Wine Cask Puzzle (math09)

—- Run LPL Code , HTML Document –

Problem: A farmer leaves 45 casks of wine, of which 9 each are full, three-quarters full, half full, one quarter full and empty. His five nephews want to divide the wine and the casks, without pouring wine from cask to cask, in such a way that each receives the same amount of wine and the same number of casks, and further so that each receives at least one of each kind of casks, and not two of them receive the same number of every kind of casks (see [2] and [1]).

Modeling Steps

First, we list the nephews n (there are 5), and the type of casks c (there are 5) and define them as sets.

1. For each type of casks, we know the percentage ($0 \leq h_c \leq 1$) representing the filling level.
2. We would like to know how many casks of each type each nephew receives. We introduce an integer variable for each (nephew,cask)-combination ($x_{n,c}$), which results in 25 variables. Each variable has a lower bound of 1, since each nephew receives at least one of each type, and it has an upper bound of 5, because each nephew receives 9 (see next point) casks but from each at least one, so he cannot receive more than 5 from each cask: $1 \leq x_{n,c} \leq 5$.
3. Each nephew must receive the same number of casks – there are 45 casks and 5 nephews, so each of them receives 9. Hence, $\sum_c x_{n,c} = 9$.
4. There are exactly 9 casks of each type c, and they must be distributed among the five nephews: $\sum_n x_{n,c} = 9$.
5. Each nephew must receive the same quantity of wine. There is $(1+0.75+0.5+0.25)\cdot 9 = 22.5$ units of wine. Therefore, each nephew receives $22.5/5 = 4.5$ units. This gives the following constraint: $\sum_c h_c x_{n.c} = 4.5$.
6. The last requirement is: "no two of them receive the same number of every kind of casks". How can we model this requirement? An idea is the following: Assign a 5-digit number y_n to each nephew, that identifies the number $x_{n,c}$ of casks given to a nephew in a unique way. We cannot just add or multiply the number of cask for a nephew, because then two nephew might get the *same* number. One way is to interpret the 5 digits $x_{n,1}, x_{n,2}, \ldots, x_{n,5}$ as 5 digits in a 5-digit number. Hence $10^4 x_{n,1} + 10^3 x_{n,2} + \ldots + 10^0 x_{n,5}$ would be a unique identifier for an assignment of casks to a nephew. For example the number $y_1 = 23112$ means, that the first nephew receives 2 full, 3 three-quarter-full, 1 half-full, 1 a-quarter-full, and 2 empty casks.
7. Now we could reformulate the requirements: The numbers y_n must all be different from each other: for each pair (n, m) of nephew we have $y_n \neq y_m$.
8. The $\neq$-operation in a constraint is difficult to handle. We would need to introduce additional variables to resolve such a constraints. In our case, however, it is sufficient to order the variables and to impose

$$y_{n-1} > y_n \quad \text{forall } n \in \{1, \ldots, 4\}$$

Certainly, then all y_n are different from each other[1].

9. Furthermore we have as explained above:

$$y_n = \sum_c 10^{5-c} x_{n,c}, \qquad \text{forall } n$$

10. We are only looking for feasible solutions, or in other words, there is nothing to optimize.

Listing 1: The Complete Model implemented in LPL [4]

```
model math09 "Wine Cask Puzzle";
  set
    n,m:=[nephew1 nephew2 nephew3 nephew4 nephew5];
    c:=['Empty' '1/4'  '1/2'  '3/4'  'Full'];
  parameter
    h{c} := [0 0.25 0.5 0.75 1];
    p{c}  := 10^(#c-c); //[10000,1000,100,10,1];
  integer variable
    x{n,c} [1..9] "number of casks";
    y{n} "5-digit assigned to each nephew";
  constraint
    A{n}: sum{c} x = 9       "Each nephew gets 9 casks";
    B{c}: sum{n} x = 9       "Same number of casks";
    C{n}: sum{c} h*x = 4.5 "Same amount of wine";
    D{n}: sum{c} p*x = y         "A 5-digit for each n";
    E{n|n>1}: y[n-1] > y[n]    "Each different";
  solve;
  Writep(x);
end
```

A solution is given in the following table.

	Empty	1/4	1/2	3/4	Full
nephew1	3	1	1	1	3
nephew2	2	2	1	2	2
nephew3	2	1	2	3	1
nephew4	1	3	2	1	2
nephew5	1	2	3	2	1

Table 1: Solution of the Win Cask Puzzle

Further Comments: The last requirement was formulated by a 'trick': The variables y_n are ordered in a strictly ascending way. Then they are certainly different from each other. This is a efficient way to avoid the $\neq$-operation. In some situations we cannot avoid it. In LPL, there is a convenient way of formulating the constraint that several variables must be different from each other using the `Alldiff` function. Hence, one could alternatively formulate the constraint as follows:

```
E{n}: Alldiff( {m} y[m] );
```

[1] Note the operation $A > B$ has the same meaning as $A - 1 \geq B$ since all numbers are integer

However, note that the other formulation is more efficient.

We also introduced a 5-digit number to identify the different nephews. Could we model this requirement in another – maybe more straightforward – way without using additional variables y_n? There are various possibilities. Certainly, we might formulate the requirement as a logical condition: "For each pair (m, n) of nephew , at least 1 (out of all c) $x_{m,c}$ must be different from $x_{n,c}$". This can be formulated in LPL directly as follows:

```
constraint E{m,n|m<n}: atleast(1){c} (x[n,c]<>x[m,c]);
```

From the logical point of view, "at least one" is exactly the same as "or". Hence in LPL the constraint can also be written as:

```
constraint E{m,n|m<n}: or{c} (x[n,c]<>x[m,c]);
```

This constraint replaces both constraint `D` and `E` in the original model. A complete model code is (see math09a[2]) :

```
model math09a "Wine Cask Puzzle (II)";
  set n,m:=[nephew1 nephew2 nephew3 nephew4 nephew5];
      c:=['Empty' '1/4'  '1/2'  '3/4'  'Full'];
  parameter  h{c} := [0 0.25 0.5 0.75 1];
  integer variable x{n,c} [1..5];
  constraint
    A{n}: sum{c} x = 9;
    B{c}: sum{n} x = 9;
    C{n}: sum{c} h*x = 4.5;
    F{n,m|n>m}: or{c} (x[n,c]<>x[m,c]);
  solve;
  Writep(x);
end
```

This model is more concise and straightforward for the modeler to formulate all the requirements. Internally, LPL will automatically translate this into additional constraints and variables to make the model linear. We neither need to declare the parameter p_n nor the variables y_n, and the two constraints `D` and `E` are not needed. However, it is worth to mention again a note of warning: This concise formulation – although in this context the best choice – might have its price: The automatical translation of LPL cannot fully exploit the 'trick' used above, and the resulting model that is generated by LPL may be somewhat less efficient from the solving point of view. There is no noteworthy difference in this model, however in larger models the difference may be important. The user, of course, has always the possibility to inspect the translation steps of LPL and to act correspondingly.

Questions

1. What happens if one has forgotten to formulate the constraint `B`, that there are an equal number of casks of each type?

2. One could have interpreted the last requirement "not two of them receive the same number of every kind of casks" in a more restrictive way: "Each nephew receives a different number of casks *of each type*", hence all x_n, c must be different from each other. Would this be a plausible interpretation?

3. Still another interpretation of "not two of them receive the same number of every kind of casks" would be "looking at a particular type of casks, then each nephew must receive

[2]https://lpl.matmod.ch/lpl/Solver.jsp?name=/math09a

a different number of casks". What is the solution now? Formulate it by adding new constraints.

Answers

1. Each nephew still receives 9 casks, which totals to 45 casks. But there are not necessarily 9 empty casks anymore, for example, to distribute. (Check this by removing the corresponding constraint.)

2. Of course, one could interpret it in this way. But a little reasoning shows that this cannot be the meaning of the requirements, since then no solution exists. Why? Suppose that all $x_{n,c}$ must be different from each other. Since there are 25 variables, 25 different integer values must be assigned to $x_{n,c}$. Say, 1 to 25. But this is impossible.

3. The same reasoning is valid as for answer 2. Suppose nephew 1 receives one full cask, then nephew 2 must receive another number of full casks. Since there are 5 nephews, 5 different numbers of casks must be assigned to the different nephews: say 1 to 5 in any permutation. Since the full casks must sum up to 9, this is not possible, since the five smallest numbers already sum up to $5 + 4 + 3 + 2 + 1 = 15$. In any case, one could formulate the requirement in LPL by adding the following constraint:

```
constraint G{c}: Alldiff({n} x);
```

References

[1] Nolan Clare. http://www.chlond.demon.co.uk/academic/puzzles.html.

[2] Kraitchik M. *Mathematical Recreations*. W.W. Norton and Company, 1942.

[3] MatMod. Homepage for Learning Mathematical Modeling : https://matmod.ch.

[4] Hürlimann T. Reference Manual for the LPL Modeling Language, most recent version. https://matmod.ch/lpl/doc/manual.pdf.