MODA - Assignment 2

Michael Emmerich, LIACS, Leiden University

April 2023

The second assignment consists of a mini-project. You are encouraged to use the DESDEO library for the solution, that we will explain in the lecture.

Mini Research Project: Solving Multi-objective Optimization Problems in Practice (10pt)

When you work as a practitioner in optimization of operational research, coming up with a problem specific metaheuristic supported by a optimization library, such as DESDEO, and demonstrating its capability to solve the problem, e.g. finding an approximation of the Pareto front, is a skill that is often needed. You can choose one out 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 make your own example data set for demonstrating 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 in a tour, and the time windows should be important for deciding which tour to follow.
You are invited to design your own customized mutation, initialization, and optionally, recombination operator.

Submission modalities for part 2

We ask you for sending us the PDF of a paper summarizing the problem, approach and results, and also a poster, which you will present during a session in the last lecture (May 24th). Attach also the data-sets and source code.

In detail, for the poster prepare 4 slides that you have to send us (submit via **brightspace on the 22nd of May, latest**) in PDF format in landscape orientation. We will print these 4 slides as 4 A3 pages and make a poster from it. 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 an example, some solutions of the efficient set), and 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 files with data-sets, and a PDF report. The PDF report should be limited to say 6-8 pages, and can in more detail describe the project and its results (as compared to the slides).

You will present the posters in a poster session to the other members of the course and/or the course instructors.

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 that 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

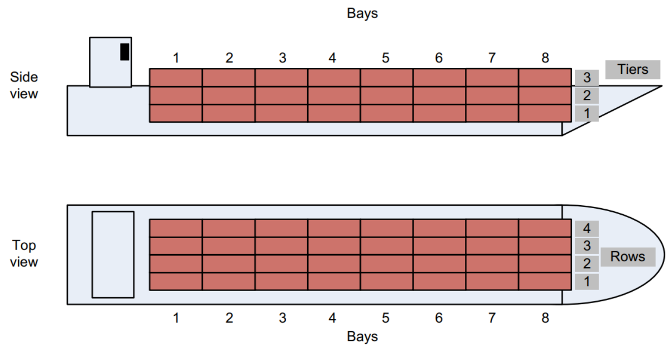
The ship MS Leiden is traveling on a route involving five harbors in northern Europe: Rotterdam, Hamburg, Kiel, Aarhus, and Copenhagen. You have been asked to help the Captain with the loading and unloading plan (for the destination harbors) that produces

- 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, the ship will be traveling to Hamburg, Aarhus, and Copenhagen, in that order. The containers destined for Rotterdam have been unloaded and the containers destined for the remaining three harbors have to be loaded on 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 much 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, $s_{ij}\{1, 2, ..., 5\}$ is a variable that signifies the safety of a connection, where 1 stands for very safe, and 5 for quite dangerous traffic situation. Finally, $l_{ij}\{1, 2, ..., 5\}$ stands for the slope. If the connection is a gentle descent or even 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 or as 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 route can be represented by a permutation. The first node is fixed as the starting node and then 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 and we have provided source code in python/desdeo for visualizing the roof geometries for tents of various objectives, and computing its objectives. The challenge in Case 4 is to come up with roof geometries above a base geometry of an 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.