TÉCNICO LISBOA

Artificial Intelligence and Decision Systems (IASD)
Mini-projects, 2017/2018

Assignment #1

Version 1.2 (15-Oct-2017)

In orbit assembly of large structures

1 Introduction

The International Space Station (ISS) is the most complex and the largest structure humans ever placed in orbit. Its assembly, in orbit, started in 1998 and has been continuously inhabited for more than 16 years. Its total mass is about 420 tons and consists of 32 modules (see figure 1).

The assembly of such a structure requires multiple rocket launches, since the payload of a single launch vehicle is limited. Moreover, launches are expensive and require extensive preparation. Therefore, the choice of which components to manifest in each launch, that is, to include in the payload of a launch vehicle, has to be carefully done.

This mini-project addresses the problem of scheduling the launch of components for the in orbit assembly of a large structure. Given a set of components to be launched, a launch timeline and costs, and a construction plan, the goal is to determine the assignment of components to launches, such as the total cost is minimized. In the next section the problem will be formally specified.

2 Problem statement

A construction plan is defined by a connected undirected acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is a set of vertices and $\mathcal{E}$ is a set of edges. Vertices represent components of the structure and an edge $(v_i, v_j) \in \mathcal{E}$ means that components v_i and v_j are directly connected in the structure. Since $\mathcal{G}$ is undirected, if $(v_i, v_j) \in \mathcal{E}$, then $(v_j, v_i) \in \mathcal{E}$.

Let $\mathcal{L} = \{l_1, \ldots, l_M\}$ be a set of launches. A launch $l_i \in \mathcal{L}$ is defined by a launch date t_i, a maximum payload p_i, a fixed cost f_i to be payed only if this launch will carry at least one component of the structure, and a variable cost u_i per unit of weight.

Consider that a given launch $l_i \in \mathcal{L}$ carries the set $S_i \subseteq \mathcal{V}$ of components into orbit (the manifest). First, the maximum payload of the launch has to

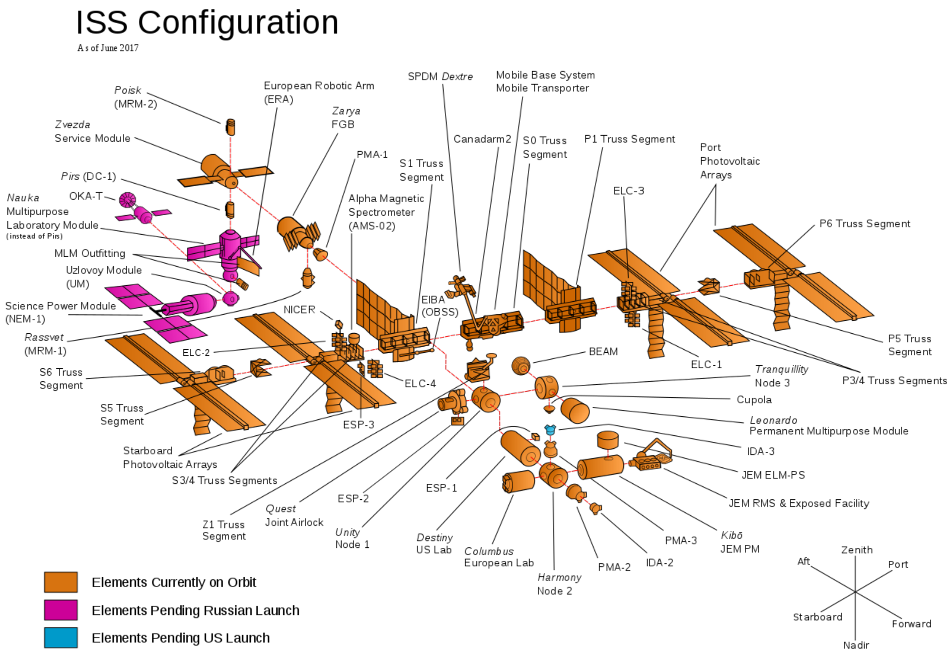


Figure 1: Exploded view of the International Space Station, as of June 2017.

be respected, that is,

$$\sum_{v_k \in S_i} w_k \leq p_i \tag{1}$$

where w_k is the weight of component $v_k \in \mathcal{V}$. And second, no component can remain unconnected in orbit, meaning that the subgraph induced[1] by the union between the components already in orbit and S_i must be connected. In other words, once all the components in S_i are deployed in orbit, all the components in orbit must be connected. The cost c_i of this launch is given by the following expression:

$$c_i = \begin{cases} 0, & \text{for } S_i = \{\} \\ f_i + u_i \sum_{v_k \in S_i} w_k, & \text{otherwise.} \end{cases} \tag{2}$$

The total cost for the assembly of the structure is then given by the sum of the costs of all launches, that is,

$$C = \sum_{l_i \in \mathcal{L}} c_i \tag{3}$$

The problem is then to determine the launch schedule, that is, the assignment of components to launches, such that all components are put into

[1]Given a set S, the induced subgraph from the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a graph $\mathcal{G}' = (S, \mathcal{E}')$ where $\mathcal{E}'$ are all edges $(a, b) \in \mathcal{E}$ such that both $a \in S$ and $b \in S$.

orbit exactly once, *i.e.*,

$$\bigcup_{l_i \in \mathcal{L}} S_i = \mathcal{V} \quad \text{and} \quad S_i \cap S_j = \{\} \text{ for } l_i, l_j \in \mathcal{L} \text{ and } i \neq j \tag{4}$$

the structure in orbit is always connected[2], and the cost C is minimized. Note that initially there is no component in orbit.

3 Implementation requirements

The problem is specified in a text data file where each line may specify a vertex, an edge, or a launch:

- a line starting with a 'V' specifies a vertex $v_i \in \mathcal{V}$ with the syntax:

  `V`*id weight*

  where `V`*id* is its unique name and *weight* its weight w_i (float);

- a line starting with a 'E' specifies an edge $(v_i, v_j) \in \mathcal{E}$ with the syntax:

  `E V`*id1* `V`*id2*

  connecting nodes `V`*id1* and `V`*id2*.

- a line starting with a 'L' specifies a launch $l_i \in \mathcal{L}$ with the syntax:

  `L` *date max_payload fixed_cost variable_cost*

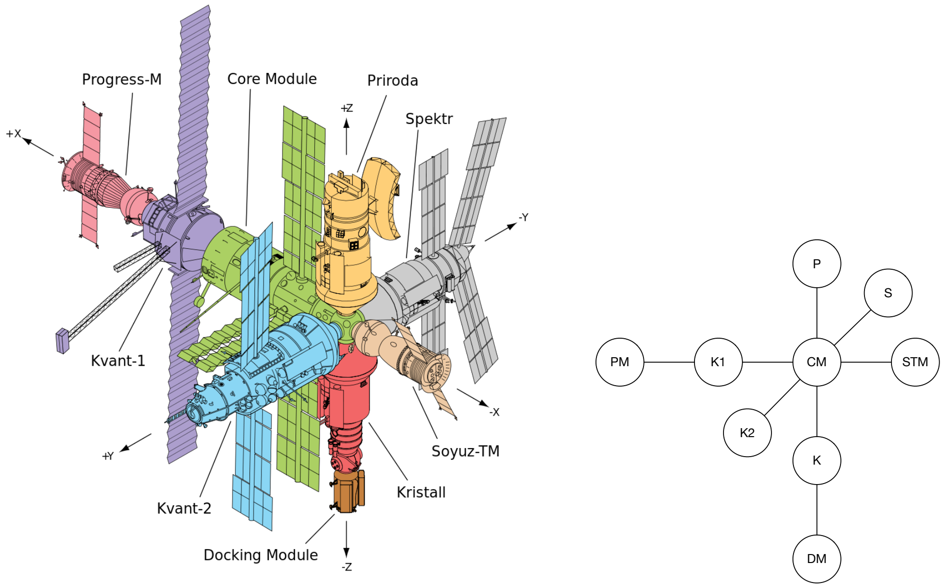
  where *date* is the launch date t_i in the format `DDMMYYYY`, *max_payload* is the maximum payload p_i (float), *fixed_cost* is the fixed cost f_i (float), and *variable_cost* is the variable cost u_i (float).

All other lines should be ignored. Lines may appear in any order. The units are irrelevant but consistent (in the examples below, tons and millions of US Dollars are considered).

As an example, the Mir space station construction plan graph (Figure 2), together with an hypothetical set of launches is described by the following lines:

```
# vertices
VCM   20.4
VDM    4.3
VK    19.64
VK1   20.6
VK2   19.64
VP    19.7
VPM    7.13
VS    19.64
```

[2]If one and only one component is in orbit, it is considered connected.



(a) Station diagram (b) Construction plan graph

Figure 2: Soviet/Russian space station Mir diagram (a), as of May 1996, and the corresponding construction plan graph (b), where the vertex names are the corresponding modules's acronyms.

```
VSTM   7.15

# edges
E VPM VK1
E VK1 VCM
E VCM VP
E VCM VS
E VCM VSTM
E VCM VK
E VCM VK2
E VK  VDM

# launches
L 14082017 22.8    62 0.4
L 10112017 22.8    62 0.4
L 28112017 140     90 1.8
L 26012018 70      85 1.2
L 09032018 22.8    62 1.2
L 14032018 250    132 1.6
L 06062018 22.8    49 0.4
L 22062018 22.8    49 0.4
L 01082018 23      32 1
L 16082018 115     86 1.2
```

The project is to be implemented in the Python programming language (**version 3**) as one programs, that can be use the *uninformed search method* or the *informed search method*, depending on its arguments. It must accepts two arguments: a flag indicating whether an uninformed (`-u`) or informed (`-i`) search method should be used, and the filename of the problem specification. For instance, to solve the above problem using the informed search method one may run[3]

```
python solver.py -i mir.txt
```

assuming that the above file is stored in the current directory with filename `mir.txt`. The name of the programs should be `solver.py`.

The output of the program, in case the problem is feasible, should be a sequence of launch manifests, one per line, in any order, with the following syntax:

launch_ date vertex1 vertex2 ... launch_ cost

where *launch_ date* is a launch date in the format `DDMMYYY`, followed by a list of vertices to be manifested to that launch, and ending with the cost of this launch, defined in (2). Zero cost launches should be omitted. The line following this sequence should contain the total cost of the solution, as defined in (3). A solution to the example problem above is given below:

```
22062018    VK1    57.240000
01082018    VCM    52.400000
16082018    VSTM VPM VK2 VS VDM VP VK    202.640000
312.280000
```

If the problem is infeasible, that is, if there is no solution, the output should be a zero:

```
0
```

4 Assignment goals

1. Develop Python (version 3) code for solving the problem described above using Search methods. The code must include two programs, one using an uninformed search method, and one using an informed search method with an heuristic function.

   In the code, the domain-independent part *should be* explicitly isolated from the domain-dependent one. In particular this should be done using different Python files, one or more for the domain-independent part and one or more for the domain-dependent part. Use self-evident filenames for the Python files. Moreover, use the General Search to develop a generic search algorithm and then particularize it to each one of the two search methods.

   Note: All code should be adequately commented.

[3] Assuming that the default python command is version 3.

2. The answers to the following points should be included in a short **technical** report, delivered together with the source code.

   (a) Describe how do you represent the problem state, the operator(s), the initial state, the goal state, and the path cost;

   (b) Identify and justify the search algorithm implemented;

   (c) Describe and justify the heuristic function developed;

   (d) Evaluate quantitatively and comment on the performance of the used heuristic function.

3. The deliverable of this Lab Assignment consists of a ZIP file containing:

   - the Python source code, including the two programs mentioned above, and
   - the short report (in **PDF** format) with no more than 2 pages.

Submission: electronic submission via *fenix*[4].
Deadline: 23h59 of 3-Nov-2017
(Projects submitted after the deadline will not be considered for evaluation.)

[4]Go to *Student* > *Submit* > *Projects* after login.