MODA - Assignment 2

Hao Wang, LIACS, Leiden University

May 2024

The second assignment consists of a mini-team project in which you will solve a multi-objective problem with your teammate and present it in the class.

- We ask you to **form a group of three**. We may allow for a group of four if three is not a divisor of the total number of students.
- You will need to pick a multi-objective problem to solve, which comprises (1) problem formulation, (2) solving it mathematically or with any multi-objective algorithms, and (3) reporting the performance and anything you learned in the project.
- The only materials you need to submit are (1) a **four-slide presentation** and (2) **the source code** you developed for it and **performance data**, e.g., the Pareto front approximation you get. We will print it in A2 as a poster.
- **In the last lecture, on the 23rd of May**, we will organize a **poster session** to mimic a mini-conference, where you will stand by your poster and explain your project to your peers and me. We will announce, via Brightspace, the details regarding the organization of the poster session.

You don't necessarily need to implement the algorithms yourself (of course, it is also fine if you wish to implement the algorithms we discussed in the lecture). You are encouraged to use one of the following algorithm libraries:

- `DESDEO` (`https://desdeo.readthedocs.io/en/latest/`): a interactive multi-objective optimization software.
- `pymoo` (`https://pymoo.org/index.html`): a commonly-used multi-objective optimization algorithm library, written in `Python`.
- `pygmo` (`https://esa.github.io/pygmo2/`): in case you want faster implementations - written in `C++` with `Python` interfaces.

Mini Research Project: Solving Multi-objective Optimization Problems in Practice

When you work as a practitioner in operational research optimization, you often need to develop a problem-specific algorithm supported by an optimization library, such as `DESDEO`, and demonstrate its capability to solve the problem, e.g., finding an approximation of the Pareto front. *You can choose one of four case projects (Case 1-4, described below) or create your own project that should be similar to the example case projects provided below.* Please collect the empirical performance data from solving the chosen problem to demonstrate that the algorithm works.
You are also allowed to change or add assumptions in the given case projects if you report this. For example, in case 1, you could think of a network with 20 places, of which 10 need to be visited on a tour, and the time windows should be important for deciding which tour to follow.

Submission instruction for the slides presentation

We ask you to send us the slides summarizing the problem, approach, and results, which you will present during a session in the last lecture (May 23rd).
In detail, for the poster,

***Submission (please read carefully)**: Please prepare four slides in PDF format in landscape orientation and upload them to Brightspace before **19th of May**, since we need at least two days to print them out. Please use the following naming scheme for your files:*

`MODA24A2_team_number.pdf`

We will print these 4 slides as an A2 poster:

- Slide 1 should describe the problem definition and example data;
- Slide 2 should describe the algorithmic approach;
- Slide 3 should describe the results (e.g., a Pareto front, for example, some solutions of the efficient set);
- Slide 4 should contain a brief discussion and summary.

In addition to the 4 slides, you are also asked to submit Python source code and data sets (if any). The lecturer/teaching assistants and other peer students will evaluate all posters. The grading criterion will be communicated via Brightspace.

Case 1: Green Delivery Routing

Consider the problem of green delivery routing.

1. A route is given by a sequence of places to be visited, say place $1, 2, 3, ..., n$.
2. A schedule can be represented by a sequence in which these n places are visited, say x_1, x_2, ..., x_n, with $x_i \in \{1, ..., n\}$, $i = 1, ..., n$.
3. For each place, there are two-time windows for delivery. The preferred time window $W_i^0 = (l_i^0, u_i^0)$ and the less preferred time window $W_i^1 = (l_i^1, u_i^1)$.
4. A binary decision variable decides for each place which time interval should be chosen.
5. The processing (or dwelling) time in place i is given by p_i (i.e., the time the actual delivery process at the client's address takes). The time to travel from place i to place j is given by $d_{i,j}$.
6. the driver needs to deliver at the earliest possible time within the chosen time window and stay within the total delivery time.
7. the actual times when the delivery at the place of the i-th visited client starts can be computed recursively, using: $T_0 = -\infty$ and $T_i = \max\{l_{x_i}^{b[x_i]}, T_{i-1} + d_{x_{i-1},x_i} + p_{x_{i-1}}\}$, $i = 1, ..., n$.
8. It must be secured that $T_i \in W_i^{b[i]}$ and $T_i + p_{x_i} \in W_i^{b[i]}$ (delivery process happens in the specified time windows).
9. From the last client the delivery driver needs to drive back to the depot, which is a place with an index 0. Hence, the total length of the tour is given by $T_n + p_{b[i]} + d_{x_i,0}$. This tour length should be minimized.
10. The number of non-preferred time intervals should be minimized.

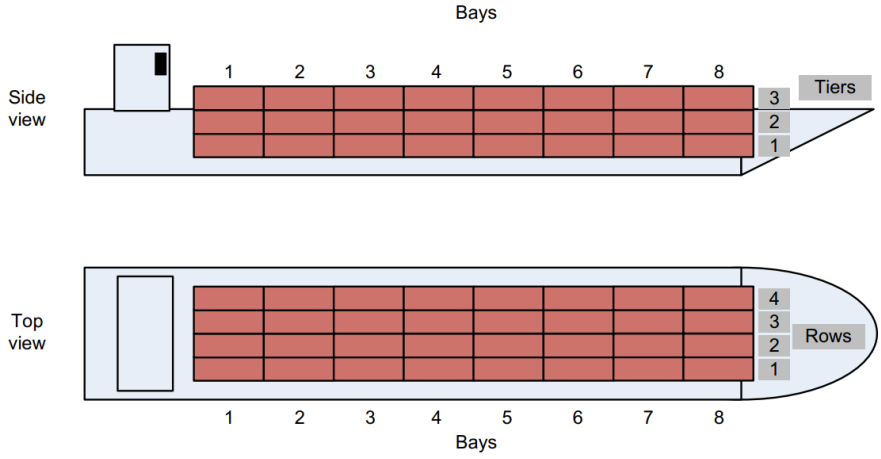
Case 2: Cargo Ship

The ship MS Leiden is traveling on a route involving five harbors in northern Europe: Rotterdam, Hamburg, Kiel, Aarhus, and Copenhagen. You have been asked to help the Captain with the loading and unloading plan (for the destination harbors) that produces

- even and well-balanced solution
- a solution that is easy to unload (so that containers that are earlier unloaded are not beneath containers that are later unloaded)
- solutions that can integrate as many containers as possible

The following details about the problem are provided:

1. The ship is now docked in Rotterdam. After visiting Rotterdam, it will travel to Hamburg, Kiel, and Aarhus and terminate in Copenhagen. The containers destined for Rotterdam have been unloaded, and the containers destined for the remaining four harbors must be loaded onto the ship.
2. MS Leiden is a rather small container ship. It has eight bays, three tiers, and four rows. The layout of the ship is shown in the figure below.
3. Containers should be loaded such that a container that has to be unloaded earlier should be placed in a higher position.
4. Each cell in the above figure is able to hold a forty-foot container. That is, the ship has room for $8 \times 3 \times 4 = 96$ forty-foot containers.
5. A forty-foot container can be placed on top of another forty-foot container but not on top of an empty cell. We assume that only forty-foot containers are loaded on the ship. Each container has a destination port.
6. Each container i has a certain weight w_i. If a container is much heavier than another i container j, say $w_i - w_j > \delta_w$, then it is not allowed to place w_i on top of w_j.
7. The containers should be balanced in a way that is evenly distributed. The longitudinal center of gravity should be as close as possible to the middle of the container section of the ship. Secondly, the latitudinal center of gravity should be as close to the middle as possible. Note that the center of gravity can be computed as the weighted sum of the centroids of the containers, where the weights are the total weight of the containers.



Case 3: Bicycle Routing

Consider a network of connections between bicycle vertices ('fietsknooppunten').

1. a distance matrix d_{ij} is given, which is the total distance between the two knots in meters.

2. Here $a_{ij} \in \{1, 2, ..., 5\}$ is a vector for scenic beauty, where 1 is the worst and 5 is the best score. $b_{ij} \in \{1, 2, ..., 5\}$ is a vector about roughness, where 1 stands for very rough path, and 5 for a smooth bicycle path, $s_{ij} \in \{1, 2, ..., 5\}$ is a variable that signifies the safety of a connection, where 1 stands for very safe, and 5 for quite dangerous traffic situation. Finally, $l_{ij} \in \{1, 2, ..., 5\}$ stands for the slope. If the connection is a gentle descent or even if the score is 1. If it is very steep, the score is 5.

Program a route planner that can plan a route that includes certain nodes, minimizes total distance, and maximizes comfort. You can allow the user to model comfort as constraints, objectives (e.g., average comfort), or both, e.g., by enforcing a minimum level for each comfort criterion using a metric penalty approach. One possibility is to aggregate the comfort objectives or add constraints on them. A permutation can represent a route. The first node is fixed as the starting node, and the route is finished when all target nodes have been reached. Demonstrate the algorithm for a small data set, say with 15-20 nodes in the network and routes with, say 7-10 nodes.

Case 4: Designing Tent Roofs using the Convex Hull Representation

In the lecture, we discussed the problem of designing roofs using the convex hull representations. The challenge in Case 4 is to come up with roof geometries above the base geometry of a half-open box. You can use continuous ranges for the coordinates of the points. Try different objectives and combinations of them, such as height, floor area, volume, and surface area. A good number of points is 8 to 12 points.