Recent Developments in Disjunctive Programming

Aldo Vecchietti[(*)] and Ignacio E. Grossmann[(**)]

[(*)] **INGAR – Instituto de Desarrollo y Diseño**
Universidad Tecnologica Nacional – Santa Fe – Argentina
e-mail: aldovec@alpha.arcride.edu.ar

[(**)] **Department of Chemical Engineering, Carnegie Mellon University,**
Pittsburgh, PA 15213, - USA
e-mail: grossmann@cmu.edu

Introduction

Accepted Formulations

Formulations MILP y MINLP are the most used in the academia and companies to solve problems with discrete decisions

Algorithms for MINLP

- **Outer Approximation (OA) Dicopt ++**
- **Branch & Bound (B&B)**
- **Generalizaed Benders Decomposition (GBD)**
- **Extended Cutting Plane (ECP)**

Modelling Environments

Modeling environments used for posing and solving a MINLP problem: GAMS, AMPL, other

Alternative Formulation to MINLP

Hybrid Formulation
(Disjunctions and 0-1 variables)

$$\min \quad Z = \Sigma_k c_k + f(x) + d^T y$$

s.t.:

$$g(x) \le 0$$
$$r(x) + Dy \le 0$$
$$Ay \ge a$$

$$\bigvee_{i \in D_k} \begin{bmatrix} Y_{ik} \\ h_{ik}(x) \le 0 \\ c_k = \gamma_{ik} \end{bmatrix} \quad k \in SD$$

$$\Omega(Y) = True$$
$$x \in R^n, \ y \in \{0,1\}^q$$
$$Y \in \{True, False\}^m, \ c_k \ge 0$$

- ❑ x c_k are continuous variables
- ❑ y (0-1) are variables
- ❑ Y_{ik} are Boolean variables to establish whether the disjunctive term is true or not
- ❑ $f(x)$ objective function
- ❑ $d^T y$ linear cost terms
- ❑ $g(x)$ constraints that are independent of the discrete decisions
- ❑ $r(x)+Dy \le 0$ mixed-integer constraints
- ❑ $Ay \ge a$ integer constraints
- ❑ $\Omega(Y)$ propositional logic relating Boolean variables

Disjunctive Formulation

From Hybrid formulation

$$\min \quad Z = \Sigma_k c_k + f(x) \quad + d^T y$$

s. t.:

$$g(x) \le 0$$

$$r(x) + Dy \le 0$$

$$Ay \ge a$$

$$\bigvee_{i \in D_k} \begin{bmatrix} Y_{ik} \\ h_{ik}(x) \le 0 \\ c_k = \gamma_{ik} \end{bmatrix} \quad k \in SD$$

$$\Omega(Y) = True$$

$$x \in R^n, \quad y \in \{0,1\}^q \quad , Y \in \{True, False\}^m, \; c_k \ge 0$$

MINLP formulation

From Hybrid formulation

$$\min \quad Z = \sum_k c_k + f(x) + d^T y$$

s.t.:

$$g(x) \le 0$$

$$r(x) + Dy \le 0$$

$$Ay \ge a$$

$$\bigvee_{i \in D_k} \begin{bmatrix} Y_{ik} \\ h_{ik}(x) \le 0 \\ c_k = \gamma_{ik} \end{bmatrix} \quad k \in SD$$

$$\Omega(Y) = True$$

$$x \in R^n, \ y \in \{0,1\}^q$$

$$Y \in \{True, False\}^m, \ c_k \ge 0$$

Relaxations for a disjunctive set

Big-M relaxation

Linear case

$$F = \vee_{i \in D} \left[a_i^T x \le b_i \right] \quad x \in R^n$$

$$a_i^T x \le b_i + M_i (1 - y_i)$$

$$_{i} \; y_i = 1$$

$$M_i = max\{ a_i^T x - b_i \mid x^{lo} \le x \le x^{up} \}$$

Non-linear case

$$F = \vee_{i \in D} \left[h_i(x) \le 0 \right] \quad x \in R^n$$

$$h_i(x) \le M_i(1 - y_i)$$

$$_{i} \; y_i = 1$$

$$M_i = max\{ h_i(x) \mid x^{lo} \le x \le x^{up} \}$$

Relaxations for a disjunctive set

Convex Hull

Linear case

(Balas, 1985)

$$F = \underset{i \in D}{\vee} \left[a_i^T x \leq b_i \right] \quad x \in R^n$$

$$x - \underset{i \in D}{} v_i = 0 \qquad x, v_i \in R^n$$

$$a_i^T v_i - b_i y_i \leq 0$$

$$\underset{i \in D}{} y_i = 1 \quad , 0 \leq y_i \leq 1, \ i \in D$$

$$0 \leq v_i \leq v_i^{up} y_i$$

Non linear case

(Lee y Grossmann, 1999)

$$F = \underset{i \in D}{\vee} \left[h_i(x) \leq 0 \right] \quad x \in R^n$$

$$x - \underset{i \in D}{} v_i = 0 \qquad x, v_i \in R^n$$

$$y_i \, h_i(v_i / y_i) \leq 0$$

$$\underset{i \in D}{} y_i = 1 \quad , 0 \leq y_i \leq 1, \ i \in D$$

$$0 \leq v^i \leq v_i^{up} y_i$$

Relaxations for a disjunctive set

Properties

Increased number of variables and constraints		
	-	Big-M
	↓	
	+	Convex Hull

The **Big-M relaxation** of a disjunctive set implies a **mixed-integer formulation** of the discrete decision

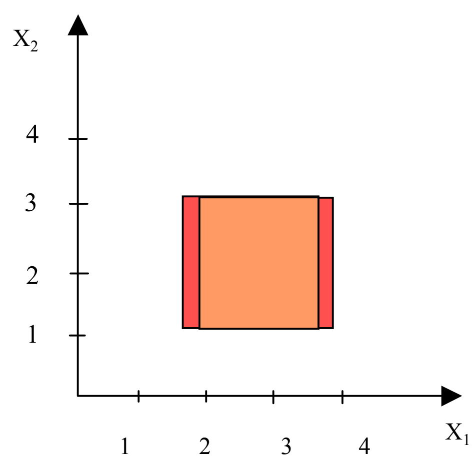
The **Convex-Hull relaxation** is related to a **disjunctive formulation** of the discrete decision

Property

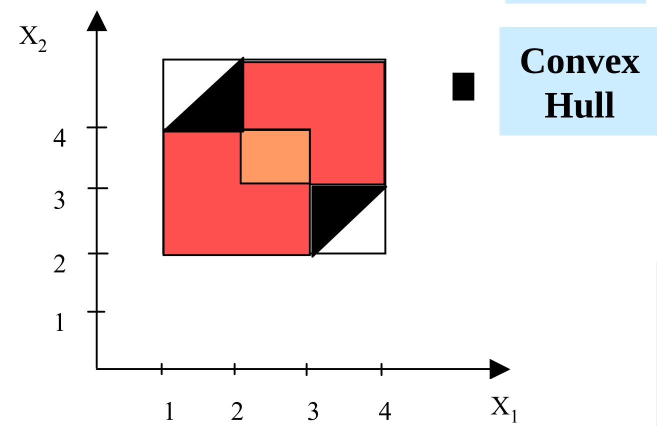
The **Convex-Hull** of a disjunctive set provides a relaxation that is **tighter or equal** to the **Big-M** model

Covex Hull vs. Big-M

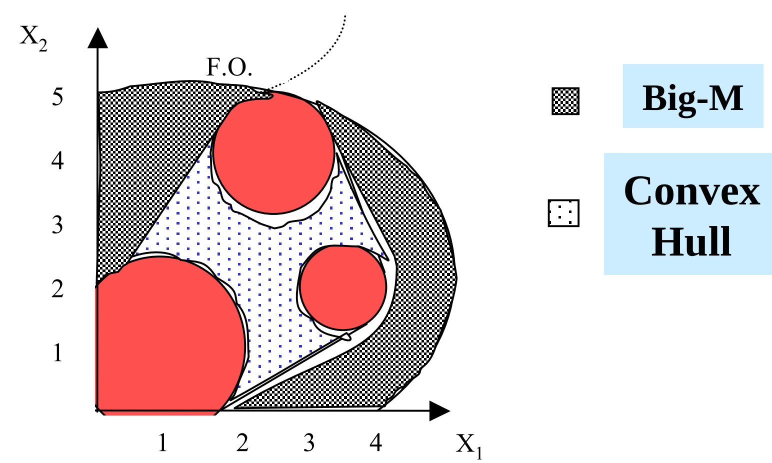
Illustrative examples



Both relaxations can be equivalent



The Convex Hull is tightest



Language for disjunctions and logic propositions

Operators, Operands and Sentences

- **Booleans Variables:** true and false values, binaries can be used instead
- **Logic Operators:** ∧ (“and”), ∨ (“or”), ⊻ (“exclusive or”), ~(“not”,!,¬), ->(implication),<->(equivalence)
- **Selection Sentences** (conditionals) for expressing disjunctions. We propose **IF..THEN..ELSE..ENDIF** sentences.
- **Special sentences** to facilitate the expressions of logic constraints Sentences like : atmost, exactly or adjacent.

Language for the expression of disjunctions

Two terms disjunction

$$\begin{bmatrix} \text{True} \\ \text{Constraints set 1} \end{bmatrix} \vee \begin{bmatrix} \text{False} \\ \text{Constraints set 2} \end{bmatrix} \Rightarrow$$

IF *(logic expression)* **THEN**

Apply constraints set 1

ELSE

Apply constraints set 2

ENDIF

$$\begin{bmatrix} Y_j \\ vt_j \geq log(2S^*_{ij}) + b_{i,j+1} \\ vt_j \geq log(2S^*_{ij}) + b_{i,j} \\ b_{ij} - b_{i,j+1} \leq \phi \\ b_{ij} - b_{i,j+1} \geq -\phi \end{bmatrix} \vee \begin{bmatrix} \neg Y_j \\ vt_j = 0 \\ b_{ij} - b_{i,j+1} = 0 \end{bmatrix} \Rightarrow$$

IF *(Y_j)* **THEN**

$$t_j \geq log(2S^*_{ij}) + b_{i,j+1}$$

$$vt_j \geq log(2S^*_{ij}) + b_{i,j}$$

$$b_{ij} - b_{i,j+1} \leq \phi$$

$$b_{ij} - b_{i,j+1} \geq -\phi$$

ELSE

$$vt_j = 0$$

$$b_{ij} - b_{i,j+1} = 0$$

ENDIF

Language for the expression of disjunctions

Several terms disjunctions

$$\begin{pmatrix} \textbf{1} \\ \textbf{Constraints set 1} \end{pmatrix} \vee \begin{pmatrix} \textbf{2} \\ \textbf{Constraints set 2} \end{pmatrix} \vee \cdots \vee \begin{pmatrix} \textbf{N} \\ \textbf{Constraints set N} \end{pmatrix}$$

IF (*Logic expression 1*) THEN
Apply constraints set 1 when logic expression 1 is true
ELSE IF (*Logic expression 2*) THEN
Apply constraints set 2 when logic expression 2 is true
ELSE IF (*Logic expression 3*) THEN
...
ELSE IF (*Logic expression N*) THEN
Apply constraints set N when logic expression N is true
ENDIF

Language for the expression of disjunctions

Example: Several terms disjunctions

$$\begin{bmatrix} Y_1 \\ (x_1-4)^2+(x_2-2)^2 \le 0.5 \end{bmatrix} \vee \begin{bmatrix} Y_2 \\ (x_1-3)^2+(x_2-4)^2 \le 1 \end{bmatrix} \vee \begin{bmatrix} Y_3 \\ (x_1-1)^2+(x_2-1)^2 \le 1.5 \end{bmatrix}$$

IF *(Y_1)* **THEN**

$$(x_1-4)^2+(x_2-2)^2 \le 0.5$$

ELSE IF *(Y_2)* **THEN**

$$(x_1-3)^2+(x_2-4)^2 \le 1$$

ELSE IF *(Y_2)* **THEN**

$$(x_1-1)^2+(x_2-1)^2 \le 1.5$$

ENDIF

Language for the expression of logic propositions

Logic operators : *$\wedge$ ("and"), $\vee$ ("or"), $\underline{\vee}$ ("exclusive or"), ~("not",!,$\neg$), ->(implication) , <->(equivalence)* **for the expression of relationships between the discrete variables**

Example:

$$Y_8 \rightarrow Y_3 \vee Y_5 \vee (\neg Y_3 \wedge \neg Y_5)$$
$$Y_1 \underline{\vee} Y_2$$
$$Y_4 \underline{\vee} Y_5$$
$$Y_6 \underline{\vee} Y_7$$

The modeler formulates these relationships that are translated into integer constraints

Language for the expression of logic constraints

Special sentences

atmost *(parameter list)*

Atmost one component of the parameter list must be true

atleast *(parameter list)*

Atleast one component of the parameter list must be true

exactly *(parameter list)*

Only one component of the parameter list must be true

adjacent *(lista de argumentos)*

Two adjacent components of the parameter list must be true

alldifferent *(parameter list)*

All components of the parameter list must take different values

Language for the expression of logic constraints

Illustrative examples

atmost ***(Y(i))***

exactly ***(Y1, Y2, Y6, Y8)***

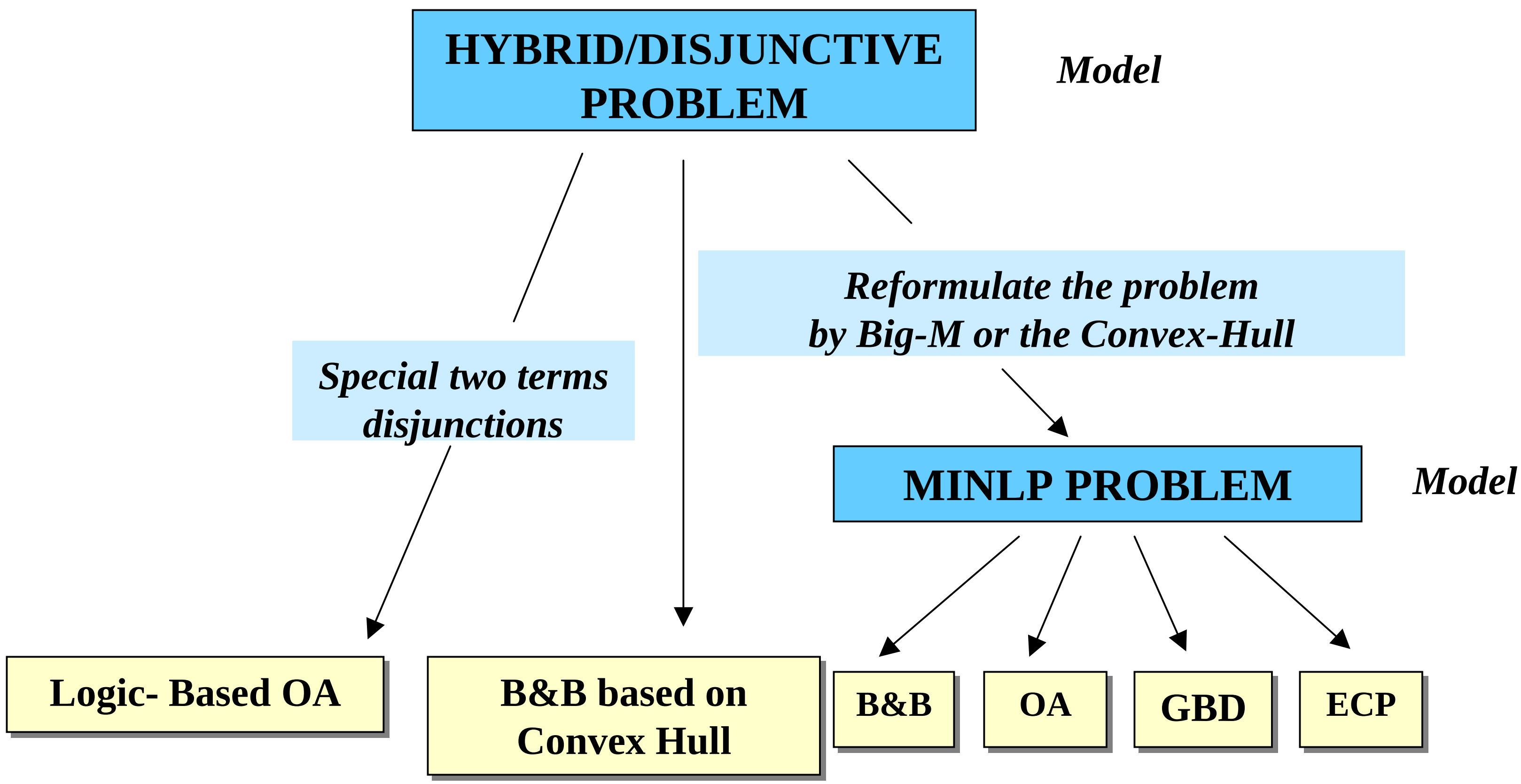
atleast ***(Y(i), Y2, Y6, Y8)***

exactly ***(Y1, Y2, Y6, Y8)***

atleast ***(Y(i), Y2, Y6, Y8)***

alldifferent ***(n(i))***

Models and Algorithms



Reformulation of a disjunctive problem into a MINLP

Disjunctive Formulation

$$\min \quad Z = \Sigma_k c_k + f(x)$$

s. t.:

$$g(x) \le 0$$

$$\bigvee_{i \in D_k} \begin{bmatrix} Y_{ik} \\ h_{ik}(x) \le 0 \\ c_k = \gamma_{ik} \end{bmatrix} \quad k \in SD$$

$$\Omega(Y) = True$$

$$x \in R^n \;, \; Y_{ik} \in \{True, False\}^m, \; c_k \ge 0$$

MINLP Formulation

$$\min Z = \quad \gamma_{ik}\, y_{ik} + f(x)$$

s. t.:

$$g(x) \le 0$$

$$x = \underset{i \in D_k}{} \nu_{ik}, \quad k \in SD1$$

$$y_{ik} h_{ik}(\nu_{ik} / y_{ik}) \le 0, \quad i \in D_k, k \in SD1$$

$$0 \le \nu_{ik} \le y_{ik} U_{ik}, \quad i \in D_k, k \in SD1$$

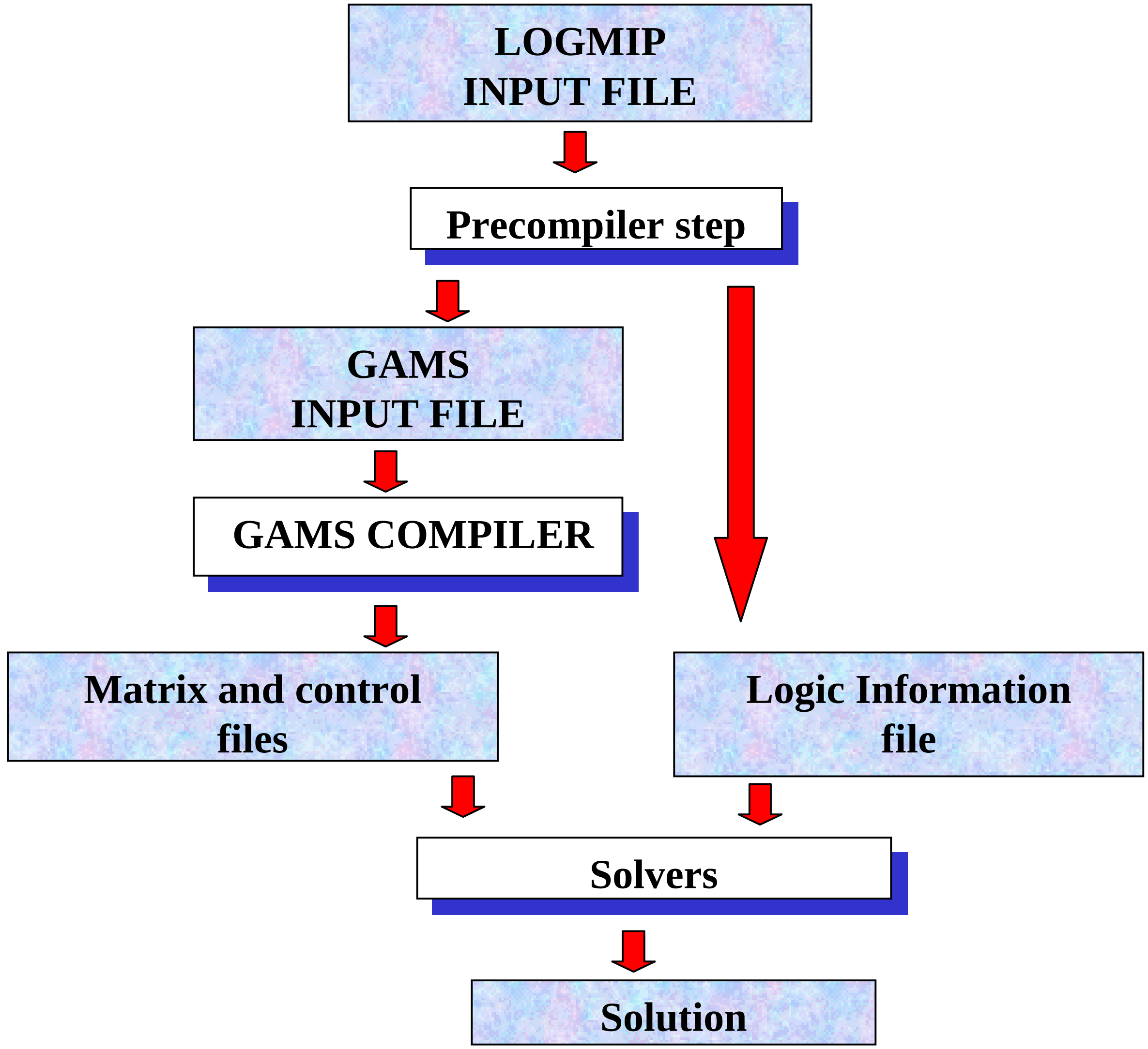
$$h_{ik}(x) \le M_{ik}(1 - y_{ik}), \quad i \in D_k, k \in SD2$$

$$\underset{i \in D_k}{} y_{ik} = 1, \quad k \in SD$$

$$Ay \le a$$

$$x, \nu_{ik} \ge 0, \; y_{ik} \in \{0, 1\}, \; SD1, SD2 \in SD$$

LOGMIP Implementation



LOGMIP Implementation

Precompiler

The implementation implies the expansion of the capabilities of a mathematical programming at the level of modeling and solution techniques for non-linear discrete problems

The objective of the precompiler step is:

- **check the syntax and semantics for disjunctions, logic propositions logic constraints, special sentences**
- **obtain the logic information and record into a file**
- **generate a file ready to be compiled by GAMS**

LOGMIP Implementation

Algorithms Implemented

Formulation		*Algorithms*
MINLP	→	OA/ER/AP : Dicopt++
	→	B&B : SBB
	→	ECP : Prototype
Disjunctive	→	Logic-Based OA (for special two terms disjunctions)
Hybrid	→	Logic-Based OA extended (for special two terms disjunctions)

LOGMIP Implementation

Problem solving

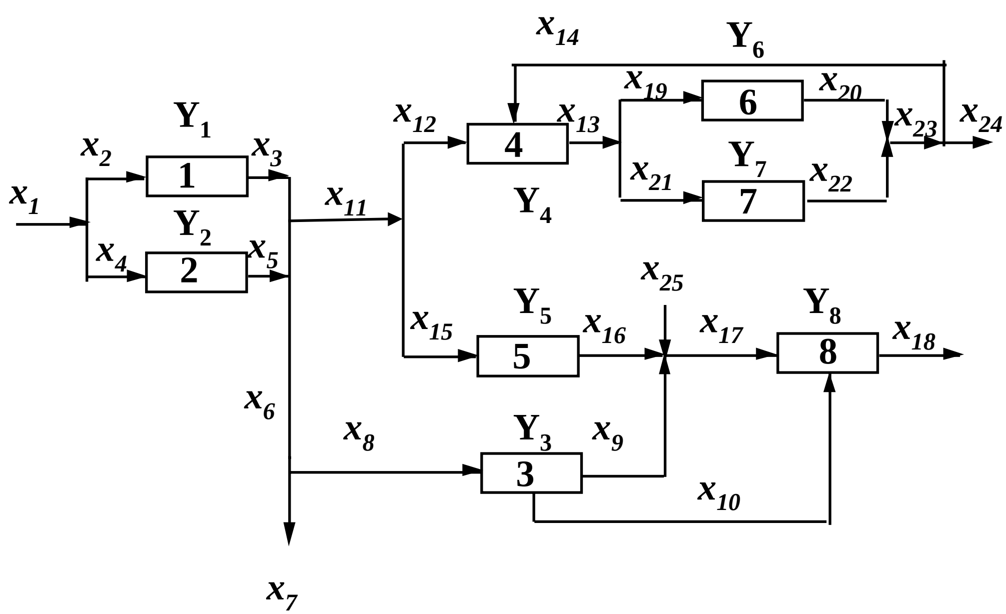
If the model is an MINLP

- By default we apply (DICOPT++)
- Alternatives SBB or ECP

If the model is a hybrid/disjunctive program

- We reformulate the problem as MINLP and proceed
- Alternative: Logic-Based OA for special two terms disjunctions

8 Process Superstructure



Results : 8 Process superstructure

Model	Algorithm	Relaxed optimum	Iterations	Variab.	Constr.	Discrete V.
MINLP	Dicopt++	15.08	1 NLP 4 major	32	33	8
	ECP	-25.13*	6 MIP	32	33	8
	GDB	15.08	1 NLP 14 major	32	33	8
MINLP (Convex hull)	Dicopt++	62.6	1 NLP 2 major	51	41	8
Disjunct. Hybrid	Logic-Based OA	---------	3 NLP 1 major	52	42	8
	Logic-Based OA ext.	----------	2 NLP 1 major	52	42	8
Optimum		68.09				

Results: prediction of infrared spectroscopy parameters

$$min\ \ Z = w_j + 2 \sum_k \sum_i y_{ki}$$

$$w_j = \sum_k \{[(c_{kj} - \sum_i p_{ki} a_{ij})^T * R^{-1}] * (\sum_k c_{kj} - \sum_i p_{ki} a_{ij})]\} \qquad \forall j$$

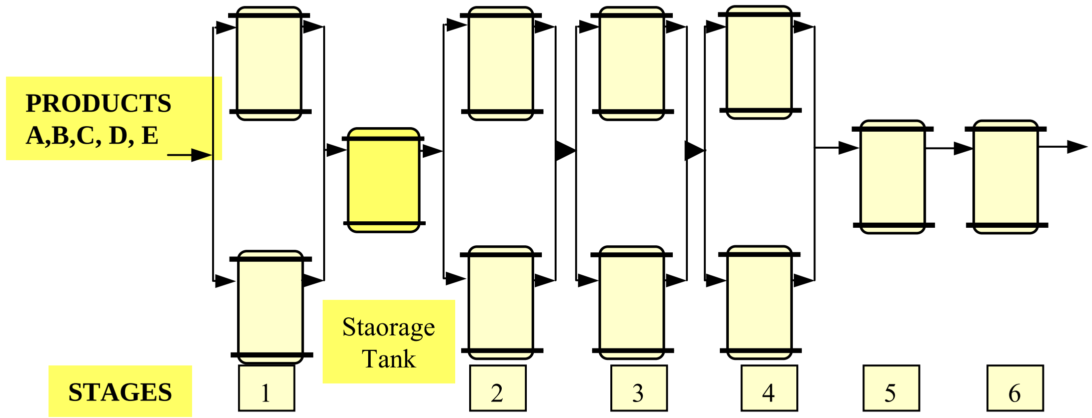
Disyunciones

i = wave number(10)
j= experimental data (8)
k= components (3)

$$\begin{bmatrix} Y_{ki} \\ P_{ki}^{min} \le P_{ki} \le P_{ki}^{max} \\ c_{ki} = 1 \end{bmatrix} \vee \begin{bmatrix} \neg Y_{ki} \\ P_{ki} = 0 \\ c_{ki} = 0 \end{bmatrix}$$

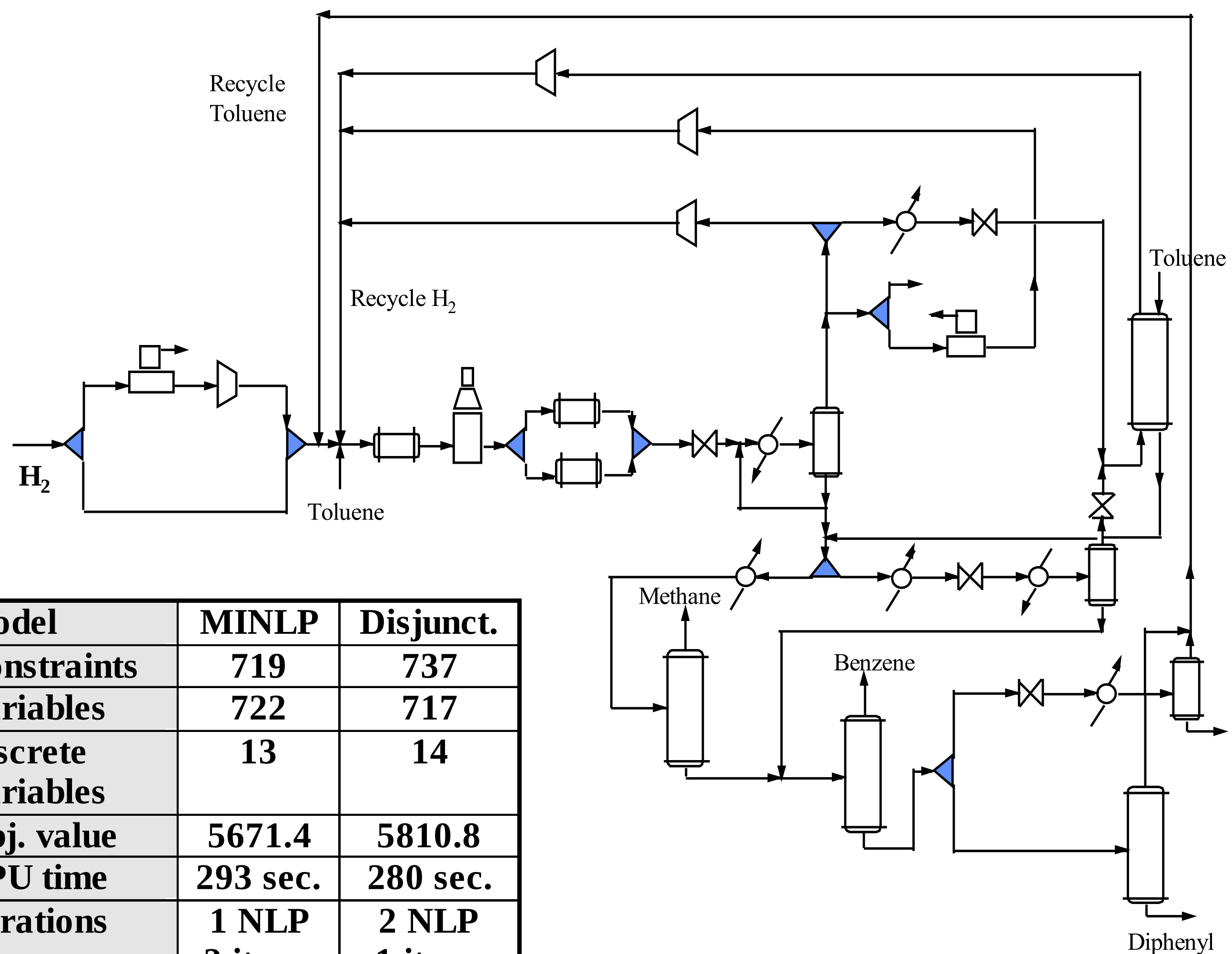
Model	MINLP	Disjunctive
Constraints	39	102
Variables	69	99
Discrete Var.	30	30
Optimum	13.98	13.98
CPU Time	55 sec	7 sec
Iterations	6	4

Results: Multiproduct Batch Plant Design



Model	MINLP	Hybrid
Constraints	186	187
Variables	112	113
Discrete variables	53	53
Obj. value	261883	261883
CPU Time	287 sec.	80 sec.
Iterations	1 NLP 10 itera	1 NLP 4 itera

Results : HDA process synthesis



Model	MINLP	Disjunct.
Constraints	719	737
Variables	722	717
Discrete Variables	13	14
Obj. value	5671.4	5810.8
CPU time	293 sec.	280 sec.
Iterations	1 NLP 2 itera.	2 NLP 1 itera.

Conclusions

- Hybrid and Disjunctive Programming provide advantages in modeling and solution techniques that complements Mixed Integer Non Linear Programming (MINLP)

- We propose a language for the expression of disjunctions and logic propositions to extend the mathematical modeling languages

- Starting with a hybrid/disjunctive model we propose to reformulate it to MINLP and then solve with any standard algorithm.

- Objective of this approach is to give the modeler several alternatives for modeling and solving a continuous/discrete non-linear program problem