# Spacecraft Assembly Problem (SAP)

It is the year 2137. You are consultants and the factory of a client can assemble one spacecraft per week. The factory manager is given a list of orders, each order specifying at the end of which week some ordered spacecrafts should be ready and which spacecraft types they should be. Your job is to decide which week to assemble each ordered spacecraft (or, equivalently, which spacecraft type to assemble each week) in order to minimise the total cost incurred by the storage of the spacecrafts that are completed before their due date and by the adaptation of the factory when switching between spacecraft types. An instance of the SAP is defined by:

- the number `weeks` of weeks for the planning;
- the number `types` of spacecraft types the factory can assemble;
- for each `t` in `1..types` and each `w` in `1..weeks`, the number `Order[t,w]` of spacecrafts of type `t` to assemble by the end of week `w`; you can assume that each `Order[t,w]` is in the integer range `0..1`;
- the cost `storageCost` of storing one spacecraft during one week;
- for each `t1` and `t2` in `1..types`, the cost `SetupCost[t1,t2]` of adapting the factory from assembling spacecrafts of type `t1` to assembling spacecrafts of type `t2`; this cost matrix respects the triangular inequality (for all `i`, `j`, `k` in `1..types`, we have `SetupCost[i,k]` + `SetupCost[k,j]` $\geq$ `SetupCost[i,j]`), but might be asymmetrical, and there is no setup cost when not changing the spacecraft type (for all `i` in `1..types`, we have `SetupCost[i,i]` = 0).

A skeleton MiniZinc model and instances of varying sizes and difficulty, in the form of datafiles using the parameter names above, are attached. Here are some clarifications by the factory manager:

- A spacecraft assembled during the week it is due incurs no storage cost.
- There is no limit on storage space: one can always store as many spacecrafts as needed.
- One cannot assemble an ordered spacecraft after its due date.
- There is no setup cost before the first spacecraft is assembled and there is no setup cost after the last spacecraft is assembled.
- If there is a stretch of one or more weeks with zero assembly directly after the assembly of a spacecraft of type `t1` and directly before the assembly of a spacecraft of type `t2`, then one must still pay the cost `SetupCost[t1,t2]`.

This problem can be modelled using at least two viewpoints: either (1) decide, for each week, which, if any, (type of) spacecraft to assemble; or (2) decide, for each (type of) spacecraft, during which week(s) to assemble it. Perform the following sequence of tasks:

A. Write and evaluate a model using the first viewpoint above.

B. Write and evaluate a model using the second viewpoint above.

C. Write and evaluate a model using the two viewpoints plus channelling constraints.

Note the parenthesised use of "(type of)" in the previous paragraph: it is up to you to decide, for each viewpoint, whether to reason with individual spacecrafts or with spacecraft types. Either way, make sure your first two models do not have essentially the same decision variables.

For each evaluation, use all ten provided instances and report the results for the chosen backends for all the considered solving technologies. The passing requirements are as follows:

1. a full draft report, with at least the first two models, is submitted before the presentation;
2. the draft report is presented during 15 to 20 minutes in the afternoon of either Tue 29 Jan or Wed 30 Jan 2019;
3. the final report, due on Fri 1 Feb 2019 by 17:00, is of sufficient quality, both scientifically and in terms of technical writing, and addresses the feedback received after the presentation;
4. the final report has correct and properly commented CP-style models for all three viewpoints, using suitable global constraints where appropriate;
5. solutions to sap_005_02 (whose minimum is 171), sap_008_03 (699), sap_010_05 (675), sap_010_06, and sap_030_05 are found and proven minimal by at least one backend under at least one viewpoint within 300 CPU seconds;
6. solutions to sap_015_10 and sap_100_15 are found, but not necessarily proven optimal, by at least one backend under at least one viewpoint within 300 CPU seconds.

The reference platform is version 2.2.3 of MiniZinc on any Linux computer of the IT department.