# Topic 4: Modelling (for CP and LCG)<sup>1</sup> (Version of 20th August 2024)

#### Pierre Flener and Gustav Björdal

Optimisation Group Department of Information Technology Uppsala University Sweden

Course 1DL442: Combinatorial Optimisation and Constraint Programming, whose part 1 is Course 1DL451: Modelling for Combinatorial Optimisation

<sup>1</sup>Many thanks to Guido Tack for feedback

VEE



Viewpoints & Dummy Values

Implied Constraints

Redundant Variables & Channelling Constraints

Pre-Computation

- 1. Viewpoints & Dummy Values
- 2. Implied Constraints
- 3. Redundant Variables & Channelling Constraints



#### Viewpoints & Dummy Values

Implied Constraints

Redundant Variables & Channelling Constraints

Pre-Computation

## 1. Viewpoints & Dummy Values

2. Implied Constraints

3. Redundant Variables & Channelling Constraints



# Recap

#### Viewpoints & Dummy Values

Implied Constraints

Redundant Variables & Channelling Constraints

Pre-Computation

## 1 Modelling: express problem in terms of

- parameters,
- decision variables,
- constraints, and
- objective.
- 2 Solving: solve using a state-of-the-art solver.



## Example (Student Seating Problem)

#### Viewpoints & Dummy Values

Implied Constraints

Redundant Variables & Channelling Constraints

Pre-Computation



## Given:

- nStudents students,
- nPgms study programmes
- nChairs chairs around nTables tables, and
- Chairs[t] as the set of chairs of table t,

find a seating arrangement such that:

- each table has students of distinct study programmes;
- 2 each table has either at least half or none of its chairs occupied;
- a maximum number of student preferences on being seated at the same table are satisfied.

What are suitable decision variables for this problem?



A viewpoint is a choice of decision variables.

### Example (Student Seating Problem)

#### Viewpoint 1: Which chair does each student sit on?

```
Viewpoints &
Dummy Values
```

Implied Constraints

Redundant Variables & Channelling Constraints Pre-Computation

#### 1 % Chair[s] = the chair of student s: 2 array[1..nStudents] of var 1..nChairs: Chair; 3 constraint all\_different(Chair); % max 1 student per chair

## Viewpoint 2: Which student, if any, sits on each chair?

```
1 int: dummyS = 0; % Advice: also experiment with nStudents+1
2 set of int: StudentsAndDummy = 1..nStudents union {dummyS};
3 % Student[c] = the student, possibly dummy, sitting on chair c:
4 array[1..nChairs] of var StudentsAndDummy: Student;
5 constraint global_cardinality_closed(Student, [dummyS]++[i|i in 1..nStudents],
        [nChairs - nStudents] ++ [1 | i in 1..nStudents]);
        %all_different(Student) if nStudents+1..nChairs are dummy students
```

We revisit this problem at slide 19 and the choice of dummy values in Topic 5: Symmetry, as well as in Topic 8: Inference & Search in CP & LCG. Let us see how viewpoints differ when stating constraints.



There are n objects, s shapes, and c colours, with  $s \ge n$ . Assign a shape and a colour to each object such that:

- Viewpoints & Dummy Values
- Implied Constraints
- Redundant Variables & Channelling Constraints
- Pre-Computation

- 1 the objects have distinct shapes;
- 2 the numbers of objects of the actually used colours are distinct;
- other constraints, yielding NP-hardness and actually distinguishing the objects from the shapes, are satisfied.

This problem can be modelled from different viewpoints:

- 1 Which colour, if any, does each shape have?
- 2 Which shapes, if any, does each colour have?
- 3 Which shape and colour does each object have?
- 4 . . .

Each viewpoint comes with benefits and drawbacks.



#### Viewpoint 1: Which colour, if any, does each shape have?

```
1 int: n; % number of objects
          2 int: s; % number of shapes
          3 constraint assert(s >= n, "Not enough shapes");
Viewpoints &
Dummy Values
          4 int: c: % number of colours
          5 int: dummyColour = 0; % Advice: also experiment with c+1
Implied
Constraints
          6 set of int: ColoursAndDummy = 1..c union {dummyColour};
          7 % Colour[i] = the colour, possibly dummy, of the object of shape i:
Redundant
Variables &
          8 array[1..s] of var ColoursAndDummy: Colour;
Channelling
          9 % There are n objects:
Constraints
          10 constraint count(Colour,dummyColour) = s - n;
          11 % The numbers of objects of the actually used colours are distinct:
Computation
          12 constraint all_different_except(global_cardinality(Colour,1..c),{0});
          13 % The objects have distinct shapes:
               implied by lines 6 and 8!
          14 %
          15 % ... state here the other constraints ...
          16 solve satisfy:
```

So what are the shape and colour of a particular object?! Map the objects onto the shapes with non-dummy colour!

Pre-



#### Viewpoint 2: Which shapes, if any, does each colour have?

```
1 int: n; % number of objects
          2 int: s; % number of shapes
          3 constraint assert(s >= n, "Not enough shapes");
Viewpoints &
Dummy Values
          4 int: c: % number of colours
Implied
          5 %
Constraints
          6 %
          7 % Shapes[i] = the set of shapes of colour i:
Redundant
Variables &
          8 array[1..c] of var set of 1..s: Shapes;
Channelling
          9 % There are n objects:
Constraints
                implied by line 14 below!
          10 %
          11 % The numbers of objects of the actually used colours are distinct:
Computation
          12 constraint all_different_except([card(Shapes[colour]) | colour in 1..c],{0});
          13 % The objects have distinct shapes:
          14 constraint n = card(array_union(Shapes));
          15 % ... state here the other constraints ...
          16 solve satisfy:
```

Post-process: map the objects onto actually used shapes. Can we also model this viewpoint without set variables? reg Yes, see next slide!

Pre-



#### Viewpoint 2: Which shapes, if any, does each colour have?

```
1 int: n; % number of objects
          2 int: s; % number of shapes
Viewpoints &
          3 constraint assert(s >= n, "Not enough shapes");
Dummy Values
          4 int: c: % number of colours
          5 %
Implied
Constraints
          6 %
          7 % NbrObj[i,j] = the number of objects of colour i and shape j:
Redundant
Variables &
          8 array[1..c,1..s] of var 0..1: NbrObi;
Channelling
          9 % There are n objects:
Constraints
          10 constraint n = sum(NbrObj);
          11 % The numbers of objects of the actually used colours are distinct:
Computation
          12 constraint all_different_except([sum(NbrObj[colour,..]) | colour in 1..c],{0});
          13 % The objects have distinct shapes:
          14 constraint forall(shape in 1..s) (sum(NbrObj[.., shape]) <=1 );
          15 % ... state here the other constraints ...
          16 solve satisfy:
```

Which model for viewpoint 2 is clearer or better? R Ask others and try!

Pre-



Implied

Pro-

### Example (Objects, Shapes, and Colours)

#### Viewpoint 3: Which shape and colour does each object have?

```
1 int: n; % number of objects
          2 int: s; % number of shapes
Viewpoints &
          3 constraint assert(s >= n, "Not enough shapes");
Dummy Values
          4 int: c; % number of colours
          5 % Shape[i] = the shape of object i:
Constraints
          6 array[1..n] of var 1..s: Shape;
Redundant
          7 % Colour[i] = the colour of object i:
Variables &
          8 array[1..n] of var 1..c: Colour;
Channelling
          9 % There are n objects:
Constraints
               implied by lines 6 and 8!
          10 %
          11 % The numbers of objects of the actually used colours are distinct:
Computation
          12 constraint all different except(global cardinality closed(Colour, 1..c), {0});
          13 % The objects have distinct shapes:
          14 constraint all different(Shape);
          15 % ... state here the other constraints ...
          16 solve satisfy;
```

We needed to use two parallel arrays in lines 6 and 8 with the same index set but different domains in order to mimic records of two decision variables.

UPPSALA UNIVERSITET

Viewpoints & Dummy Values

Implied Constraints

Redundant Variables & Channelling Constraints

Pre-Computation Which viewpoint is better in terms of:

- Size of the search space:
  - Viewpoint 1:  $\mathcal{O}((c+1)^s)$ , which is independent of n
  - Viewpoint 2: O(2<sup>s·c</sup>), which is independent of n
  - Viewpoint 3: O(s<sup>n</sup> · c<sup>n</sup>)

Does this actually matter?

- Ease of formulating the constraints and the objective:
  - It depends on the unstated other constraints.
  - Ideally, we want a viewpoint that allows global constraints to be used.
- Performance:
  - Hard to tell: we have to run experiments!
- Readability:
  - Who is going to read the model?
  - What is their background?

There are no correct answers here:

we actually need to think about this and run experiments.



Viewpoints & Dummy Values

#### Implied Constraints

Redundant Variables & Channelling Constraints

Pre-Computation

#### 1. Viewpoints & Dummy Values

## 2. Implied Constraints

3. Redundant Variables & Channelling Constraints



## Example (Magic Series of length n: model C)

The element at index i in I = 0..(n-1) is the number of occurrences of i. Solutions: Magic=[1,2,1,0] and Magic=[2,0,2,0] for n=4.

Viewpoints & Dummy Values

#### Implied Constraints

Redundant Variables & Channelling Constraints

Pre-Computation **Decision variables:** Magic =  $\begin{array}{cccc} 0 & 1 & \cdots & n-1 \\ \hline \in I & \in I & \cdots & \in I \end{array}$ 

## **Problem Constraint:**

forall(i in I)(Magic[i] = sum(j in I)(Magic[j] = i))

or, logically equivalently but better:

forall(i in I)(Magic[i] = count(Magic,i))

or, logically equivalently and even better:

global\_cardinality\_closed(Magic, arrayld(I, [i | i in I]), Magic)
Implied Constraints:

COCP/M4CO 4



Viewpoints & Dummy Values

Implied

Constraints

Redundant

Channelling Constraints

Computation

Pre-

### Definition

An implied constraint, also called a redundant constraint, is a constraint that logically follows from other constraints.

#### Benefit:

Solving may be faster, without losing any solutions. However, not all implied constraints accelerate the solving. **Good practice in MiniZinc:** 

Flag implied constraints using implied\_constraint. This allows backends to handle them differently, if wanted (see Topic 9: Modelling for CBLS):

```
predicate implied_constraint(var bool: c) = c; VS
predicate implied_constraint(var bool: c) = true;
```

## Example

constraint implied\_constraint(sum(Magic) = n);

In Topic 5: Symmetry,

we see the equally recommended  $\verb"symmetry_breaking_constraint".$ 

COCP/M4CO 4



Viewpoints & Dummy Values

Implied Constraints

Redundant Variables & Channelling Constraints

Pre-Computation 1. Viewpoints & Dummy Values

**2. Implied Constraints** 

## 3. Redundant Variables & Channelling Constraints



### Example (n-queens)

Use both the  $n^2$  decision variables Queen [r, c] in 0..1 and the n decision variables Row [c] in 1..n.

## Definition

Dummy Values Implied Constraints

Viewpoints &

Redundant Variables & Channelling Constraints

Pre-Computation A redundant decision variable denotes information already denoted by other variables: mutual redundancy (same information) vs non-mutual redundancy.

**Benefit:** Easier modelling, or faster solving, or both. Careful, the terminology differs: derived parameters vs redundant variables.

## Examples (see Topic 6: Case Studies)

- Each Queen [.., c] slice is mutually redundant with the variable Row [c].
- Best model of Black-Hole Patience: mutual redundancy.
- Models 1 and 3 of Warehouse Location: non-mutual redundancy.
- Sport Scheduling: mutual redundancy.



#### Example (n-queens)

One-way channelling from each decision variable Row[c] to one of its mutually redundant decision variables of the slice Queen[..,c]: constraint forall(c in 1..n) (Queen[Row[c],c] = 1); What sets the other decision variables of the slice Queen[..,c]?

# Definition

Redundant Variables & Channelling Constraints

Constraints

Viewpoints &

Dummy Values

Pre-Computation A channelling constraint fixes the value of either some (1-way channelling) or all (2-way channelling) decision variables when the values of the decision variables they are redundant with are fixed. This applies to both sets of decision variables.

## Examples (see Topic 6: Case Studies)

- Best model of Black-Hole Patience: 2-way channelling.
- Models 1 and 3 of Warehouse Location: 1-way channelling.
- Sport Scheduling: 2-way channelling.



#### Viewpoints & Dummy Values Implied Constraints Redundant Variables & Cherentian

Channelling Constraints

Pre-Computation

## Example (Student Seating, viewpoint 2 revisited)

```
1 int: dummyS = 0; % Advice: also experiment with nStudents+1
2 set of int: StudentsAndDummy = 1..nStudents union {dummyS};
3 % Student[c] = the student, possibly dummy, sitting on chair c:
4 arrav[1..nChairs] of var StudentsAndDummy: Student;
5 constraint global_cardinality_closed(Student, [dummyS]++[i|i in 1..nStudents],
    [nChairs - nStudents] ++ [1 | i in 1..nStudents]);
6 int: dummyP = 0; % Advice: also experiment with nPgms+1
7 set of int: PgmsAndDummy = 1..nPgms union {dummyP};
8 % Pgm[s] = the given study programme of student s:
9 array[1..nStudents] of 1..nPgms: Pgm;
10 % Programme[c] = the programme of the student on chair c:
11 array[1..nChairs] of var PgmsAndDummy: Programme; % non-mut. red. w/ Student
12 % 1-way channelling from Student to Programme, in case dummyS = 0:
13 constraint forall(c in 1..nChairs)
    (Programme[c] = array1d(StudentsAndDummy, [dummyP] ++ Pgm)[Student[c]]);
14 % (1) Each table has students of distinct study programmes:
15 constraint forall (T in Chairs)
    (all_different_except([Programme[c] | c in T]), {dummyP});
16 ... % constraint (2) and objective (3) of slide 5
  Note that Student uniquely determines Programme via Pgm. but not
  vice-versa: one can also formulate (1) directly with Student via Pgm.
```



Viewpoints & Dummy Values

Implied Constraints

Redundant Variables & Channelling Constraints

Pre-Computation **1. Viewpoints & Dummy Values** 

**2. Implied Constraints** 

3. Redundant Variables & Channelling Constraints



#### Example (Prize-Pool Division)

Consider a maximisation problem where the objective function is the division of an unknown prize pool by an unknown number of winners:

```
Viewpoints &
Dummy Values
Implied
Constraints
```

```
Redundant
Variables &
Channelling
Constraints
```

Pre-Computation

```
1 ...
2 array[1..5] of int: Pools = [1000,5000,15000,20000,25000];
3 var 1..5: x; % index of the actual prize pool within Pools
4 var 1..500: nbrWinners; % the number of winners
5 constraint ... x ... nbrWinners ...;
6 solve maximize Pools[x] div nbrWinners; % implicit: element!
```

**Observation:** We should beware of using the div function on decision variables, because:

- It yields weak inference, at least in CP and LCG solvers.
- Its inference takes unnecessary time and memory.

Idea: We can precompute all possible objective values, as derived parameters.



#### Viewpoints & Dummy Values

Constraints Redundant Variables & Channelling Constraints

Pre-Computation

## Example (Prize-Pool Division, revisited)

Precompute a 2d array of derived parameters, indexed by 1..5 and 1..500, for each possible value pair of x and nbrWinners: 2 array[1..5] of int: Pools = [1000,5000,15000,20000,25000]; 3 var 1..5: x; % index of the actual prize pool within Pools 4 var 1..500: nbrWinners; % the number of winners 5 constraint ... x ... nbrWinners ...; 6 array[1..5,1..500] of int: ObjVal = array2d(1..5, 1..500, [Pools[p] div n | p in 1..5, n in 1..500]); % div on par!

7 solve maximize ObjVal[x,nbrWinners]; % implicit: 2d-element!

## Example (Kakuro Puzzle, reminder from Topic 3: Constraint Predicates)

We precomputed all\_different\_sum  $(X, \sigma)$  for  $|X| \in 2..7$  and  $\sigma \in 3..35$ , say table ([x,y], [|1,3|3,1|]) for all\_different\_sum ([x,y], 4) and table ([y,z], [|1,2|2,1|]) for all\_different\_sum ([y,z],3), because MiniZinc has no all\_different\_sum predicate and its definition by a conjunction of all\_different and sum has too poor inference.