FIT5216: Modelling Discrete Optimization Problems
Assignment 4: Cargo Handling

1 Overview

For this assignment, your task is to write a MiniZinc model for a given problem specification, and a report on your findings.

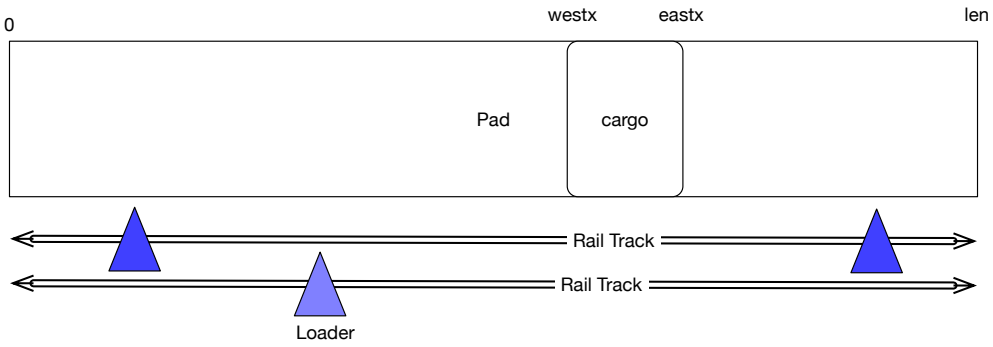
- Submit your model to the MiniZinc auto grading system (using the submit button in the MiniZinc IDE). You must submit via the IDE to be graded and receive marks.
- Note there are two projects for the autograder: for PARTS A-D and PARTS E-F
- Submit your model (copy and paste the contents of the .mzn file) using the Moodle assignment.
- Submit your report using the Moodle assignment.

You have to submit by the due date (Thursday 11th June 2020, 11:59pm), using MiniZinc and using the Moodle assignment, to receive full marks. You can submit as often as you want before the due date. Late submissions without special consideration receive a penalty of 10% per day. Submissions are not accepted more than 3 days after the original deadline.

This is an **individual assignment**. Your submission has to be **entirely your own work**. We will use similarity detection software to detect any attempt at collusion, and the **penalties are quite harsh**. If in doubt, contact your teaching team with any questions!

2 Problem Statement

Your task is to organize operations at a bulk mineral cargo port. The port stores bulk mineral cargoes, e.g. iron ore, for ships that arrive to the port. Each ship takes on a number of cargoes and then departs. The aim of the model is to pack the cargoes onto a one dimensional *pad*, and using (ship) loaders to remove the cargoes onto ships. The loaders have to be scheduled. Hence it is a combined packing and scheduling problem.



The crux of the problem is packing the pad and scheduling the loaders. The loaders run on rails alongside the pad, and there may be two loaders on the same rail, which means they cannot cross. In order to load a cargo the loader must move to its west end. While loading it slowly moves from west end to east end. The time to load the cargo is determined by the size of the cargo. The assignment is broken into parts that each use the same data files.

2.1 Part A: Packing the Pad

The core of the problem is packing cargoes onto the pad and taking them off again. This is a two dimensional packing problem: the first dimension is the length on the pad used, the second is time. Note that cargoes always use the entire height of the pad.

For this part you simply have to pack the cargoes so they don't overlap in space-time.

Data is of the form

```
int: NL; % number of loaders
         % number of rails = NL for parts A-E
         % number of rails = (NL + 1) div 2 for part F
set of int: LOADER = 1..NL;

enum CARGO;
array[CARGO] of int: size; % size in 10000 tonne units

int: makespan;              % maximum time considered.
set of int: TIME = 0..makespan;

enum SHIP; % set of ships
array[CARGO] of SHIP: ship;    % which ship takes a cargo
array[SHIP] of TIME: arrival;  % when ship arrives

int: len; % length of pad
set of int: POS = 0..len;

int: build_time;    % time to build 1 unit (10000 tonnnes)
int: load_time;     % time to load 1 unit
int: loader_speed;  % time for loader to move 1 unit
opt bool: partF;    % are we solving partF
```

For this part we are only interested in the cargo data. The critical decisions for each cargo are the position where it is placed (the western most end `westx`), when it is started to be built (`build`) and when it is started to be loaded (`load`).

The decisions for cargoes are given below. For now the `which` variable is unimportant and can be set to 1.

```
array[CARGO] of var POS: westx;
array[CARGO] of var TIME: build;
array[CARGO] of var TIME: load;
```

`array[CARGO] of var LOADER: which;`

A cargo occupies a position on the pad from its `westx` to its `eastx` = `westx` + `size`. It occupies a time on the pad from its build start time to the end of its loading time. No two cargoes can overlap in both time and position (but note that a cargo could be in position 0..4 and another in position 4..8 at the same time and this is not an overlap, or on the pad from time 0..10, and then another cargo in the same position from time 10..14 and this is not an overlap).

We assume that building a cargo requires `build_time` times its size in time units, and building must be finished before loading begins. The *dwell time* of the cargo is the period in between when building ends and loading begins.

Loading a cargo requires `load_time` times its size.

Output of the plan should be of the form

```
westx = array of positions of western ends;
eastx = array of positions of eastern ends;
build = array of build start times;
endbuild = array of build end times;
load = array of loading start times;
finish = array of loading end times;
which = array of which loaders used;
```

For part A, the `which` array can be set to all 1s.

2.2 Part B: Loader Assignment

Each cargo must be loaded onto a ship by one of the loaders in the port. For part B you need to assign a loader and make sure each loader is only busy at one task at a time. The input and output format is unchanged, but the `which` are now important.

2.3 Part C: Ship Constraints

Each cargo must be loaded onto the ship it is destined for. We now consider the ship information.

No cargo destined for a ship *sh* can be loaded before ship *sh* arrives in port. No two cargoes can be loaded onto the same ship at the same time, they have to go one after the other.

2.4 Part D: Loader Movement

Until now the model ignores the movement of the loaders. This is not correct, they take time to move along the rails.

If a loader finishes loading a cargo with eastern extent x and then has to start loading a cargo with western extent y then there has to be at least $|y - x| \times$ `load_speed` time between these two events.

2.5 Part E: Objective

Now we can consider the objective. The aim is to ensure that ships are not staying in port any longer than required. A ship is at berth from its arrival time to the end of the load time of the last cargo destined for that ship.

Add an objective which minimizes the total time of ships at berth.

2.6 Part F: Rail Constraints

In practice there are two loaders per rail line, and they cannot cross.

Add constraints in your model so that the two loaders on rail i numbered $2i-1$ and $2i$ for $i \in 1..\texttt{NL}$ div 2 remain so the western one $2i-1$ is never east of the eastern one $2i$. Note they can legitimately be in the same position (this is for simplicity, it is not very realistic). Note that if there are an odd number of loaders the last loader is on its own rail and has no further constraints.

2.7 Part G: Report

The operator of the port wants to determine whether it is worthwhile expanding the port. Use and extend your model to determine the value of e.g. adding a new rail and one or two loaders, reducing the build time by improving build speed, buying loaders with faster movement speed, reducing the load time by buying quicker loaders, moving to have three loaders on a rail, or any other modification that you consider could improve the throughput of the port (i.e. reduce the objective).

Use the total set of scenarios given in the `data` directory to determine the worth of each change, where a one time unit change in at berth time is worth $100,000. Write a report suggesting the three most worthwhile upgrades to the port that its management should consider.

Note that if you can't get part F working, then you should make it clear that you are considering extensions to part E in your report.

2.8 Visualization

In order to help you visualize the results of the schedule we provide an alternate output. In order to use this output, remove the comments from the last two lines in the provided models.

This output generates a visual representation of the schedule. The visualisation should automatically load when using the IDE. Note that the visualisation will display the schedule diagrammatically, and can also show an additional integer (the last argument in the call to `cargo_viz`) which is intended to be used to display the objective. For example for the input data

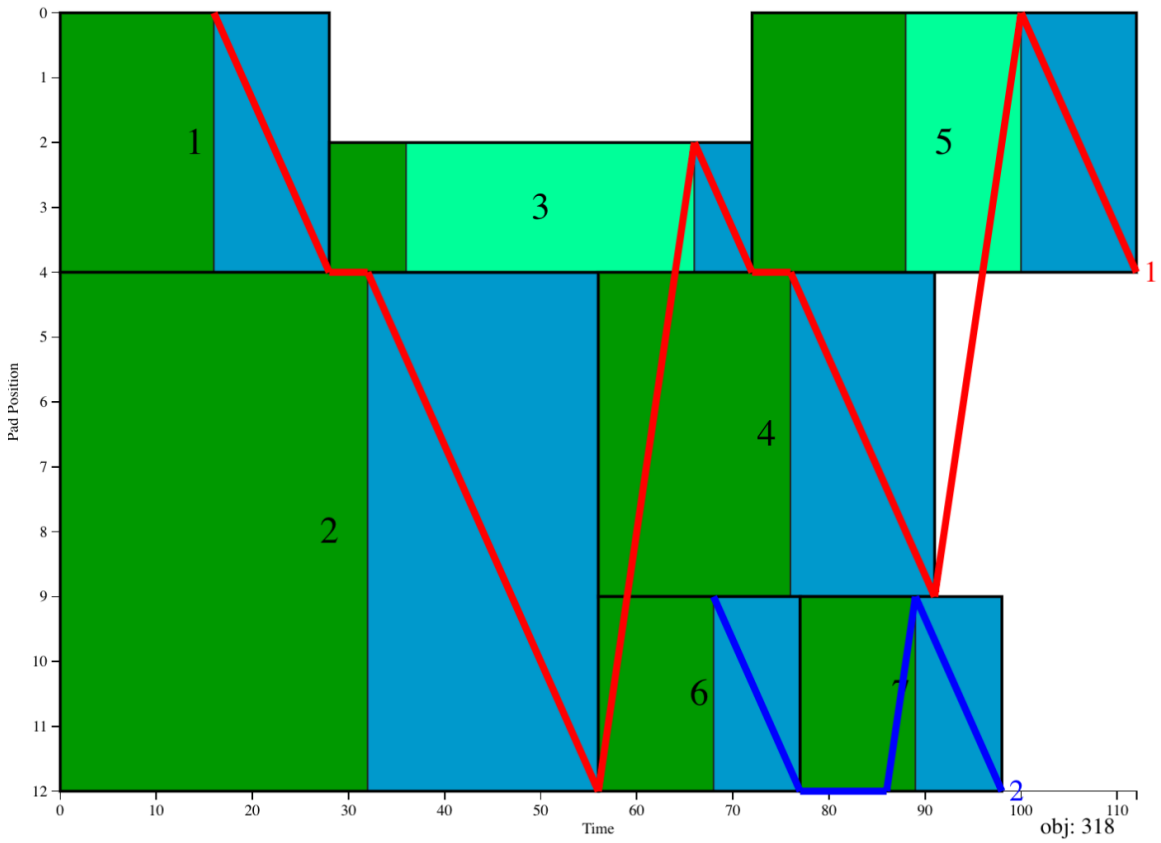
```
NL = 2;
CARGO = anon_enum(7);
size = array1d(CARGO,[4, 8, 2, 5, 4, 3, 3]);
makespan = 200;
SHIP = {SH_1,SH_2,SH_3,SH_4};
ship = array1d(CARGO,[SH_1,SH_2,SH_3,SH_1,SH_4,SH_2,SH_3]);
arrival = array1d(SHIP,[0, 0, 20, 40]);
len = 12;
build_time = 4;
```

```
load_time = 3;
load_speed = 1;
partF = true;
```

gives a solution

```
westx  = [0, 4, 2, 4, 0, 9, 9];
eastx  = [4, 12, 4, 9, 4, 12, 12];
build    = [0, 0, 28, 56, 72, 56, 77];
endbuild = [16, 32, 36, 76, 88, 68, 89];
load  = [16, 32, 66, 76, 100, 68, 89];
finish   = [28, 56, 72, 91, 112, 77, 98];
which    = [1, 1, 1, 1, 1, 2, 2];
```

The visualization results in



Note that the visualization may show an overlap in loader positions, this does not necessarily indicate an error. The grader checks that loaders have a feasible solution, the visualizer doesn't worry about this.

3 Instructions

Edit the provided `mzn` model files to solve the problems described above. You are provided with some sample data files to try your model on. Your implementations can be tested locally by using the *Run* icon in the MiniZinc IDE.

The report has to be submitted in PDF format. It does not have to follow any particular structure (e.g. you do not need an abstract, table of contents etc.). The explanations should be concise. The overall limit for the report is 3 pages, including all graphs, tables etc.

4 Marking

The auto-grading server is using the chuffed solver to grade this assignment. Your model needs to be able to find at least one solution for each instance within the 1 minute time-out in order to get any marks.

You can get a maximum of 40 marks for this assignment, which will be worth 20% of your overall unit marks.

The marks for the correctness of your model are automatically calculated. You will receive up to 5 marks for locally tested data and up to 20 marks for your model as tested on the auto-grading server. You will only get full marks if you implement all parts.

The code comments and your report will be marked by your tutors, and they are worth the remaining 15 marks, with the following allocation:

Code comments: 5 marks, Part G report, 10 marks.