Modelling for Combinatorial Optimisation (1DL451) and Constraint Programming (1DL441)
Uppsala University – Autumn 2019
Assignment 2: Spacecraft Assembly Problem (SAP)

Prepared by Pierre Flener, Gustav Björdal, and Jean-Noël Monette
— Deadline: 13:00 on Wednesday 9 October 2019 —

Read the source code of the relevant demo report at http://user.it.uu.se/~pierref/courses/COCF/demoReport before tackling any assignment: it contains problem-independent instructions on what to hand in, as well as useful problem-independent indications on how to proceed. It is strongly recommended to read the Submission Instructions and Grading Rules at the end of this document before attempting to address its tasks. It is also strongly recommended to prepare and attend the help sessions, as huge time savings may ensue.

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; note that each Order[t,w] can be any non-negative integer;
- 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] ≥ 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 at http://user.it.uu.se/~pierref/courses/COCF/assignments/assignment2/assignment2.zip
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 $t_1$ and directly before the assembly of a spacecraft of type $t_2$, then one must still pay the cost $\text{SetupCost}[t_1,t_2]$.

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

A. In order to reason with individual spacecrafts, pre-compute at least the derived parameter $\text{OrderDueWeek}$, as indicated in the skeleton model. State the used array comprehension(s) also in your report.

B. Write and evaluate a MiniZinc model $\text{SAP1.mzn}$ using the first viewpoint above.

C. Write and evaluate a MiniZinc model $\text{SAP2.mzn}$ using the second viewpoint above.

D. Write and evaluate a MiniZinc model $\text{SAP3.mzn}$ channelling the two viewpoints above.

Make sure your models yield, for each instance, the same optimal objective value, when optimality is proven.

**Hints.** Start with the first viewpoint and model it incrementally, based on the specification: first model the assembly schedule, then extend the model with the storage costs, and finally extend it with the setup costs. Make sure the model is correct before and after each extension. Start working on the second viewpoint only after finishing the model for the first viewpoint: some insights should carry over, which can save you a lot of time. After modelling under the first two viewpoints, the third model could just be the concatenation of the first two models, augmented with channelling constraints. Some constraints might become implied when channelling the two models: removing some of those constraints might improve or worsen the solving time.

For each evaluation, use all provided instances and report the results for the chosen backends for all the considered technologies. Use any time-out of maximum 20 CPU minutes per instance in order to avoid too long solving times. Note that the numbers of weeks and spacecraft types are not necessarily indicative of the difficulty of an instance. See Table 1 at the end of this document for the minimum requirements for each possible score on this assignment. For your convenience, here are the minimal objective values for three instances:

<table>
<thead>
<tr>
<th>instance</th>
<th>minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>sap_005_02</td>
<td>171</td>
</tr>
<tr>
<td>sap_006_02</td>
<td>280</td>
</tr>
<tr>
<td>sap_008_04</td>
<td>431</td>
</tr>
<tr>
<td>sap_010_05</td>
<td>675</td>
</tr>
</tbody>
</table>

Solo teams may skip Task D and skip using $\text{storageCost}$ within their models, but are highly encouraged to try nevertheless.
After performing all the experiments, answer the following sequence of questions:

E. Which model was easier to write and which one is easier to understand?

F. Which improvements of your models have you performed to render them more efficient? Quantify the effect of each improvement, without necessarily running all versions of a model on all the instances under all the chosen backends: use your best judgement.

G. For each of the chosen technologies and backends, which model is more efficient?

H. Which combination of model, technology, and backend would you recommend to the factory manager for solving future instances of the problem? Why? Factor in your answer to Question E as one may also want to consider the maintainability of the chosen model.

I. Briefly describe a real-world situation where (a variation) of the SAP can occur.

Submission Instructions

All task answers, with imported MiniZinc models or (relevant parts of) Gecode programs, must be in a single report in PDF format; all other formats are rejected. Furthermore:

- Identify the team members and state the team number(s) inside the report.
- Address each task of each problem, using the numbering and the ordering in which they appear in this assignment statement.
- Take the instructions of the relevant demo report at http://user.it.uu.se/~pierref/courses/COCP/demoReport as a strict guideline for the structure and content of a model description, model evaluation, and task answer in the report, as well as an indication of the expected quality of the report.
- You must use the script explained in the current MiniZinc cheatsheet at http://user.it.uu.se/~gusbj192/courses/M4CO; it conducts the experiments and generates a result table that can be automatically imported (rather than manually copied) into a LaTeX report: each time you change the model, it suffices to re-run that script and re-compile your report, without any tedious number copying!
- Write clear task answers, source code, and comments.
- Justify all task answers, except where explicitly not required.
- State any assumptions you make that are not in this document.
- Thoroughly proofread, spellcheck, and grammar-check your report.
- Upload all MiniZinc models or Gecode programs.
- Match exactly the uppercase, lowercase, and layout conventions of any filenames and I/O texts imposed by the tasks, as we will process your models automatically.
- Write a paragraph, which will not be graded, describing your experience with this assignment: Which aspects were too difficult or too easy? Which aspects were interesting or boring? This will help us improve the course in the coming years.
- Remember that when submitting you implicitly certify (a) that your report and all its uploaded attachments were produced solely by your team, except where explicitly stated otherwise and clearly referenced, (b) that each teammate can individually explain any part starting from the moment of submitting your report, and (c) that your report and attachments are not freely accessible on a public repository.

Unless the team spans both course 1DL441 and course 1DL451, only one of the teammates submits the solution files (one PDF report with answers to all the tasks, and all model or program files), without folder structure and without compression, via the Student Portal, whose clock may differ from yours, by the given hard deadline.
Grading Rules

If all tasks have been seriously attempted, and all requested models exist in files with the imposed names and the comments exemplified in the relevant demo report, and your models produce correct outputs for some instances in reasonable time, then you score at least 1 point (read on), otherwise your final score is 0 points. Furthermore:

- If your models meet the minimum requirements of the first four rows or all five rows in Table 1 below and your task answers are mostly correct, then you get a final score of 4 or 5 points, depending also on the quality of your model comments; you are not invited to the grading session.
- If your models meet the minimum requirements of at most the first three rows in Table 1 or your task answers have many errors, then you get an initial score of 1 or 2 points, depending also on the quality of your model comments; you are invited to the grading session, where you can try and increase your initial score by 1 point into your final score.

If the assistant figures out a minor fix that is needed to make a model run as per our instructions above, then, instead of giving 0 points, the assistant may deduct 1 point at her/his discretion.

<table>
<thead>
<tr>
<th>score</th>
<th>optimality</th>
<th>feasibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>sap_005_02</td>
<td>sap_008_04</td>
</tr>
<tr>
<td>2.5</td>
<td>sap_008_04</td>
<td>sap_015_10</td>
</tr>
<tr>
<td>3.5</td>
<td>sap_010_05</td>
<td>sap_030_05</td>
</tr>
<tr>
<td>4.5</td>
<td>sap_010_06</td>
<td>sap_100_15</td>
</tr>
<tr>
<td>5</td>
<td>sap_030_05</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: In each row, a solution to the instance in the ‘optimality’ column must be found and proven optimal by at least one backend under at least one viewpoint within 300 CPU seconds, and a solution to the instance in the ‘feasibility’ column must be found by at least one backend under at least one viewpoint within 300 CPU seconds: the reference platform is version 2.3.1 of MiniZinc on any Linux computer of the IT department.