

# CS 524: Introduction to Optimization

## Lecture 22 : Constraint logic extensions

Michael Ferris

Computer Sciences Department  
University of Wisconsin-Madison

October 25 , 2023

# Constraint Logic Programming

Binary variables  $\delta_i$  represent statements  $P_i$  via the following construction:

$$\delta_i = \begin{cases} 1 & \text{if statement } P_i \text{ is true} \\ 0 & \text{if statement } P_i \text{ is false} \end{cases}$$

$P_i$  could be “do project  $i$ ” or “ $f(x) \leq 0$ ”

$\delta_i$  is an indicator variable for whether the statement is true or false.

Standard boolean algebra notation for connectives between statements:

$\vee$  means ‘or’

$\wedge$  means ‘and’

$\neg$  means ‘not’

$\rightarrow$  means ‘implies’

$\leftrightarrow$  means ‘if and only if’

$\underline{\vee}$  means ‘exclusive or’

Other connectives such as “nor” or “nand” are also used in the literature.

# Equivalences between CLP and MIP

- Next slides detail standard ways to equivalently express statement logic in terms of constraints on the corresponding indicator variables in a MIP.
- The examples shown in the table are useful in building models since they construct a tight approximation of the logic typically, even when the solution algorithm used to solve the MIP relaxes some of the variables to be continuous (i.e. in  $[0, 1]$  instead of being in  $\{0, 1\}$ ).
- Note that we add some slides (More details, definition of  $y$  variables representing other modeling constructs) that are for information only here. Please keep these only for future reference!

	Statement	MIP Constraint
1.	$\neg P_1$	$\delta_1 = 0$
2.	$P_1 \vee P_2$	$\delta_1 + \delta_2 \geq 1$
3.	$P_1 \underline{\vee} P_2$	$\delta_1 + \delta_2 = 1$
4.	$P_1 \wedge P_2$	$\delta_1 = 1, \delta_2 = 1$
5.	$\neg(P_1 \vee P_2)$	$\delta_1 = 0, \delta_2 = 0$
6.	$P_1 \rightarrow P_2$	$\delta_1 \leq \delta_2$
7.	$P_1 \rightarrow (\neg P_2)$	$\delta_1 + \delta_2 \leq 1$
8.	$P_1 \leftrightarrow P_2$	$\delta_1 = \delta_2$
9.	$P_1 \rightarrow (P_2 \wedge P_3)$	$\delta_1 \leq \delta_2, \delta_1 \leq \delta_3$
10.	$P_1 \rightarrow (P_2 \vee P_3)$	$\delta_1 \leq \delta_2 + \delta_3$
11.	$(P_1 \wedge P_2) \rightarrow P_3$	$\delta_1 + \delta_2 \leq 1 + \delta_3$
12.	$(P_1 \vee P_2) \rightarrow P_3$	$\delta_1 \leq \delta_3, \delta_2 \leq \delta_3$
13.	$P_1 \wedge (P_2 \vee P_3)$	$\delta_1 = 1, \delta_2 + \delta_3 \geq 1$
14.	$P_1 \vee (P_2 \wedge P_3)$	$\delta_1 + \delta_2 \geq 1, \delta_1 + \delta_3 \geq 1$

## Example: `pitcher.gms`

Consider the draft day statement: “if DE and ST are signed, then BS cannot be signed.”

- Let  $P_{DE}$  mean “sign DE” and represent this using an indicator  $\delta_{DE}$  with  $\delta_{DE} = 1$  iff  $P_{DE}$  is true
- Similarly for  $P_{ST}$  and  $\delta_{ST}$  and  $P_{BS}$  and  $\delta_{BS}$
- The statement “DE and ST are signed” is the statement  $P_{DE} \wedge P_{ST}$  is true, while “BS is not signed” can be expressed as  $\neg P_{BS}$  is true.
- The draft statement “if DE and ST are signed, then BS cannot be signed” is thus equivalent to the statement  $(P_{DE} \wedge P_{ST}) \rightarrow (\neg P_{BS})$  being true.
- The table above (line 11) expresses the truth of this statement by simply imposing the constraint:

$$\delta_{DE} + \delta_{ST} \leq 1 + (1 - \delta_{BS})$$

Note that  $1 - \delta_{BS}$  represents  $\neg P_{BS}$ .

# Logical AND and OR

- Statements 2. and 4. of the table above allow forcing of AND or OR
- If we wish to have a new binary variable  $\delta_n$  represent the AND or OR condition then we need the following if and only if statements

$$P_n \leftrightarrow (P_1 \wedge \cdots \wedge P_k) \quad \delta_n + k \geq 1 + \sum_{i=1}^k \delta_i, \delta_j \geq \delta_n, j = 1, \dots, k$$

(or equivalently)  $\delta_n = \min(\delta_1, \dots, \delta_k)$

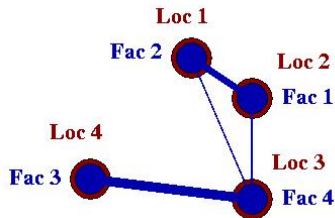
(or equivalently)  $\delta_n = \delta_1 \times \delta_2 \times \cdots \times \delta_k$

$$P_n \leftrightarrow (P_1 \vee \cdots \vee P_k) \quad \sum_{i=1}^k \delta_i \geq \delta_n, \delta_n \geq \delta_j, j = 1, \dots, k$$

(or equivalently)  $\delta_n = \max(\delta_1, \dots, \delta_k)$

Note: It's *not* necessary to make  $\delta_n$  a binary variable! Thus cut down on the number of binary variables in the model. An upper bound of 1 on  $\delta_n$  can be added instead.

# QAP – The second most famous problem



## “Quadratic” Assignment Problem

- Set of facilities  $F$
- Set of locations  $L$
- $d_{ij}$  distance from  $i \in L$  to  $j \in L$
- $f_{kl}$  flow from  $k \in F$  to  $l \in F$

$$x_{kj} = \begin{cases} 1 & \text{assign facility } k \text{ to location } j \\ 0 & \text{Otherwise} \end{cases}$$

$$\min \sum_{k \in F} \sum_{i \in L} \sum_{l \in F} \sum_{j \in L} d_{ij} f_{kl} x_{ki} x_{lj}$$

$$\sum_{k \in F} x_{kj} = 1 \quad \forall j \in L$$

$$\sum_{j \in L} x_{kj} = 1 \quad \forall k \in F$$

$$x_{kj} \in \{0, 1\} \quad \forall k \in F, \forall j \in L$$



- $x_{ki}x_{lj}$  is **nonlinear**!
- What you **really** want it is to count  $d_{ij}f_{kl}$  towards your objective **if and only if** you assign facility  $k \rightarrow i$  and facility  $l \rightarrow j$
- There is one more “trick” you should know – when you are multiplying two binary variables, that you may have missed above!

# A Final Trick

## Modeling Trick: Linearizing product of two binaries

$$z_{kilj} = 1 \Leftrightarrow x_{ki} = 1, x_{lj} = 1 \Leftrightarrow x_{ki} \wedge x_{lj}$$

$$z_{kilj} \leq x_{ki} \quad \forall k \in F, i \in L, l \in F, j \in L$$

$$z_{kilj} \leq x_{lj} \quad \forall k \in F, i \in L, l \in F, j \in L$$

$$z_{kilj} \geq x_{ki} + x_{lj} - 1 \quad \forall k \in F, i \in L, l \in F, j \in L$$

- This is a special case of  $P_{z_{kilj}} \Leftrightarrow (P_{x_{ki}} \wedge P_{x_{lj}})$

$$z_{kilj} + 2 \geq 1 + x_{ki} + x_{lj}, x_{ki} \geq z_{kilj}, x_{lj} \geq z_{kilj}$$

- We can also do it using (turn constraint on)  $z_{kilj} = 1 \Rightarrow x_{ki} + x_{lj} \geq 2$

$$-x_{ki} - x_{lj} + 2 \leq 2(1 - z_{kilj}), \quad \text{i.e. } x_{ki} + x_{lj} - 2z_{kilj} \geq 0$$

$y = x\delta$  for continuous  $x$ , binary  $\delta$

- Must know lower and upper bounds  $L \leq x \leq U$
- MIP formulation:

$$\begin{aligned} L\delta &\leq y \leq U\delta \\ L(1 - \delta) &\leq x - y \leq U(1 - \delta) \end{aligned}$$

$y = \min(x_1, x_2)$  for continuous variables  $x_1, x_2$

- Must know lower and upper bounds  $L_i \leq x_i \leq U_i$
- Introduce binary variables  $\delta_1, \delta_2$  to mean

$$\delta_i = \begin{cases} 1 & \text{if } x_i \text{ is the minimum value} \\ 0 & \text{otherwise} \end{cases}$$

- MIP formulation:

$$L_i \leq x_i \leq U_i$$

$$y \leq x_i$$

$$y \geq x_1 - (U_1 - L_{\min})(1 - \delta_1)$$

$$y \geq x_2 - (U_2 - L_{\min})(1 - \delta_2)$$

$$1 = \delta_1 + \delta_2$$

- Obvious extension to minimum of  $n$  variables

$y = \max(x_1, x_2, \dots, x_n)$  for continuous variables  $x_1, \dots, x_n$

- Must know lower and upper bounds  $L_i \leq x_i \leq U_i$
- Introduce binary variables  $\delta_i$ ,  $i = 1, \dots, n$  to mean

$$\delta_i = \begin{cases} 1 & \text{if } x_i \text{ is the maximum value} \\ 0 & \text{otherwise} \end{cases}$$

- MIP formulation:

$$L_i \leq x_i \leq U_i$$

$$y \geq x_i$$

$$y \leq x_i + (U_{\max} - L_i)(1 - \delta_i)$$

$$1 = \sum_i \delta_i$$

$y = |x_1 - x_2|$  for continuous variables  $x_1, x_2$

- Must know lower and upper bounds  $0 \leq x_i \leq U$
- Introduce binary variables  $\delta_1, \delta_2$  to mean

$$\delta_i = \begin{cases} 1 & \text{if } x_i - x_j \text{ is the positive value } (j \neq i) \\ 0 & \text{otherwise} \end{cases}$$

- MIP formulation:

$$0 \leq x_i \leq U$$

$$0 \leq y - (x_1 - x_2) \leq 2U\delta_2$$

$$0 \leq y - (x_2 - x_1) \leq 2U\delta_1$$

$$1 = \delta_1 + \delta_2$$

# Extensions

- Also can turn on/off constraints using the model construct: indicator constraints
- Other operators like “before”, “last” or “notequal”, “allDifferent” are often allowed in constraint logic programming (CLP) languages
- There is a growing literature on how to reformulate some of these within a MIP code and lots of specialized codes that treat these constraints explicitly
- Merging these two techniques (MIP and CLP) is an active area of research
- The techniques used in CLP are essentially clever ways to do complete enumeration very efficiently and quickly.