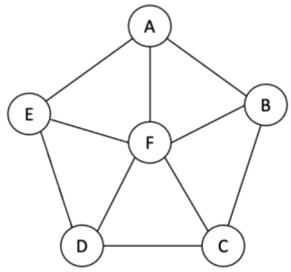
Constraint Programming

2021/2022– Mini-Test #1
Monday, 15 November, 9:00 h, Room 127-II
Duration: 1.5 h (open book)

1. Finite domain Constraints - Propagation (6 pts)



Consider the constraint network on the right, where nodes represent variables, all with domain {**1,2,3**}, and the arcs constraints of difference (**“=”**).

a) **(1 pt.)** Show that the problem is not satisfiable.

b) **(2 pt.)** What pruning is achieved if node-consistency is maintained? And arc-consistency?

c) **(2 pt.)** Show that there are implicit constraints of equality (e.g., A = C) induced by the original constraints. Would maintaining path-consistency detect such constraints? And maintaining 4-consistency?

d) **(1 pt.)** What implicit equality constraints would remain if variable A would have domain {0,1,2,3}. Would these equalities impose the grounding of any variable (i.e. assigning it a fixed value).

2. Global Constraints (5 pts)

Consider a global constraint that given an array of decision variables imposes that they are *shuffled* into another array according to a given index mapping. For example, given a mapping `M`, the decision variables of vector B should be obtained by mapping the variables from vector A according to M, i.e. `B[i] = A[M[i]]`). For example, for `M = {1,2,0}` we should have

```
B[0] = A[1], B[1] = A[2] and B[2] = A[0].
```

a) **(2pt.)** Implement in Choco this constraint in a function **`shuffle`** with signature

```
public static void shuffle (Model md, IntVar [] S, IntVar [] M, IntVar [] T)
```

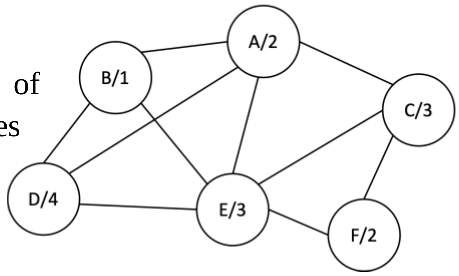
Suggestion: Consider the global constraint **element** available in Choco.

b) **(3 pt.)** Assuming you are given two arrays with **n** decision variables representing the starting times (**S**) and durations (**D**) of **n** tasks, use the **`shuffle`** constraint to complete the code below so as to constrain the tasks to be executed with no overlaps nor gaps between them.

```
n = ...;
IntVar [] S = md.intVarArray(n, sLo, sUp); // the starting times
IntVar [] D = md.intVarArray(n, dLo, dUp); // the duration
...
```

3. Modelling with Finite Domain Constraints (9 pts)

Graph Colouring Variant



Given a graph, the standard Graph Colouring problem consists of finding a colouring of the nodes such that any two adjacent nodes have different colours, and:

(SAT) using only colours of a set **{1, 2, ..., n}**; or

(OPT) minimizing the number of colours **n**.

Here we want to model a variant of this problem, assuming that there is an extra colour, **0**, that satisfies any constraint of difference (i.e., **0≠k**, or **k≠0** is satisfied, for any value of **k**, including **k = 0**).

The figure shows a solution with 4 colours (1 to 4) and no zeros. However, if only **2** colours, plus 0s are acceptable a minimal solution would require that at least two nodes A and E take value 0, and this is an optimal solution (there is no solution with colours **{0,1,2}** that uses less than 2 zeros).

The problem you should model is a mixed problem, such as to minimize the number of 0s in a colouring of the graph with colours in the set **{0, 1, ..., n}**.

Assume that the graph is given as the adjacency matrix (a Boolean matrix where a 1 represents a connection between the corresponding nodes). For the graph, the matrix would be (each row represents a node)

G = {{0,1,1,1,1,0}, {1,0,0,1,1,0}, {1,0,0,0,1,1}, {1,1,0,0,1,0}, {1,1,1,1,0,1}, {0,0,1,0,1,0}}

a) **(2 pt.)** Specify a model for this problem in Choco. More precisely, declare the decision variables you chose as well as their domains, together with the model and solver you propose.

b) **(4 pt.)** (SAT) What constraints would you consider to modelling the satisfaction problem, i.e., to find a solution that satisfies the problem (i.e., only uses the colours from 0 to n)? **Remember** : A zero is considered different from any colour, including the 0.

c) **(3 pt.)** (OPT) Complete the optimisation problem, i.e., to model a problem that aims at minimising the number of 0s in the solution.