# CSE5012: Evolutionary Computation and Its Applications
# Assignment 2: Single Objective Combinatorial Optimization: Delivery Scheduling Problem (DSP)

Han Zhang and Liyan Song

06 March 2023

## 1 Overview

The main task of this assignment is to apply Evolutionary Algorithm (EA) to optimize a single objective combinatorial optimization problem: delivery scheduling problem (DSP), which is part of the space optimization competition hosted in GECCO 2022 (`https://www.esa.int/gsp/ACT/projects/gecco-2022-competition/`). Please fetch the assignment project from `https://github.com/SUSTech-EC2023/Assignment2`. This assignment has 100 marks, consisting of 20% for the programming, 30% for the report in Chinese and 50% for the in-class presentation with slides in English. In the end, this assignment will take up 20% of your final score of this course.

## 2 DSP Task

You should design and implement your own EA to solve the DSP. The goal is to assign 340 asteroids to twelve stations as efficiently as possible. More details of the DSP can be found in `https://optimize.esa.int/challenge/spoc-delivery-scheduling/About`.

**Programming language** Python

**Problem statement** Asteroids from the Trappist belt contain large amounts of 3 particular materials (denoted as A, B and C). To collect and process these materials, 12 stations have been deployed in the belt. Each station can only become active in **one** time interval, which is called the ***activity window***, and a station can only receive materials during their activity windows. In practice, only one station can be in its active window **at a time**. For each asteroid, a finite number of possible transfer opportunities to different stations are identified. Given an asteroid and station pair, the amount of material delivered may change depending on the transfer opportunity chosen.

Your task is to schedule the delivery of the asteroids to the stations (i.e., choose the transfer to perform out of the possible ones for all 340 asteroids). Your objective is to maximize the minimum mass collected among all stations and all three materials. You have only **80 days** to deliver as many asteroids as possible. In a word, there are 80 days to deliver three materials (A, B and C) from 340 asteroids to 12 stations. To collect as many materials as possible, you should give a solution to make the following decisions under the below constraints:

- The activity window of each station. Note that only one station can be active at a time, and each station can only become active once.
- The delivery (assignment) of 340 asteroids to 12 stations (i.e., determine the sole station for each asteroid).

**Data** You are provided with a database, which can be downloaded from the GitHub Project of Assignment 2 or from `https://api.optimize.esa.int/data/spoc/scheduling/candidates.txt`), consisting of a list of asteroids and a list of delivery opportunities per asteroid to a subset of the stations.

To use the evaluation code provided with the problem, the database must be named as **candidates.txt** and located in the relative path **.\data\spoc\scheduling\candidates.txt** with respect to your work directory.

Each asteroid is identified by an integer, denoted as **asteroid_id** namely {1,...,340}. Similarly, each station is identified by an integer, denoted as **station_id** namely {1,...,12}. Note that the first station is indexed by 1, rather than 0.

Given an asteroid-station pair **{asteroid_id, station_id}**, delivery opportunities can be represented as a 4-tuple $[T_{arr}, m_1, m_2, m_3]$, where $T_{arr}$ is the arrival epoch (for the concept of epoch, please refer to `https://en.wikipedia.org/wiki/Epoch_(astronomy)`) at the station and $m_1, m_2$ and $m_3$ denote the delivered masses of materials A, B and C, respectively. Each delivery opportunity is identified by an integer **opportunity_id** ranging in [1, 8]. Note that the first opportunity is indexed by 1, rather than 0. A delivery opportunity is therefore completely identified by a 3-tuple **[asteroid_id, station_id, opportunity_id]**.

Conceptually, the database has the following structure:

```
{
    "Asteroid ID":{
        "Station ID": [["Arrival time", "Mass1", "Mass2", "Mass3"], ["
            Arrival time", "Mass1", "Mass2", "Mass3"], ... ],
        "Station ID": [["Arrival time", "Mass1", "Mass2", "Mass3"], ["
            Arrival time", "Mass1", "Mass2", "Mass3"], ... ],
    },
    "Asteroid ID":{
        "Station ID": [["Arrival time", "Mass1", "Mass2", "Mass3"], ["
            Arrival time", "Mass1", "Mass2", "Mass3"], ... ],
        "Station ID": [["Arrival time", "Mass1", "Mass2", "Mass3"], ["
            Arrival time", "Mass1", "Mass2", "Mass3"], ... ],
    },
}
```

Example:

```
{
    "42": {
        "5": [[17.34, 0.012923, 0.045365, 0.093846], [38.77, 0.010074,
            0.039874, 0.083761]]
    }
}
```

This entry describes two opportunities for transferring Asteroid 42 to Station 5, with respective arrivals epochs of 17.34 and 38.77 days, followed by the delivered masses for materials A, B and C, respectively. The opportunity assigned to the arrival time 17.34 appears first and is assigned an identification **opportunity_id** equal to 1. The corresponding assignment is therefore identified by the **[asteroid_id, station_id, opportunity_id]** 3-tuple [42, 5, 1]. The opportunity assigned to the arrival time 38.77 appears second and is assigned an identification **opportunity_id** equal to 2. The corresponding assignment is therefore identified by the **[asteroid_id, station_id, opportunity_id]** 3-tuple [42, 5, 2].

**Encoding of solution** The decision vector **x** consists of two parts:

- Initial and final epochs defining the station activity windows: **[T_1i, T_1f, ..., T_12i, T_12f]**, where **T_ki** and **T_kf** are the initial and final epochs defining Station k's activity window **[T_ki, T_kf]**. All epochs are float variables in [0, 80]. This first part must contain 24 entries (twice the number of stations). Note that epochs are ordered according to the **station_id** identification and therefore not necessarily in chronological order.
- Asteroid assignments to stations: **[asteroid_id_k, station_id_k, opportunity_id_k]**, for $k = 1, ..., 340$

The asteroid assignments correspond to 3-tuples **[asteroid_id, station_id, opportunity_id]** of integer variables defining:

- The identification number **asteroid_id** of the asteroid being allocated in [1, 340]
- The identification number **station_id** to which the asteroid **asteroid_id** is being allocated, in [0, 12]. **Unassigned asteroids that are not delivered to any station shall be associated to a station_id equal to 0**.

- The identification number **opportunity_id** indicates which transfer opportunity from the dataset is used for the **asteroid_id** to **station_id** assignment.

  Not all **{asteroid_id, station_id}** pairs have the same number of transfer opportunities available, and there are at most 8 transfer opportunities per pair. Note that **opportunity_id** must correspond to an existing allocation, otherwise the assignment will be flagged as infeasible. Unassigned asteroids, associated to a **station_id** equal to 0, can be assigned any **opportunity_id** in [1, 8] and will deliver no mass.

The full decision vector is the concatenation of all entries: **[T_1i, T_1f, ..., T_12i, T_12f, asteroid_id_1, station_id_1, opportunity_id_1, ..., asteroid_id_340, station_id_340, opportunity_id_340]**

**The decision vector x must contain entries for all asteroids in the database, making it fixed length (1044 entries)**.Unassigned asteroids that are not delivered to any station shall be associated to a **station_id** equal to 0.

**Constraints**

- Each processing station can only become active for one time interval, given as its activity window $[T_{ki}, T_{kf}]$ for $k = 1, ..., 12$. All deliveries must occur within 80 days, and stations cannot remain open to new asteroids past that deadline. Enforced via the problem box bounds as returned by *udp.get_bounds()* (please refer to Section 2 Utilities/Hints for details of *udp*). Mathematically, this constraint corresponds to:

$$0 \leq T_{ki} \leq T_{kf} \leq 80 \tag{1}$$

- Only one processing station can be in its active window at a time. Stations can only receive materials during their active windows. The sequence defining the order in which stations welcome new asteroids is not constrained and can be any permutation of 1,2,3,4,5,6,7,8,9,10,11,12. Formally:

$$[T_{ji}, T_{jf}] \cap [T_{ki}, T_{kf}] = \varnothing \; for \; j, k = 1, ..., 12 \; and \; j \neq k \tag{2}$$

- Each asteroid can only be assigned once, and assignments must be consistent with the opportunities provided in the candidate database.

  Implemented as two equality constraints, satisfied when zero:

  - 1) eq_constraint_1 (= *udp.fitness(x)*[1]) represents the number of missing or wrongly indexed asteroids in the decision vector **x**.
  - 2) eq_constraint_2 (= *udp.fitness(x)*[2]) represents the number of asteroid assignment violations in the decision vector **x**, i.e. the number of [asteroid_id, station_id, opportunity_id] 3-tuples with invalid opportunity_id indices not consistent with the database.

- There must be at least one day separating all active windows. In other words, the time interval between the arrival of the last asteroid at a station and the first asteroid at the next station cannot be shorter than one day.

  Let us consider any two consecutive station activations $j$ and $k$, so that station $j$ is activated right before station $k$. The time interval constraint mathematically translates to:

$$T_{ki} - T_{jf} > 1 \tag{3}$$

  Implemented as an inequality constraint: ineq_constraint_1 (= *udp.fitness(x)*[3]) returns one minus the minimum time gap between stations. Satisfied when zero or smaller.

- Asteroids delivered to a given station must have arrival times included in the station's activity window. Once a station stops welcoming new asteroids and another station opens its doors, no more asteroids can be delivered to it for the remainder of the 80 days time span.

  Let us denote, for a given station identified by the station_id $k$ with $k \in [1, 12]$, $\tau_k$ the set of arrival times of asteroids delivered to that station. Then, this constraint is mathematically equivalent to:

$$T_{ki} \leq T_{arr} \leq T_{kf} \;\; \forall T_{arr} \in \tau_k, \; for \; k \in [1, 12] \tag{4}$$

  Implemented as an inequality constraint ineq_constraint_2 (= *udp.fitness(x)*[4]) represents the number of violations of this constraint among the allocated asteroids. Satisfied when zero.

**Objective** The objective of this challenge is to maximize the minimum mass collected among all processing stations and among all three materials $M_{min}$. Let $M_{A,i}, M_{B,i}$, and $M_{C,i}$ denote the total masses of materials A, B and C accumulated at the $i$-th station after all its assigned asteroids have been delivered. $M_{min}$ is defined as:

$$min_{i=1,\dots,12} \left( min_{j \in \{A,B,C\}} M_{j,i} \right) \tag{5}$$

For any decision vector $\mathbf{x}$, the fitness and constraint evaluations can be retrieved with the following line of code:

```
M_min, eq_constraint_1, eq_constraint_2, ineq_constraint_1,
    ineq_constraint_2 = upd.fitness(x)
```

**Submitting to competition official website** To submit a solution, you can prepare a submission file with the submission_helper.py (can be downloaded from Assignment2 GitHub Project or `https://api.optimize.esa.int/data/tools/submission_helper.py`) via

```
from submisson_helper import create_submission
create_submission("spoc-delivery-scheduling","delivery-scheduling",x,"
    submission_file.json","submission_name","submission_description")
```

and submit it in `https://optimize.esa.int/submit`.

**Utilities / Hints**

- This competition is purely combinatorial, and no astrodynamics computations of any kind are required for its resolution.
- Have a detailed look at the User Defined Problem (UDP) accessible under Evaluation code to understand how the evaluation works. The evaluation code file can be downloaded from `https://api.optimize.esa.int/media/problems/spoc-delivery-scheduling-delivery-scheduling-1677586214648.py` or Assignment2 GitHub Project.
- To decode the information from a decision vector $\mathbf{x}$ into a summary printed on screen, call *udp.pretty(x)*.
- Plots for the total material masses and the schedule vs. opportunities per station can be obtained by calling *udp.plot(x)*.
- An example chromosome is provided in the source code and can be evaluated by calling *udp.example()*. Please note that this solution corresponds to a minimal working example and should not be taken as a baseline for what the winning score should be.
- Note that not all **{asteroid_id, station_id}** pairs have the same number of transfer opportunities available. There are at most 8 transfer opportunities for a given couple **{asteroid_id, station_id}** in the database, but it can and will often be less.
- The method *convert_to_chromosome()* available in the evaluation code allows you to create a valid decision vector $\mathbf{x}$ entry from an incomplete decision vector where not all asteroids entries are present. This can be particularly useful if you do not deliver all asteroids in your solution and want to auto-complete the decision vector with the unassigned asteroids. Note that the method will not perform any constraint checks on the station activity windows and on the already assigned asteroids of the incomplete chromosome. However, the method does expect all 12 station windows (the first 24 entries) to be present in the decision vector.
- If you have a valid selection of [**asteroid_id, station_id, opportunity_id**] 3-tuples meeting the problem constraints, then you can always define station activity windows to be the time interval between the first and last asteroid arrival at any given station.

**Experimental requirements** You need to test your proposed EA with at least **10 runs** to gain the mean performance.

**Project** A Python project is provided in GitHub `https://github.com/SUSTech-EC2023/Assignment2`.

- The fold "*data*" contains the database. You shouldn't edit it.
- "*candidates.txt*" in the fold "*data/spoc/scheduling*" contains a list of asteroids as well as, for each of these asteroids, a list of delivery opportunities to a subset of the stations. You shouldn't edit it.
- "*submission_helper.py*" receives your decision vector $\mathbf{x}$ as one parameter and generates a submission file that can be submitted to the competition's official website. You shouldn't edit it.
- "*spoc_delivery_scheduling_evaluate_code.py*" is the evaluation code. You shouldn't edit it.
- "*testEA.py*" is used to test your EA. You shouldn't edit it. We will use this file to test your program when marking your assignment.
- "*EA.py*" is where you should implement your EA. Normally, this is the only file that you need to submit.

**Evaluation of program**

1. The submitted "*EA.py*" will be tested.
2. Any program that fail running will receive 0 mark.
3. Any behavior that modifies the *candidates.txt*, *submission_helper.py*, or *spoc_delivery_scheduling_evaluate_code.py* receives 0 mark.

**Remarks**

- You are not encouraged to add new files or folders. "*EA.py*" is the only code file that you will submit. Feel free to define multiple functions inside the "*EA.py*" file.

# 3 Program (20% Marks)

You need to submit a Python file entitled "*EA.py*".

**Remarks**

1. Any program that is failed to run receives 0 mark!
2. Normalization, neatness and readability of your program also contribute to a higher score.
3. Your program score would depend on the algorithm evaluation explained below.

**Algorithm evaluation** Performance score of your program will be measured based on the optimization performance of the problem. First, a fitness score (FS) will be calculated based on the final decision vector found by your EA.

$$FS = M_{min} + eq\_constraint\_1 + eq\_constraint\_2 + min(0, ineq\_constraint\_1) + ineq\_constraint\_2 \quad (6)$$

All variables in Eq.(6) can be obtained by the following program statement:

```
M_min, eq_constraint_1, eq_constraint_2, ineq_constraint_1,
    ineq_constraint_2 = upd.fitness(x)
```

Then, your decision vector will be compared to the best result on the competition's leaderboard (`https://optimize.esa.int/challenge/spoc-delivery-scheduling/p/delivery-scheduling`) by calculating a distance (D) between your $FS_{your}$ and the worldwide best score $FS_{best}$ submitted on 06 February 2023:

$$D = \frac{FS_{your}}{FS_{best}} \quad (7)$$

Your mark of program will be determined by the criteria listed in Table 1.

Table 1: Program evaluation criteria

| $D$ | Program mark (up to 20) |
|---|---|
| $D \geq 1$ | up to 20 |
| $0.9 \leq D < 1$ | up to 18 |
| $0.7 \leq D < 1$ | up to 16 |
| $D \leq 0.7$ | up to 14 |

Table 2: Fitness value

| Entry | Value |
|---|---|
| $FS$ | |
| $M_{min}$ | |
| $eq\_constraint\_1$ | |
| $eq\_constraint\_2$ | |
| $ineq\_constraint\_1$ | |
| $ineq\_constraint\_2$ | |

# 4 Report in Chinese (30% Marks)

A report in the PDF format must be submitted. MS Word and LaTeX templates can be found at `https://www.ieee.org/conferences/publishing/templates.html`. You should use these templates.

The report should include the below items:

1. Title, your name, your student ID, your affiliation, and your email
2. Abstract
3. Introduction
4. Background or related work
5. Your EA algorithm(s) / approach(es), including
   - EA operators adopted in your experiments,
   - pseudo-codes of your operators,
   - the reasons why you opt for those EA operators,
   - and so on.
6. Experimental setup
   - Datasets, number of runs, your platform and so on.
7. Experimental results and discussion. **Remarks:**
   - This section MUST contain a performance table (c.f. Table 2) of $FS$ of Eq. (6).
8. Conclusion and future works
9. References

**Remarks**

1. Please write in **Chinese**.
2. Please be careful of the grammar, spelling and format.

# 5 Oral Presentation (50% Marks)

Oral presentation in Chinese (or in English if you preferred) with slides in English, followed by Q&A panel (the duration will be informed later). Evaluation criteria include, but not limited to:

- Description of your objective functions. E.g., what ware they, their characteristics, why are they challenging, and so on.
- Your proposed / adopted EA operators and the reasons.
- Results and discussion.
- Presentation of the slides: format, typeset, spelling, grammar and so on.
- Language and clearness.

# 6 Submission

## 6.1 Important Dates

| Important date | What to do? | Where to submit? | Remark |
|---|---|---|---|
| 2023/04/14 (Friday) | First submission<br>1. First report<br>2. Program | BlackBoard | · 20 marks of first report<br>· 20 marks of program |
| 2023/04/16 (Sunday) | Slide submission | BlackBoard | - |
| 2023/04/17 (Monday) | Assignment viva | No submission | · 50 marks |
| 2023/04/21 (Firday) | Final report submission | BlackBoard | · 10 marks<br>· In the final report, the differences from your first report should be highlighted in blue. |

**Remarks** All deadlines are exactly 23:59 at Beijing Time.

## 6.2 What to Submit?

You must submit a first report, a final report, the slides, the program, and a performance excel:

**1) Report** Each student should submit one single file for report in the PDF format. The report MUST be entitled as

- **assignment2-report-{studentName}-{studentID}.pdf**.

Example: *assignment1-report-LiyanSong-12345678.pdf*.

**2) Final Report** Each student should submit one single file for final report in the PDF format. You are expected to update your report after getting the feedback on your first report and presentation. The differences from your first report should be highlighted in blue. The final report MUST be entitled as

- **assignment2-finalreport-{studentName}-{studentID}.pdf**.

Example: *assignment2-finalreport-LiyanSong-12345678.pdf*.

**3) Presentation** Each student should submit one single file for her/his presentation. The submitted file can be of any of the following formats:

- **assignment2-presentation-{studentName}-{studentnumber}.pdf**
- **assignment2-presentation-{studentName}-{studentnumber}.ppt**
- **assignment2-presentation-{studentName}-{studentnumber}.pptx**

Example: *assignment2-presentation-LiyanSong-12345678.pdf*.

**4) Program** Each student should submit one single file for the program. The submitted file can be of any of the following formats:

- **assignment2-program-{studentName}-{studentnumber}.rar**
- **assignment2-program-{studentName}-{studentnumber}.zip**

within which your program file that named "*EA.py*" should be contained.
Example: *assignment2-program-LiyanSong-12345678.zip*.

### 6.3 Where to Submit?

All assignments MUST be submitted to BlackBoard site, any other form of submission is NOT accepted.

## 7 Prohibitions

You will get 0-score for this assignment if any of the following cases happens:

- You submit more than one file for your program.
- You use other programming languages.
- You don't respect the naming policy of files.
- The report/program submission is delayed 3 days (included) or more (If the submission is delayed for 1 day, 25% discounts on the score of this part of assignment).
- Plagiarism.