They are credited in the submission document, as well as in the submitted code,
2. They do not solve the given task directly (i.e., a solver accepting a model is OK, but using the Paradiseo package for your heuristic is not)
3. They do not carry out any metaheuristic search strategy (i.e., you must write the metaheuristic yourself),
4. They do not completely encompass all operations of a single heuristic.
5. The code license permits its use in 3rd party works.
6. The use of large language models is permitted, but is subject to this citation policy. Large language models may not be used to code the entire project.

Cheating policy: This project is a group project and as such there may be no sharing of code between groups (not even showing another group code!!!). Discussion of high level concepts is, however, allowed and encouraged. Example: Group 1 to group 2: "We are using a genetic algorithm in which the solution representation is x,y,z." is OK.

Grading Projects will be evaluated based on the following criteria:

1. Quality of the initial solution heuristic and its solutions (if applicable)
2. Quality and performance of the neighborhood heuristics (if applicable)
3. Code quality (in terms of the metaheuristic implementation)
4. Document quality (in terms of the metaheuristic implementation)
5. Insight and analysis of description document and presentation
6. Final solution quality

1 Sports Scheduling

The DOT-Liga is one of the most well-known sports leagues in the world, and until the upcoming season they have planned their game schedules by hand. Their long time employee who did the planning is now retiring and they have hired you to help them create a schedule for their sports league. The league contains n teams that compete over $w = 2(n-1)$ weeks. Each game consists of two teams playing against each other. There are three time slots per week in which games are played: Monday, Friday, and Saturday (M/F/S). The problem consists of the following constraints:

1. Every team plays every other team twice; once at their home stadium and once at the other team's stadium.

2. Every team plays every week.

3. Each combination of time slot and pair of teams is associated with a profit p_{kij}, $k \in \{M, F, S\}, 1 \leq i, j \leq n$ due to TV revenue.

4. Only one game should be played in the Monday time slot, whereas up to t percent of the remaining games may be played on Friday or Saturday. Should multiple games be assigned to Monday, only the profit for the game with the lowest profit will be earned. All other profit will be lost (i.e., it is set to 0).

5. Every team should play in the Monday time slot at least once.

6. There should be at least r weeks between when two teams play each other. If they play against each other earlier by q weeks, where $q < r$, the profit is reduced. Assume that p' is the full profit, then $p_{kij} = \begin{cases} p'_{kij} & q = 0 \\ \frac{p'_{kij}}{1+q^2} & 1 \leq q < r \end{cases}$ for $k \in \{M, F, S\}, 1 \leq i, j \leq n$.

Your task is to design a metaheuristic to compute a feasible schedule for the DOT-Liga that respects all of the constraints listed above and maximizes the total profit from the games played.
We define a metaheuristic as in the slides in which one or more heuristics are applied through a *high level search procedure.* Exact approaches/solvers may be used as part of your metaheuristic (e.g., fix-and-resolve LNS), however they may not be used to solve the entire problem.

1.1 Example

Consider an instance with the following parameters:

- $n = 6$ $(w = 10)$
- $t = 2/3$
- $s = 4$
- $r = 1$

The profit matrix is the following. Note that it is not symmetric.

		Away Team (j)																	
		Monday						Friday						Saturday					
		1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6
Home Team (i)	1	-	15	20	16	28	9	-	8	10	9	5	9	-	2	3	5	3	5
	2	13	-	8	12	10	12	6	-	5	11	7	4	1	-	3	2	9	10
	3	19	10	-	14	15	13	8	4	-	12	8	6	7	8	-	4	5	2
	4	18	14	12	-	13	16	10	7	4	-	4	2	6	7	5	-	2	1
	5	25	15	13	17	-	11	10	7	5	3	-	10	6	8	5	4	-	7
	6	10	14	23	21	10	-	4	8	2	7	1	-	8	3	2	7	4	-

An example feasible schedule is the following:

	Game Times				
Week	M	F_1	F_2	S_1	S_2
1	5 vs. 1	3 vs. 6	2 vs. 4	-	-
2	6 vs. 2	1 vs. 4	3 vs. 5	-	-
3	5 vs. 4	1 vs. 6	-	3 vs. 2	-
4	6 vs. 4	1 vs. 3	-	5 vs. 2	-
5	1 vs. 2	3 vs. 4	5 vs. 6	-	-
6	6 vs. 3	4 vs. 1	-	2 vs. 5	-
7	3 vs. 1	4 vs. 2	-	6 vs. 5	-
8	1 vs. 5	-	-	4 vs. 3	2 vs. 6
9	4 vs. 5	2 vs. 3	-	6 vs. 1	-
10	4 vs. 6	5 vs. 3	2 vs. 1	-	-

The objective function value is 351.

1.2 Technical Requirements

1.2.1 Command line

Your program must be written in the programming language Python 3 and the code must be executed with the command line:

```
python3 sports.py ⟨timeout (s)⟩ ⟨path to instance⟩
```

The first parameter provides the time limit and the second parameter provides the path to the instance. Note that the angle brackets are not given on the command line, i.e., an example execution of your program is:

```
python3 sports.py 30 data/example.in
```

The above command line executes the metaheuristic for 30 seconds on the instance `example.in`, which is available in the folder `data`. It is essential that your program works on the command line. If it does not, it fails. Our advice is to use the `argparse` module offered by Python, and to code your program from the beginning using the command line, rather than hard coding information.

1.2.2 Instance file (Input)

```
n: <integer>
t: <float in [0,1]>
s: <integer>
r: <integer>
p: <list of integers>
```

The format of p is as follows. First the $n \times n$ matrix for day M is given in a flattened format in which the rows are appended to each other sequentially. Then comes day F and day S in the same format. The entries (k, i, j) with $i = j$ in p are given as -1. We provide code for reading instances that you can use. Note that no questions regarding instance reading will be answered, other than questions regarding any bugs in the instance reader we provide.

1.2.3 Output

There are two different cases for the output of your solution procedure. Your output must conform to the following specification or it will be marked as incorrect, and you will receive 0 points for the implementation without any exception.

Case 1: Maximum runtime is reached without finding a solution:

```
### RESULT: Timeout
```

Note that having lots of timeouts will lead to a bad grade, as the instances we provide will always have at least one feasible solution.

Case 2: A solution exists:

```
### RESULT: Feasible
### OBJECTIVE: <Objective Value>
### Game <week>-M: <home> <away>
### Game <week>-F: <home> <away>
### Game <week>-S: <home> <away>
### CPU-TIME: <cpu-time (in seconds)>
```

In the output format, when there are multiple games on Friday or Saturday, simply include multiple lines. Some output for the example instance is provided below. Due to the length of the output, we only provide the first two weeks.

```
### RESULT: Feasible.
### OBJECTIVE: 311
### Game 1-M: 1 5
### Game 1-F: 2 4
### Game 1-F: 3 6
### Game 2-M: 6 4
### Game 2-F: 3 1
### Game 2-S: 2 5
...
### CPU-TIME: 12.00
```