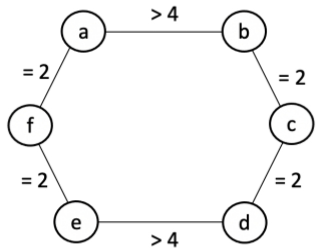
Constraint Programming

2020/2021– Mini-Test #1
Tuesday, 3 November, 9:00 h, Room 204-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 {**1,2,3,4,5,6}**. Arcs labelled "**>p**" and "**=q**" constrain the absolute difference between the connected variables to satisfy that condition (i.e. **| a - b | > 4** and **| a - f | = 2**).

a) **(1 pt)** Is the problem satisfiable? If so, how many different solutions exist? Justify your answer.

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

c) **(2 pt)** Are there any implicit binary constraint in the network? Would path consistency infer such constraints? Justify your answer.

d) **(1 pt)** Notice that the problem exhibits some symmetry, namely horizontal and vertical reflections? Given the analysis you did in the previous items could you add additional symmetry breaking constraints. Justify.

2. Global Constraints (5 pts)

Consider a "global" constraint, `distribute,` that enforces the elements of an array so as the number of times the values appear in a solution are all different, except if they do not appear at all. For example, given an array of 4 elements with domains {0,1,2}, the assignment [2,2,2,2] is a possible solution (2 appears 4 times and both 0 and 1 appear 0 times), as is [1,1,1,2] (1 appears 3 times, 2 appears once, and 0 does not appear), but not [0,2,2,1] since values 0 and 1 appear the same number of times (once) in that assignment.

a) **(3pt)** Implement in Choco this constraint in a function **distribute** with signature

```
function void distribute(Model md, IntVar [] A, int n)
```

where the domain of all variables in array A can only take values in the range 0..n-1.

Suggestion: Use the predefined constraints

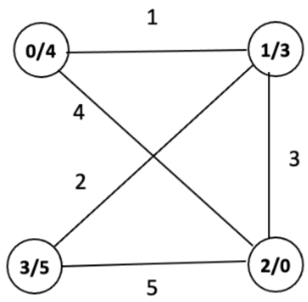
- `count(int v, IntVar [] A , IntVar c) : it` constrains the number of variables in `A` that have value `v` to be exactly `c`; and
- `allDifferentExcept0(A): it` constrains the variables in `A` to have different values, except value 0 that may appear more than once.

b) **(2 pt)** If your implementation of this global constraint maintained domain consistency, what values would be pruned from the domain of the array A, with the initial domains shown.

```
A[0] in {2, 4}
A[1] in {0, 1}
A[2] in {1, 4}
A[3] in {0, 3, 5}
A[4] in {0, 3, 5}
A[5] in {2, 3}
```

3. Modelling with Finite Domain Constraints (9 pts)

Graceful Graphs (Prob. 53 of CSPLIB)



As shown in the figure, a labelling **f** of the nodes of a graph with **q** edges is graceful if:

- **f** assigns each node a unique label from 0, 1, … , q;
- each arc (**x,y)** is labelled **with |f(x)−f(y)|**; and
- the arc labels are **all different**.

The graph shown represents a graceful labelling, where each node is denoted with a pair a/b where a in the *order* of the node and b its **label** (for example the arc connecting **node 3** with **label 5** and **node 1** with **label 3** has **label 2 = 5 – 3**).

Assume that you are given the number of nodes, **n,** and a graph specified by an **m x 2** matrix where each row represents an arc, and the two columns represent the order of the nodes connected by the arc. For example, the graph above is represented by matrix **G** and the number of nodes **n**:

```
int [][] G = {{0,1},{0,2},{1,2},{1,3},{2,3}};
int n = 4;
```

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 graceful labelling of the graph?

c) **(3 pt)** (OPT) Assume that a rank is defined for a label of the graph as the sum of the product of the order of the nodes by their label. For example, the label shown has rank

```
R = 0*4 + 1*3 + 2*0 + 3*5 = 18
```

Adapt your model to find the minimal graceful label of the graph.