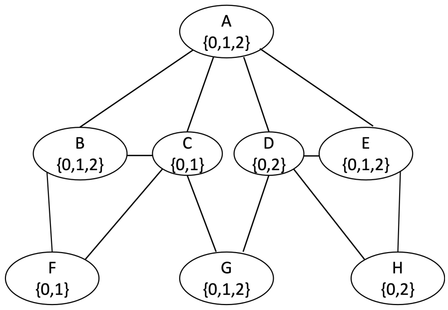
Constraint Programming

2021/2022– Exam #1
Friday, 21 January 2022, 13:00 h, Room XXX
Duration: 1.5 h (open book)



1. Finite domain Constraints - Propagation (6 pts)

Consider the constraint network on the right. All arcs represent constraints of difference (≠). Variables represented in the nodes have the domains shown (for example, A has domain {0,1,2} and C has domain {0,1}.

a) **(2 pt)** Is the problem satisfiable? If so, how many different solutions exist? Justify your answer.

b) **(1 pt)** What pruning is achieved initially, if node-consistency is maintained? And arc-consistency?

c) **(2 pt)** Would maintaining path consistency infer any type of constraints? Would it fix any value for the variables? Justify your answer.

d) **(1 pt)** Assuming now that the variables may have arbitrary domains and the constraints are arbitrary, can you show that if the constraint network is satisfiable if it can be made path-consistency? Justify your answer.

2. Global Constraints (5 pts).

Consider a "global" constraint, **`smooth_increase(V, k)`**, on a `IntVarArray V` and an `IntVar k`, that constrains the values of these arrays to be in increasing order, such that the difference between consecutive variables is either `1` or `2`, and where value `2` should occur exactly `k` times. For example, array `V = {4,5,7,9}` and `k = 2` satisfy the constraint, whereas `V = {4,5,8,9}` and `k = 2` do not satisfy the constraint (the difference between 5 and 8 is 3). Nor do `V = [4,5,6,8]` and `k = 2` satisfy the constraint, since a difference of 2 occurs only once (8-6), but not twice.

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

```
function void smooth_increase(Model md, IntVar [] V, IntVar k)
```

Note: You may, but not necessarily, use the predefined **`count`** constraint, with the syntax below, which constrains the number of variables in **`A`** that have value **`v`** to be exactly **`c`**.

```
count(int v, IntVar [] A , IntVar c)
```

b) **(2 pt)** If your implementation would maintain domain consistency, what values would be pruned from the domains of the array **`V`**, and variable **`k`**, with initial domains below

```
k in {3,4,5}     and     V[0] in {  2,3,4,5,  7,8  }
                         V[1] in {1,2,  4,  6,7,8  }
                         V[2] in {1,  3,4,  6,  8,9}
                         V[3] in {  2,3,4,5,  7    }
                         V[4] in {  2,  4,  6,  8,9}
```

c) **(1 pt)** In the same conditions of the previous item, assume that constraint `k > 3` is posted. Would there be any further pruning of the domains?

3. Modelling with Finite Domain Constraints (9 pts)

Problem: Bin Packing (variant)

There are several variants of the bin packing problem that consist of fitting a set of items can be fitted in a set of bins. Here we consider the following variant:

Given a set of n items with volumes $v_1, \ldots, v_n$ and a set of k bins, all with the same capacity C, minimize the number of bins needed to fit all the items. Of course, the capacity C of the bins is larger than volume of the largest item ($C > v_i$, for all i).

a) **(3 pt)** Specify a model for this problem in Choco. First, declare the decision variables you adopt as well as their domains, together with the model and solver you propose. Assume the set of volumes is given by a integer vector `V` with size `n`, where all the elements are positive integers and the capacity of the bin is a given constant, as in

```
int V = [...]; // problem size
int C = ...; // Bins capacity
```

b) **(3 pt)** What constraints would you consider for modelling the satisfaction problem, i.e. to fit a bin for every item, even if the number of items is minimized?

Suggestion: Assume you have available a constraint

```
sum_if(int [] A, intVar[] B, int k, intVar s)
```

that given arrays `A` and `B` of the same size, constrains variable `s` to be the sum of all values in `A` where the corresponding values in `B` are `k`. For example, for `A = {1,3,2,4,2}`, `B = {5,6,1,9,7}` and `k=2`, variable `s` is constrained to `8 = 1+7.`

c) **(3 pt)** Model now the minimization problem, i.e. extend the problem with constraints that enforce the solution to minimize the number of bins