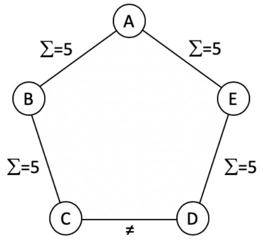
Constraint Programming

2019/2020– Mini-Test #1
Friday, 31 October, 20:00 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 **{1,2,3,4}**. Arcs labelled Σ=k (e.g. between variables x and y) constrain the connected variables to have a sum equal to k (i.e. x+y = k). Arcs labelled ≠ indicate constraints of difference.

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)** Are there any implicit binary constraint in the network? What type of consistency would be needed to infer such constraints? Justify your answer.

d) **(1 pt)** What priunning would be achieved by path-consistency? Justify your answer.

Proposed Solution

a) The problem is not satisfiable, and this can be shown as follows:

- Variables A and C must have the same value as they have the same sum wrt B. Similarly
- Variables A and D must have the same value (equal sum wrt. variable E),
- Hence, variables C and D must have the same value, due to the above Σ=5 constraints.

However, variables C and D must be different, according to the ≠ constraint. Hence the problem has no solutions.

b) Node-consistency does not prune any domain, since there are no unary constraints. Arc-consistency does not achieve any pruning either, since for any constraint Σ=5 any value in the domain {1,2,3,4} of one variable has a support in the other variable (since 1+4=5, 2+3 = 5). On the other hand, the ≠ constraint between variables C and D does not prune any value of the variables (they both have more than 1 value in their domains).

c) As hinted in answer a) there are implicit equality constraints between variables A, B, C, D and E namely A = C = D and B = E. For example, since since A+B = 5 and A+E = 5, it must be B = E.

d) Path consistency imposes some implicit binary constraints between the variables, and in particular it imposes C = D, which is contradictory with the explicit constraint C ≠ D. Hence, imposing path-consistency would detect the unsatisfiability of the problem.

2. Global Constraints (5 pts)

Consider a "global" constraint, `pairing`, that enforces the elements of two arrays `A` and `B` of decision variables with equal size , whose domains are composed of values in the range `0..k`, to have the same number of values, i.e. if `q` variables of `A` take value `p`, then `q` variables of `A` should also take that value.

For example, arrays `A = [2,2,7,7,1,7]` and `B = [7,1,7,2,7,2]` satisfy the constraint, but arrays `A` and `C = [2,2,2,7,1,7]` do not (since value 2 appears twice in `A` but three times in `C`).

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

```
function void pairing(Model md, IntVar [] A, IntVar [] B, int k)
```

Suggestion: Use the predefined count constraint, with the syntax below, which constrains the number of variables in `A` to have value `v` to be exactly `c`.

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

b) **(2 pt)** If your implementation of this global constraint were such that it maintained domain consistency, what values would be pruned from the domain of the array A, if the domain of its elements were.

`A[0] in {6, 8,` ~~`9`~~`}`	`B[0] in {1, 3, 5}`
`A[1] in {1, 3, 5}`	`B[1] in {`~~`1, 2, 4,`~~ `6, 8}`
`A[2] in {3, 5, 7,` ~~`9`~~`}`	`B[2] in {1, 4,` ~~`5`~~`}`
`A[3] in {4,` ~~`9`~~`}`	`B[3] in {1,` ~~`2`~~`, 7}`
`A[4] in {6, 8}`	`B[4] in {1,` ~~`2`~~`, 6}`

c) **(1 pt)** In the same conditions of the previous item, assume that constraint `A[1]+B[3] < 7` is posted. Would there be any further pruning of the domains?

Proposed Solution

a) Since we know that the possible values in the variables range form `0..k`, we only have to count the number of occurrences of each of these values in an IntVar array `C` of `k+1` elements, and guarantee that both arrays `A` and `B` have the same counts for each of the values.

```
function void (Model md, IntVar [] A, IntVar [] B, int k){
     IntVar [] C = intVarArray("C", k+1, 0, k)
     for (int v = 0; v < k; v++){
          md.count(v, A, C[v]);
          md.count(v, B, C[v]);
     }
}
```

b) Domain pruning should remove those values that appear in the domain of the variables of one of the arrays, but not in the other. This is the case of value 9, that should be removed from the domain of variables `A[0], A[2]` and `A[3]`, and value 2, that should be removed from `B[3]` and `B[4]`.

Now, variable `A[3]` can only take value `4`, and at least one variable in `B` should take that value. Since the only one is `B[2]`, the constraint should fix `A[3] = B[2] = 4`.

Since `A[0]` and `A[4]` have only the values `6` and `8` in their domain the corresponding values should appear twice in `B`. Since `6` and `8` only appear in `B[1]` and `B[4]`, they must take these values (given the pigeon hole principle). Finally, the remaining values of variables `A[1]` and `A[2]` have compatible values `{1,3,5,7}` in variables `B[0]` and `B[3]`.

c) If `A[1]+B[3] < 5`, than value 7 is removed from `B[3]`, and hence `B[3]` is fixed to `1`. Moreover, value `5` is removed from `A[1]`. Since there is only one `1` in the domain of the variables of `A`, it must then be `A[1] = B[3] = 1`.

Moreover, value `7` must be removed from `A[2]`, since there is no longer any variable in `B` with `7` in its domain.

Finally, variables `A[2]` and `B[0]` will have their domains pruned to `{3, 5}`, with an implicit constraint that they should have the same value (`A[2] = B[0]`).

3. Modelling with Finite Domain Constraints (9 pts)

Low Autocorrelation Binary Sequences (Prob. 5 of CSPLIB)

These problems have many practical applications in communications and electrical engineering. The objective is to construct a binary sequence S_i of length n that minimizes the autocorrelations between bits. Each bit in the sequence takes the value +1 or -1. With non-periodic (or open) boundary conditions, the k^th autocorrelation, C_k is defined to be $\sum_{i=0}^{n-k-1} S_i \times S_{i+k}$.

For example, for sequence S_1 = [1, -1, 1, 1, -1] and n = 5 it is

$C_1 = -2 = S_0 * S_1 + S_1 * S_2 + S_2 * S_3 + S_3 * S_4 = - 1 - 1 + 1 -1$

$C_2 = -1 = S_0 * S_2 + S_1 * S_3 + S_2 * S_4 = 1 - 1 - 1$

$C_3 = 2 = S_0 * S_3 + S_1 * S_4 = 1 + 1$

$C_4 = -1 = S_0 * S_4 = -1$

The ultimate aim (i.e. problem LABS_OPT(n)), is to minimize a sequence S of size n, for which the sum of the squares of these autocorrelations, $E = \sum_{k=1}^{n-1} C_k^2$, is minimal. Nevertheless we may also consider the satisfaction version of the problem, LABS_SAT(n, Z) that aims to find a sequence S of size n with an autocorrelation less that a certain value Z. In the example above, we have

$E(S_1) = 10 = (-2)^2 + (-1)2 + (2)^2 + (-1)^2$

However, for sequence S_2 = [-1, -1, -1, 1, -1] we would have $E(S_2) = 2 = (0)^2 + (-1)^2 + (0)^2 + (1)^2$, which is lower than $E(S_1)$.

Hence, S_1 is not a solution of the problem LABS_SAT(5,9) but S_2 is. In fact, S_2 is a solution of the problem LABS_OPT(5), since there is no sequences S_x of size 5 for which $E(S_x) < 2$.

a) **(2 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 an arbitrary value of n.

b) **(4 pt)** (SAT) Next, what constraints would you consider to model the satisfaction problem (n,Z), i.e. find a binary sequence S (± 1 bits) such that E(S) < Z?

Note: To simplify notation, assume that to model the summation operator, where X is a decision variable, `b >= a` are integers, and E[i] are expressions on integers (decision) variables.

$$X = \sum_{i=a}^{b} E[i]$$

you can use a built-in constraint (similar to that there is in Comet) with syntax

```
model.sum(i in (a, b), E[i], X).post()
```

Assume further that Boolean (decision) variables are automatically cast to 0/1 (decision) variables, that can be summed.

c) **(1 pt)** (OPT) Adapt the previous model to consider the minimization problem, i.e. find a binary S (± 1 bits) that minimises E(S).

d) **(2 pt)** Notice that for any sequence S there is a symmetric sequence T (i.e. where T[i] = - S[i], for all bits). How would you adapt your model to avoid computing symmetric sequences?

Proposed Solution:

a) We should organise a solution as an array of n binary digits, S_0 to S_{n-1}, and consider the correlation coefficients C_1 to C_k, and their sum E. Hence, we could use the following decision variables (where the bounds n*n*n are very conservative estimates)

```
int n = ...;
Model md = new Model();
Solver sv = md.getSolver();
IntVar [] S = new md.intVarArray("S", n, {-1,1});
IntVar [] C = new md.intVarArray("C", 0, n*n*n)
ntVar E = md.intVar("E", 0, n*n*n);
```

b) We should express the coefficients C from the decision variables S, and have the sum of their squares assigned to E. Moreover we should constraint E to be lower than Z. With the suggested notation this could be modelled as follows.

```
// express the correlation coefficients as sums of products
for (int k = 0; k < n; k++)
    model.sum(i in (0, n-k), S[i]*S[i+j], C[k]).post()

// express E as the sum of the squares of the
model.sum(i in (1, n-1), C[i]**C[i], E);

// constrain E to be less than Z
model.arithm(E, "< ", Z);
```

c) To specify a minimization problem, all that is required is to explicitely set this objective, to be added to the previous model

```
// set E as the variable to minimize
md.setObjective(Model.MINIMIZE, E);
```

d) To avoid symmetric solutions and since S[0] can only be +1 or -1, a symmetry breaking constraint would impose only one of these values, for example S[0] = 1.

```
// avoid symmetric solutions
model.arithm(S[0], "=", 1);
```