Department of Mathematics and Computer Science
University of Southern Denmark, Odense

Friday 13th October, 2023
Marco Chiarandini

DM841 - Discrete Optimization

Exercise 1

Recall the definition of domain consistency and relaxed domain consistency such as bound(**Z**), range, and bound(**D**) consistency. What is the consistency level of the following CSPs?

$$\mathcal{P} = \langle x_1 \in \{1,3\}, x_2 \in \{1,3\}, x_3 \in \{1,3\}, x_4 \in \{1,3\}, \mathcal{C} \equiv \text{alldifferent}(x_1, x_2, x_3, x_4)\rangle$$

Solution

The weakiest form is bound(**Z**) consistency. Accordingly we should ask whether the bounds of x_i, $i = 1,2,3,4$ have a bounded support. For $x_1 = 1$ we can have $x_2 = 2$, $x_3 = 3$ but no value would be left for x_4. Hence, $\mathcal{P}$ is not bound(**Z**) consistent. Since this is the weakiest form of those asked all the others are also not satisfied.

$$\mathcal{P} = \langle x_1 \in \{1,2,3\}, x_2 \in \{2,3\}, x_3 \in \{2,3\}, x_4 \in \{1,2,3,4\}, \mathcal{C} \equiv \text{alldifferent}(x_1, x_2, x_3, x_4)\rangle$$

Solution

Same as above, it is not consistent in any of the forms.

$$\mathcal{P} = \langle x_1 \in \{1,3\}, x_2 \in \{2\}, x_3 \in \{1,2,3\}, \mathcal{C} \equiv \text{alldifferent}(x_1, x_2, x_3)\rangle$$

Solution

It is bound(**Z**). For being range consistent all values of x_i $i = 1,2,3$ must belong to a bounded support. This is not true since $x_3 = 2$ has no support in x_2. For being bound(**D**) it must be that for all x_i, $i = 1,2,3$ its bounds belong to a support. This holds true for this case. Since it is not range consistent then it is also not arc consistent.

$$\mathcal{P} = \langle x_1 \in \{1,3\}, x_2 \in \{2\}, x_3 \in \{1,3\}, \mathcal{C} \equiv \text{alldifferent}(x_1, x_2, x_3)\rangle$$

Solution

It is bound(**Z**) and bound(**D**). Since 2 is not anymore in the domain of x_3 then it is also range consistent. It is also arc consistent.

$$\mathcal{P} = \langle x_1 \in \{1,3\}, x_2 \in \{1,3\}, x_3 \in \{1,3\}, \mathcal{C} \equiv \text{alldifferent}(x_1, x_2, x_3)\rangle$$

Solution

bound(**Z**): yes, bounded support hence we can reintroduce the value 2 in the support variables
range: yes, bounded support hence we can reintroduce the value 2 in the support variables
bound(**D**): no.

Exercise 2

Let $V = \{\{x, y, z\}$ and $D(x) = \{3, 4, 5\}, D(y) = \{0, 1, 2, 3\}, D(z) = \{1, 5\}\}$. Define one or more propagators implementing the constraint $x \leq y$. Compute the propagator on the constraint store defined. Is the propagator strong idempotent? Is it weak idempotent?

Solution

A propagator p_{leq} for $x \leq y$ can be defined as follows:

$$p_{\leq}(D)(x) = \{n \in D(x) \mid n \leq \max D(y)\}$$
$$p_{\leq}(D)(y) = \{n \in D(y) \mid n \geq \min D(x)\}$$
$$p_{\leq}(D)(z) = D(z)$$

The propagator computes:

$$p_{\leq}(D)(x) = \{n \in D(x) \mid n \leq 3\}$$
$$p_{\leq}(D)(y) = \{n \in D(y) \mid n \geq 3\}$$
$$p_{\leq}(D)(z) = D(z)$$
$$\{D(x) = \{3\}, D(y) = \{3\}, D(z) = \{1, 5\}\}$$

We should distinguish the properties of each propagator individually taken from the propagator that arises from the collection of the three. Each individual propagator is strongly idempotent because the current space cannot be tighten by another application of the propagator. The overall p_{leq} propagator is also strongly idempotent. Applying it twice would not change the result.
Both cases are not subsumed: for some tightenings if the domain of a variable changes then it might propagate again.

Exercise 3

Define a propagator for $x + y = d$ and then for $ax + by = d$. Finally, generalize it to the linear equality $\sum a_i x_i = d$.

- Is it necessary to perform several iterations of your propagator? Is it idempotent?
- What type of consistency it produces? (domain or bound consistency level?)
- Apply the propagator to the following example: $D(x) = \{2..7\}, D(y) = \{0..2\}, D(z) = \{-1..2\}$, $x = 3y + 5z$. Compare the result with your answer to the previous point.

Solution

Domain consistency:

$$D(x) \leftarrow \{n \in D(x) \mid \exists n_y \in D(y), n_z \in D(z) : n = n_y + n_z\}$$

$$\forall n_x \in D(x)\, \exists n_y \in D(y), n_z \in D(z) : n = n_y + n_z$$
$$\forall n_y \in D(y)\, \exists n_x \in D(x), n_z \in D(z) : n = n_x - n_z$$
$$\forall n_z \in D(z)\, \exists n_x \in D(x), n_y \in D(y) : n = n_x - n_y$$

It is strongly idempotent.
Range consistency:

$$D(x) \leftarrow \{n \in D(x) \mid n \leq \max D(y) + \max D(z) \wedge n \geq \min D(y) + \min D(z)\}$$

Bound($\mathbb{Z}$) consistency:

$$D(x) \leftarrow \{n \in D(x) \mid n \leq \max D(y) + \max D(z) \wedge n \geq \min D(y) + \min D(z)\}$$

$$\begin{array}{l} \forall n_x \in \{\min D(x), \max D(x)\} \\ \exists n_y \in [\min D(y) .. \max D(y)] \\ \exists n_z \in [\min D(z) .. \max D(z)] \\ n_x = n_z + n_y \end{array}$$

$x + y = z$

$x \in \{1,2,4,8\}$	$y \in \{1,2,4\}$	$z \in \{1,2,3,4,6\}$	
$x \in \{2,4,8\}$	$y \in \{1,3,4\}$	$z \in \{1,3,4\}$	domain
$x \in \{2,4,8\}$	$y \in \{1,3,4\}$	$z \in \{1,2,3,4\}$	bound(Z)

$$\begin{array}{l} D(x) \leftarrow \{n \in D(x) \mid \min D(z) - \max D(y) \leq n \leq \max D(z) - \min D(y)\} \\ D(y) \leftarrow \{n \in D(y) \mid \min D(z) - \max D(x) \leq n \leq \max D(z) - \min D(x)\} \\ D(z) \leftarrow \{n \in D(z) \mid \min D(x) + \min D(y) \leq n \leq \max D(x) + \max D(y)\} \end{array}$$

They are each strongly idempotent. But all together they are not idempotent. Hence, they need to be iterated before reaching fixed point.

$x = 3y + 5z$

$x \in \{2..7\}$	$y \in \{0..2\}$	$z \in \{-1..2\}$	
$x \in \{3,5,6\}$	$y \in \{0,1,2\}$	$z \in \{0,1\}$	domain
$x \in \{2..7\}$	$y \in \{0..2\}$	$z \in \{0,1\}$	bound(Z)

$$\begin{array}{l} D(x) \leftarrow \{n \in D(x) \mid 3 \min D(y) + 5 \min D(z) \leq n \leq 3 \max D(y) + 5 \max D(z)\} \\ D(x) \leftarrow \{n \in D(x) \mid 3 \cdot 0 + 5 \cdot (-1) \leq n \leq 3 \cdot 2 + 5 \cdot 2\} \\ D(x) \leftarrow \{n \in D(x) \mid -2 \leq n \leq 16\} \\ D(x) \leftarrow \{2..7\} \end{array}$$

$$\begin{array}{l} D(y) \leftarrow \{n \in D(y) \mid -5/3 \max D(z) + 1/3 \min D(x) \leq n \leq -5/3 \min D(z) + 1/3 \max D(x)\} \\ D(y) \leftarrow \{n \in D(y) \mid -5/3 \cdot (2) + 1/3 \cdot 2 \leq n \leq -5/3 \cdot (-1) + 1/3 \cdot 7\} \\ D(y) \leftarrow \{n \in D(y) \mid -10/3 + 1/3 \leq n \leq 5/3 + 7/3\} \\ D(y) \leftarrow \{n \in D(y) \mid -3 \leq n \leq 4\} \\ D(y) \leftarrow \{0..2\} \end{array}$$

$$\begin{array}{l} D(z) \leftarrow \{n \in D(z) \mid 1/5 \min D(x) - 3/5 \max D(y) \leq n \leq 1/5 \max D(x) - 3/5 \min D(y)\} \\ D(z) \leftarrow \{n \in D(z) \mid 1/5 \cdot 2 - 3/5 \cdot 2 \leq n \leq 1/5 \cdot 7 - 3/5 \cdot 0\} \\ D(z) \leftarrow \{n \in D(z) \mid -4/5 \leq n \leq 7/5\} \\ D(z) \leftarrow \{0,1\} \end{array}$$

Checking y and x does not change the situation.

Exercise 4 Usefulness of Weak Idempotency

Assume $V = \{x, y\}$ and $D(x) = [0..3], D(y) = [0..5]$ Consider the constraint $3x = 2y$ and the propagator p_{32}

$$\begin{array}{l} p_{32}(D)(x) = D(x) \cap \{\lceil (2 \min D(y))/3 \rceil, \ldots, \lfloor (2 \max D(y))/3 \rfloor\} \\ p_{32}(D)(y) = D(y) \cap \{\lceil (3 \min D(x))/2 \rceil, \ldots, \lfloor (3 \max D(x))/2 \rfloor\} \end{array}$$

Apply the propagator three times and state whether the propagator is strong idempotent. If it is not is there a constraint store for which it is weak idempotent?

Solution

The propagator p_{32} is not idempotent. $D' = p_{32}(D)$ is

$$D'(x) = [0..3] \cap [0..\lfloor 10/3 \rfloor] = [0..3]$$
$$D'(y) = [0..5] \cap [0..\lfloor 9/2 \rfloor] = [0..4]$$

Now $D'' = p_{32}(D')$ is

$$D''(x) = [0..3] \cap [0..\lfloor 8/3 \rfloor] = [0..2]$$
$$D''(y) = [0..4] \cap [0..\lfloor 9/2 \rfloor] = [0..4]$$

Hence $p_{32}(p_{32}(\mathcal{D})) = \mathcal{D}'' \neq \mathcal{D}' = p_{32}(\mathcal{D})$ and the propagator is not idempotent. Further $D''' = p_{32}(D'')$ is

$$D'''(x) = [0..2] \cap [0..\lfloor 8/3 \rfloor] = [0..2]$$
$$D'''(y) = [0..4] \cap [0..\lfloor 6/2 \rfloor] = [0..3]$$

The new bound for y is 3 and is obtained without rounding. In this case we are guaranteed that the proagator is at a fixedpoint, that is, it is weakly idempotent with respect to the reached domain extension $\mathcal{D}'''$. Knowing that it is weakly idempotent on $\mathcal{D}'''$ is useful since we can avoid running the propagator again unlees the problem changes.

Exercise 5

How would you implement a propagator for $\max(x, y) = z$?

Solution

Let's rewrite constraint as:

$$z = \max(x_1, x_2)?$$

and write the propagator for z. Using as example the revise reduction rule presented in Slide 17 of "Propagation Events and Implementations" and assuming to have the information `Mtype` and Δ_j:

```
function propagate_max(inout: z; in: z=max(x_i,x_j), Mtype, Delta_j):
    if min(x_i)>max(x_j):
      changes, Delta_z <-  propagate_=(inout: z; in: z=x_i )
      return changes, Delta
    Changes<-0
    switch Mtype do:
      case RemValue:
        nothing;
      case IncMin:
        remove all v > max(min(x_i),min(x_j)) from D(z)
      case DecMax:
        remove all v > max(max(x_i),max(x_j)) from D(z)
      case Instantiate to k:
        if (min(x_i)>k)
          nothing
        else if (max(x_i)<k)
          remove all v != k from D(z)

    Changes <- the types of changes performed;
    Delta_i <- all values removed from D(x_i);
    return Changes, Delta_i
```

Exercise 6

The `element` constraint models the following very common situation: we have to decide which among four goods to buy; for each good we have a different price; how to propagate the price while the variable is not yet assigned a good?

- Define a way to handle this situation without using the `element` constraint.
- Define a propagator for the `element`(a, x, y) constraint.
- Is it necessary to perform several iterations of your propagator? Is it idempotent?
- What type of consistency it produces? (domain or bound consistency level?)
- Run your propagator on the following example: $a = [4, 5, 7, 9], D(x) = \{1, 2, 3\}, D(y) = \{2..8\}$.
- Can you devise a clever data structure that would speed up the propagation? Would it be worth storing the data structure for successive iterations?
- In gecode the element constraint is used also to implement a particular type of channeling, namely, $z = x_y$, where y and z are single variables and x is an array of variables. Write formally the propagators for x, y, z for the constraint element(z, x, y).

Solution

Exercise 7 Steel Mill Slab Design

Model the following problem and report the model in written form.
Steel is produced by casting molten iron into slabs. A steel mill can produce a finite number, σ, of *slab sizes*. Let sizes be expressed in weight. An order has two properties, a *colour* corresponding to the route required through the steel mill and a *weight*. Given d input orders, the problem is to pack the orders onto slabs, the number and size of which are also to be determined, such that the total capacity of the slabs in minimized. In other words, we want to minimize the waste of steel (steel produced in addition to the actual sizes of orders). This assignment is subject to two further constraints:

Capacity constraints: The total weight of orders assigned to a slab cannot exceed the slab capacity.

Colour constraints: Each slab can contain at most p of k total colours (p is usually 2). The colour constraints arise because it is expensive to cut up slabs in order to send them to different parts of the mill.

The above description is a simplification of a real industrial problem. For example, the problem may also include inventory matching, where surplus stock can be used to fulfil some of the orders.
A numerical example:
$\sigma = 3, d = 5, k = 3, p = 2$. Slab sizes = '12, 10, 7' (note, we can use more than one slab of a certain size and we do not need to use all slab sizes).

Order	1	2	3	4	5
Size	5	8	3	4	6
Color	1	3	2	1	2

A solution:

```
--------------- ------- ---- --- -- --
Slab Capacity     12     10   7
Orders           1,3,4   2    5
Load              12     8    6
Loss               0     2    1
--------------- ------- ---- --- -- --
```

Solution

When first formulated, problems may look undefined. For this it is important to ask questions in order to make the formulation crystal clear. For this specific case, the following is an improved formulation.

The problem consists in producing d orders from a set of slabs. Several orders can be made from the same slab but there is no limitation on the number of slabs that can be requested. Each order o has a color c_o and requires an amount of capacity (weight) w_o of the slab to which it is assigned. Each slab has a capacity (weight) that must be chosen from the increasing set of weights $\{u_1, u_2, ..., u_k\}$.
The constraints of the problem are:

1. an order must be produced from a single slab,
2. the sum of order weights on a slab must not exceed the slab capacity,
3. a slab can be used to produce orders of at most p different colors.

The first two constraint describe a bin-packing problem. The third one is called the color constraint. Therefore, this problem is also called a variable sized bin-packing problem with color constraints in the literature. In the steel mill slab design problem, the objective is to use as few slabs as possible to satisfy the demand. More precisely, the objective is to minimize the cumulative sum of the weights of the slabs used. An obvious lower bound to this problem is the sum of order weights.

Here is another small, illustrative example of an instance and of a solution. Assume that we have $p = 2$ and $d = 10$ orders whose weights and colors are:

Order	1	2	3	4	5	6	7	8	9	10
Weight	1	3	2	9	9	11	3	3	5	2
Color	Red	Black	Black	Red	Red	White	Red	White	Black	Red

and let the set of possible slab weight be $\{5, 7, 9, 11, 15, 18\}$. A solution to this problem is to use 4 slabs and assign the orders in the following way:

Slab	Orders	Weight sum	Slab weight
1	1, 2, 3	6	7
2	4, 5	9	9
3	6, 7, 8	17	18
4	9, 10	7	7

Note that in this solution:

- there are no more than two different colors on the same slab;
- the maximum slab weight is not exceeded;
- the slab weight is just large enough for producing the orders;
- the cost of this solution (the sum of the slab weights) is 41;

- a lower bound is the sum of order weights that is 39

In the next page you find the most common model from Antoine Gargani and Philippe Refalo An efficient model and strategy for the steel mill slab design problem Principles and Practice of Constraint Programming–CP 2007, 77–89, 2007.
A Minizinc implementation of the model described is:

```
% Some parameters omitted
array [Orders] of int: size;
array [Orders] of Colors: color;
array [0..maxCapa] of 0..maxCapa: slack = ... ;

% Variables:
array [Orders] of var Slabs: placedIn;

% Constraints:
array [Slabs] of var 0..maxCapa: load;
constraint bin_packing_load(load, placedIn, size);
array [Slabs] of var 0..2: nColors;
constraint forall(s in Slabs)(nColors[s] = ... );
% Objective:
var int: objective = sum(s in Slabs)(slack[load[s]]);
solve minimize objective;
```

A Gecode model is:

```
IntVarArray x[Orders](Slabs); // the slab in which the order is placed
IntVarArray l[Slabs](0..maxCap); // the load in each slab

multiknapsack(x,weight,l);
forall (s in Slabs)
  sum(c in Colors) (or(o in colorOrders[c])(x[o] == s)) <= 2;

cost(sum(s in Slabs) loss[l[s]])

branch(x)
```

However, in Stefan Heinz, Thomas Schlechte, Rüdiger Stephan, and Michael Winkler Solving steel mill slab design problems Constraints 17(1), 39–50, 2012 it is shown that MILP is by far more efficient in solving steel mill problems.
Further references:
- https://www.csplib.org/Problems/prob038/
- https://www.csplib.org/Problems/prob038/references/

3 The model

To solve the Steel Mill Slab Problem, this paper considers three constraint-based techniques (CP, LNS, CBLS), which are all based on the same high-level model, which captures the combinatorial substructures of this application. This section presents the common model, while the three following sections show how this model will be used differently by the various approaches. The model used in this work is based on Gargani and Refalo [3].

Let n be the number of orders. The color of order $o \in [1..n]$ is denoted by $color(o)$ and its size by $size(o)$. Let m be the number of available slabs (if there is no constraint on the number of available slabs we can set $m = n$ without restriction). Let $\mathcal{C} = \bigcup_{o \in [1..n]} \{color(o)\}$ be the set of colors. Let $\mathcal{O}_c$ be the set of orders with color c ($c \in \mathcal{C}$). For an order $o \in [1..n]$, X_o is a variable denoting the slab assigned to order o. The domain of X_o is the interval $[1..m]$ of slab indices.

The following expression indicates whether color c is used in slab i: $\bigvee_{o \in \mathcal{O}_c} (X_o = i)$. The constraint that at most two different colors are present in a slab is:

$$\forall i \in [1..m], \sum_{c \in \mathcal{C}} \left(\bigvee_{o \in \mathcal{O}_c} X_o = i \right) \leq 2, \tag{1}$$

in which a boolean expression has value 1 when true and 0 otherwise. For each slab $j \in [1..m]$, a load variable L_j indicates the total size of the orders assigned to this slab. These variables are linked to order variables with the set of constraints:

$$\forall j \in [1..m], L_j = \sum_{o \in [1..n] : (X_o = j)} size(o). \tag{2}$$

With each slab $j \in [1..m]$ comes a loss variable F_j (free space available in slab j). Since the objective is to minimize the available free space,[3] F_j is the minimum possible free space available in slab j. This minimum free space is obtained from L_j by choosing the smallest slab capacity larger or equal to L_j. The available slab capacities[4] are $capa = \{0, c_1, \ldots, c_p\}$ with $0 < c_1 < \ldots < c_p$. The possible free spaces are precomputed in an array $\mathbf{F}$ for all possible load values l: $\forall l \in [0..c_p]$, $\mathbf{F}[l] = \min\{c - l \mid c \in capa \ \wedge \ c \geq l\}$. This is well defined as we assume $c_p \geq \max_{o \in [1..n]} size(o)$. The free space of slab j is written with an element constraint as $F_j = \mathbf{F}[L_j]$. The objective to minimize is $\sum_{j \in [1..m]} F_j$ that is the total loss. Although there is no explicit variable telling the capacity chosen for a slab j, this can be easily deduced as $L_j + F_j$. The model also replaces the set of constraints given in (2) by the global packing constraint `Pack` which enables more propagation in CP [10] and more incrementality in CBLS.

[3] This is equivalent to minimizing the total steel produced as the consumed part is constant and equal to $\sum_{o \in [1..n]} size(o)$.

[4] We introduce the capacity of 0 to allow empty slabs.