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 δ_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 "$f(x) \leq 0$"

δ_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'
$\veebar$	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 on being in $\{0,1\}$).
- Note that we add some slides (More details, definition of y variables representing other modeing 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.	$P_1 \veebar P_2$	$\delta_1 + \delta_2 = 1$
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 δ_{DE} with $\delta_{DE} = 1$ iff P_{DE} is true
- Similarly for P_{ST} and δ_{ST} and P_{BS} and δ_{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 $\neg P_{BS}$.

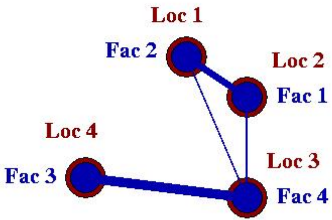
Logical AND and OR

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

$$
\begin{array}{ll}
P_n \leftrightarrow (P_1 \wedge \cdots \wedge P_k) & \delta_n + k \geq 1 + \sum_{i=1}^{k} \delta_i,\ \delta_j \geq \delta_n,\ j = 1, \ldots, k \\
 & \text{(or equivalently) } \delta_n = \min(\delta_1, \ldots, \delta_k) \\
 & \text{(or equivalently) } \delta_n = \delta_1 \times \delta_2 \times \cdots \times \delta_k \\
P_n \leftrightarrow (P_1 \vee \cdots \vee P_k) & \sum_{i=1}^{k} \delta_i \geq \delta_n,\ \delta_n \geq \delta_j,\ j = 1, \ldots, k \\
 & \text{(or equivalently) } \delta_n = \max(\delta_1, \ldots, \delta_k)
\end{array}
$$

Note: It's *not* necessary to make δ_n a binary variable! Thus cut down on the number of binary variables in the model. An upper bound of 1 on δ_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 $l \in F$

QAP

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

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

$$\begin{aligned} \sum_{i \in F} x_{ij} &= 1 \; \forall j \in L \\ \sum_{j \in L} x_{ij} &= 1 \; \forall i \in F \\ x_{ij} &\in \{0, 1\} \; \forall i \in F, \forall j \in L \end{aligned}$$

QAP

- $x_{ki}x_{lj}$ is nonlinear!
- What you really want it is to count $d_{ij}f_{kl}$ towards your objective if and only if you assign facility $f \to i$ and facility $k \to j$
- 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

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

$$\begin{aligned} z_{kilj} &\leq x_{ki} \;\forall k \in F, i \in L, l \in F, j \in L \\ z_{kilj} &\leq x_{lj} \;\forall k \in F, i \in L, l \in F, j \in L \\ z_{kilj} &\geq x_{ki} + x_{lj} - 1 \;\forall k \in F, i \in L, l \in F, j \in L \end{aligned}$$

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

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

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

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

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

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

$$L\delta \leq y \leq U\delta$$
$$L(1-\delta) \leq x - y \leq U(1-\delta)$$

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

- Must know lower and upper bounds $L_i \leq x_i \leq U_i$
- Introduce binary variables δ_1, δ_2 to mean

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

- MIP formulation:

$$\begin{aligned} L_i &\leq x_i \leq U_i \\ y &\leq x_i \\ y &\geq x_1 - (U_1 - L_{min})(1 - \delta_1) \\ y &\geq x_2 - (U_2 - L_{min})(1 - \delta_2) \\ 1 &= \delta_1 + \delta_2 \end{aligned}$$

- Obvious extension to minimum of n variables

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

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

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

- MIP formulation:

$$\begin{aligned} L_i &\leq x_i \leq U_i \\ y &\geq x_i \\ y &\leq x_i + (U_{max} - L_i)(1 - \delta_i) \\ 1 &= \sum_i \delta_i \end{aligned}$$

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

- Must know lower and upper bounds $0 \leq x_i \leq U$
- Introduce binary variables δ_1, δ_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:

$$\begin{aligned} &0 \leq x_i \leq U \\ &0 \leq y - (x_1 - x_2) \leq 2U\delta_2 \\ &0 \leq y - (x_2 - x_1) \leq 2U\delta_1 \\ &1 = \delta_1 + \delta_2 \end{aligned}$$

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.