Disjunctive programming: algorithms, implementation and solution of linear and nonlinear models

Aldo Vecchietti
INGAR - Instituto de Desarrollo y Diseño
UTN - Facultad Regional Santa Fe
e-mail: aldovec@ceride.gov.ar
Motivation

- Disjunctive Programming has proved to be a successful modeling framework for problems involving discrete decisions.

- LogMIP: develop a system for solving disjunctive problems in GDP formulation.
  - Generate a language for the expressions of disjunctions, logic constraints and logic propositions.
  - Implement and develop techniques and algorithms for solving linear/nonlinear disjunctive problems.
General Hybrid/Disjunctive Problem (GHDP)

\[
\begin{align*}
\min \quad Z &= \sum_k c_k + f(x) + d^T y \\
\text{st} \quad g(x) &\leq 0 \\
&\quad r(x) + Dy &\leq 0 \\
&\quad Ay &\geq a \\
\forall k \in D_k \quad &\begin{bmatrix}
Y_{ik} \\
\gamma_{ik} \\
\end{bmatrix} \\
&\leq h_{ik}(x) &\leq 0 \\
& c_k = \gamma_{ik} \\
\end{align*}
\]

\( \Omega(Y) = \text{True} \)

This is a general formulation that can be used for LogMI P
Disjunction Relaxations

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

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

**Big-M**

\[
a_i^T x \leq b_i + M_i (1 - y_i)
\]

\[
\sum_i y_i = 1
\]

\[
M_i = \max \{ a_i^T x - b_i \mid x^{lo} \leq x \leq x^{up} \}
\]

**Convex Hull**

\[
x - \sum_{i \in D} v_i = 0 \quad x, v_i \in \mathbb{R}^n
\]

\[
a_i^T v_i - b_i y_i \leq 0
\]

\[
\sum_{i \in D} y_i = 1 \quad , 0 \leq y_i \leq 1, \ i \in D
\]

\[
0 \leq v_i \leq v_i^{up} y_i
\]
DISJUNCTION CHARACTERIZATION

**improper**

\[ R_i \subseteq R_j \quad \forall i \neq j \]

**proper**

\[ \bigcap_{i \in D} R_i \neq \emptyset \]

\[ \bigcap_{i \in D} R_i = \emptyset \]
Improper Disjunction

\[ \min Z = (x_1 - 3.5)^2 + (x_2 - 4.5)^2 \]

sujeto a:

\[
\begin{bmatrix}
Y_1 \\
1 \leq x_1 \leq 3
\end{bmatrix} \lor 
\begin{bmatrix}
Y_2 \\
2 \leq x_1 \leq 3, 3 \leq x_2 \leq 4
\end{bmatrix}
\]

• The disjunction could be replaced by the disjunction term with the largest feasible region.

The x space
Proper disjunction - Non-empty Intersection

For this case is not clear which relaxation is tighter

Both relaxations are equivalent

The convex hull relaxation has a tighter feasible region

The objective function plays an important role:

- when located inside the region of one term both relaxations are competitive
- in general the convex hull relaxation is tighter
Proper disjunction - Empty Intersection

Both relaxations are equivalent

The convex hull relaxation has a tighter feasible region

For this case can be asserted that for the general case the convex hull renders a tighter feasible region
Improper disjunction
Special Interest in Process Engineering
(Synthesis Problems)

\[
\begin{align*}
\text{min } Z &= (x_1 - 1.1)^2 + (x_2 - 1.1)^2 + c_1 \\
\text{s.t.} \\
\begin{bmatrix}
Y_1 \\
x_1^2 + x_2^2 \leq 1 \\
c_1 = 1
\end{bmatrix} &\lor 
\begin{bmatrix}
\neg Y_1 \\
x_1 = x_2 = 0 \\
c_1 = 0
\end{bmatrix} \\
0 \leq x_1, x_2 \leq 1; 0 \leq c_1 \\
Y_1 \in \{\text{true, false}\}
\end{align*}
\]

Both relaxations have the same feasible region
Improper disjunction
Special Interest in Process Engineering
(Synthesis Problems)

Convex hull

Including the y-space

Big-M

\[ \min Z = (x_1 - 1.1)^2 + (x_2 - 1.1)^2 + y_1 \]

s.t.
\[ x_1^2 + x_2^2 \leq y_1^2 \]
\[ 0 \leq x_1 \leq y_1 \]
\[ 0 \leq x_2 \leq y_1 \]
\[ 0 \leq y_1 \leq 1 \]

\[ \min Z = (x_1 - 1.1)^2 + (x_2 - 1.1)^2 + y_1 \]

s.t.
\[ x_1^2 + x_2^2 \leq y_1 \]
\[ x_1 \leq 1; 0 \leq y_1 \leq 1 \]
System linked to GAMS
Problems can be formulated in GHDP
Problems can be linear or nonlinear discrete
Provides:
• Language to write disjunctions
• Operators and sentences for logic propositions
• Linear and nonlinear solvers

http://www.ceride.gov.ar/logmip
LogMIP: Modeling two terms disjunction

\[
\begin{bmatrix}
\text{condition} \\
\text{constraint } s \text{ set } t
\end{bmatrix} \lor
\begin{bmatrix}
\neg \text{condition} \\
\text{constraint } s \text{ set } f
\end{bmatrix}
\]

Conditions in this LogMIP version are Boolean (binary) variables

Declaration sentence: **Disjunction TTD**;

**TTD is**
**IF (condition) THEN**

\textit{constraints set (names) to satisfy when condition is TRUE;}

**ELSE**

\textit{constraints set (names) to satisfy when condition is FALSE;}

**END IF;**
LogMIP: Modeling a multi-term disjunction

\[
\begin{align*}
\text{condition 1} &\lor \text{condition 2} &\lor & \ldots &\lor &\text{condition N} \\
\text{constraint s set 1} & &\text{constraint s set 2} & &\text{constraint s set N}
\end{align*}
\]

**Declaration sentence:**
**Disjunction MTD;**

**Definition sentence:**
MTD is
IF (condition\textsubscript{1}) THEN
  constraints set 1 (names) to be satisfied when condition\textsubscript{1} is True;
ELSIF (condition\textsubscript{2}) THEN
  constraints set 2 (names) to be satisfied when condition\textsubscript{2} is True;
ELSIF (condition\textsubscript{3}) THEN
  ...
ELSIF (condition\textsubscript{N}) THEN
  constraints set N (names) to be satisfied when condition\textsubscript{N} is True;
END IF
LogMIP: Posing logic propositions

Operands: Boolean (binary) variables (must correspond to disjunctions and conditions)

Operators: and, or, not, --> (implication), <--> equivalence

Y('2') --> Y('3') or Y('4') or Y('5');
Y('1') and not Y('2') --> not Y('3');
Y('2') --> not Y('3');
Y('3') --> Y('8');

More declarative special sentence:

- atmost, at least, exactly

Syntax:

- [atmost]
- [atleast] (<list of boolean variables>, n)
- [exactly]

atmost(Y('1'), Y('2'));
atleast(Y('4'), Y('5'));
EXAMPLE 1

\[
\begin{align*}
\text{min } Z &= T \\
\text{s.t. } & T \geq x_1 + 8 \\
& T \geq x_2 + 5 \\
& T \geq x_3 + 6 \\
& Y_1 \left[ x_1 - x_3 + 5 \leq 0 \right] \lor \left[ x_3 - x_1 + 2 \leq 0 \right] \\
& Y_2 \left[ x_2 - x_3 + 1 \leq 0 \right] \lor \left[ x_3 - x_2 + 6 \leq 0 \right] \\
& Y_3 \left[ x_1 - x_2 + 5 \leq 0 \right] \lor \left[ x_2 - x_1 \leq 0 \right] \\
T, x_1, x_2, x_3 & \geq 0 \\
Y_k & \in \{\text{true, false}\}, k = 1,2,3.
\end{align*}
\]
EXAMPLE 1: LogMIP File

SET J /1*3/;
BINARY VARIABLES Y(J);
POSITIVE VARIABLES X(J), T;
VARIABLE Z;
EQUATIONS EQUAT1, EQUAT2, EQUAT3, EQUAT4, EQUAT5, EQUAT6, EQUAT7, EQUAT8, EQUAT9, FICT, OBJECTIVE;

EQUAT1.. T =G= X('1') + 8;
EQUAT2.. T =G= X('2') + 5;
EQUAT3.. T =G= X('3') + 6;
EQUAT4.. X('1')-X('3')+ 5 =L= 0;
EQUAT5.. X('3')-X('1')+ 2 =L= 0;
EQUAT6.. X('2')-X('3')+ 1 =L= 0;
EQUAT7.. X('3')-X('2')+ 6 =L= 0;
EQUAT8.. X('1')-X('2')+ 5 =L= 0;
EQUAT9.. X('2')-X('1') =L= 0;
FICT.. SUM(J, Y(J)) =G= 0;
X.UP(J)=12.;
OBJECTIVE.. Z =E= T;

$ONTEXT BEGIN LOGMIP
DISJUNCTION D1,D2,D3;
D1 IS IF (Y('1')) THEN EQUAT4;
ELSE EQUAT5;
ENDIF;
D2 IS IF(Y('2')) THEN EQUAT6;
ELSE EQUAT7;
ENDIF;
D3 IS IF(Y('3')) THEN EQUAT8;
ELSE EQUAT9;
ENDIF;
$OFFTEXT END LOGMIP

OPTION MIP=LOGMIPC;
MODEL example1 /ALL/;
SOLVE example1 USING MIP MINIMIZING Z;
EXAMPLE 2

\[
\begin{align*}
\min c &+ 2x_1 + x_2 \\
Y_1 &
\begin{bmatrix}
-x_1 + x_2 + 2 \leq 0 \\
c = 5
\end{bmatrix}
\lor
\begin{bmatrix}
Y_2 \\
2 - x_2 \leq 0 \\
c = 7
\end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
Y_3 &
\begin{bmatrix}
x_1 - x_2 \leq 1 \\
x_1 = 0
\end{bmatrix}
\lor
\begin{bmatrix}
\neg Y_3
\end{bmatrix}
\end{align*}
\]

\[
Y_1 \land \neg Y_2 \Rightarrow \neg Y_3
\]

\[
\neg (Y_2 \land Y_3)
\]

\[
0 \leq x_1 \leq 5, \ 0 \leq x_2 \leq 5, \ c \geq 0
\]

\[
Y_j \in \{\text{true, false}\}, j = 1,2,3.
\]
EXAMPLE 2: LogMIP File

```
SET I /1*3/;
SET J /1*2/;
BINARY VARIABLES Y(I);
POSITIVE VARIABLES X(J), C;
VARIABLE Z;
EQUATIONS EQUAT1, EQUAT2, EQUAT3, EQUAT4, EQUAT5, EQUAT6, INT1, INT2, INT3, FICT, OBJECTIVE;
EQUAT1.. X('2') - X('1') + 2 =L= 0;
EQUAT2.. C =E= 5;
EQUAT3.. 2 - X('2') =L= 0;
EQUAT4.. C =E= 7;
EQUAT5.. X('1') - X('2') =L= 1;
EQUAT6.. X('1') =E= 0;
INT1.. Y('1') + Y('3') =L= 1;
INT2.. Y('2') + (1 - Y('3')) =G= 1;
INT3.. Y('2') + Y('3') =L= 1;
FICT.. SUM(I, Y(I)) =G= 0;
OBJECTIVE.. Z =E= C + 2*X('1') + X('2');
X.UP(J)=20;
C.UP=7;
```

$ONTEXT BEGIN LOGMIP

DISJUNCTION D1,D2;
D1 IS
IF Y('1') THEN
  EQUAT1;
  EQUAT2;
ELSIF Y('2') THEN
  EQUAT3;
  EQUAT4;
ENDIF;
D2 IS
IF Y('3') THEN
  EQUAT5;
ELSE
  EQUAT6;
ENDIF;
Y('1') and not Y('2') -> not Y('3');
Y('2') -> not Y('3');

$OFFTEXT END LOGMIP

OPTION MIP=LOGMIPC;
MODEL PEQUE2 /ALL/;
SOLVE PEQUE2 USING MIP MINIMIZING Z;
```
Declaring and defining disjunctions over a domain

**DI SJ UNCTI ON** disjunction_identifier[ domain_identifier, ..., domain_identifier], ...
, disjunction_identifier [ domain_identifier, ..., domain_identifier];

Example: **DI SJ UNCTI ON** D(I,J);

D(I,J) IS
  IF Y(I,J) THEN
    CONSTRAINT1(I,J);
    EQUATION1(I,J);
  ELSE
    CONSTRAINT2(I,J);
    EQUATION2(I,J);
  ENDIF;

One disjunction is defined for every pair I,J

You cannot define a domain inside the LOGMIP section. The reason is that the disjunction’s domains must be in concordance to the constraint’s domains, which are defined in the GAMS section.
In previous examples Constraint’s domains are expanded together the disjunction’s domains. If constraint’s domain are different in LogMIP than in GAMS section, LogMIP reports an error.

Disjunction’s domain are controlled by the sentence with plus other operators:

**Relational operators:**
- lt, < : less than
- le, <= : less than or equal to
- eq, = : equal
- gt, > : greater than
- ge, >= : greater than or equal to

**Logical operators:** and, or.

**Sets operators:**
- ord : order of an item in the set
- card : number of items in the set
- in : inclusion of a set item
Controlling disjunction’s domain

EXAMPLE 1

Disjunction \( D(i,j); \)

\( D(i,j) \text{ with (ord}(i) < \text{ord}(j)) \text{ IS} \)

IF \( Y(i,j) \) THEN

CONSTR1(j);
CONSTR2(i,j);

ELSE

CONSTR3(j);
CONSTR4(j,k) \text{ with (ord}(k) \geq 1);\n
ENDIF;

Suppose we have in GAMS Section: \text{SET I /1*3/ , J /1*4/ , K/1*2/ ;}\n
With this definition, the following disjunctions are generated: \( D('1', '2'), D('1', '3'), D('1', '4'), D('2', '3'), D('2', '4'), D('3', '4'). \)
Controlling disjunction’s domain

EXAMPLE 2: Controlling a domain already controlled

For this case an alias is needed in GAMS section:

```
GAMS Section
SET I /1*3/ , J /1*4/ ;
ALIAS (J,JJ);

LogMIP Section
Disjunction D(i,j);
D(i,j) with (ord(i) < ord(j)) IS
  IF Y(i,j) THEN
    CONST1(j);
    CONST2(i,jj) with (ord(jj) le 2);
  ELSE
    CONST3(j);
    CONST4(j,k) with (ord(k) lt card(k));
END IF;
```
Controlling disjunction’s domain

EXAMPLE 3: Controlling a domain via a SUBSET

GAMS Section:
SET I / 1*3/ , J / 1*4/ ;
* Define the subset k
SET K(I,J) / 1.2, 2.3, 3.4 / ;

LogMIP Section
Disjunction D(I,J);
D(I, J) with K(I,J) IS
  IF Y(I,J) THEN
    CONSTRAINT(I,J);
    CONSTRAINT(I,J);
  ELSE
    CONSTRAINT(I,J);
    CONSTRAINT(I,J);
  ENDIF;

Disjunctions generated: D(‘1’,’2’), D(‘2’,’3’) and D(‘3’,’4’).
Hierarchical Discrete Decisions
(Nested Disjunctions)

Nested disjunctions can not be used in the actual version of LogMIP.
Hierarchical decisions are common in PSE: synthesis and design problems,
e.g: discontinuous cost functions, simultaneous planning and scheduling,
synthesis and design of batch plants.
Transforming Hierarchical Discrete Decisions into GHDP form

\[
\begin{bmatrix}
Y_i \\
h(x) = 0
\end{bmatrix} \vee \begin{bmatrix}
\neg Y_i \\
B^i x = 0 \\
d_i = 0 \\
c_i = 0
\end{bmatrix}
\]

\[
(\bigvee_{j \in D_i} Z_{i,j} \\
c_i = (\alpha_i \xi_j^\eta_j + \beta_j)
\) \vee \begin{bmatrix}
\neg Y_i \\
c_i = 0
\end{bmatrix}
\]

\[
(\bigvee_{k \in E_i} Z_{i,k} \\
\gamma_i(P) = \gamma_{ik}
\) \vee \begin{bmatrix}
\neg Y_i \\
\text{variables can take any value between bounds}
\end{bmatrix}
\]

\[
(\bigvee_{m \in F_i} Z_{i,m} \\
\delta_i(T) = \delta_{im}
\) \vee \begin{bmatrix}
\neg Y_i \\
\text{variables can take any value between bounds}
\end{bmatrix}
\]
Transforming Hierarchical Discrete Decisions into GHDP form

Algorithm steps:
1- take out the inner disjunctions, leaving nested disjunctions into a set of individual ones,
2- define an extra term for the disjunction corresponding to the inner to represent the fact that none of the other terms is true
3- define the equivalence propositions between the outer and the inner disjunctions (logic propositions or algebraic constraints)
LogMIP Algorithms

HYBRID / DISJUNCTIVE PROGRAM

LINEAR
- MIP Reformulation by “Big-M” or Convex hull Relaxation
  - LogMIP PC
  - LogMIP PM
  - MILP PROGRAM
    - B&B (OSL, CPLEX, etc)

NONLINEAR
- Special two terms disjunction
  - LogMIP PV3
  - Logic Based OA
- MINLP Reformulation by “Big-M” or Convex hull Relaxation
  - MINLP PROGRAM
    - B&B (SBB)
    - OA (DICOPT++)

NOT DONE YET!
Nonlinear Disjunctive problems
Logic-Based OA algorithm

Nonlinear models solved by Logic-Based Outer Approximation needs initialization, these are needed to run the first NLP problems to provide initial values for the first MASTER MIP subproblem. More details can be found in Turkay and Grossmann (1996).

The clause INIT is used.

INIT TRUE Y('1'), Y('3'), Y('4'), Y('7'), Y('8');
INIT TRUE Y('1'), Y('3'), Y('5'), Y('8');
INIT FALSE Y('2'), Y('3'), Y('4'), Y('6'), Y('8');

Initialization entries must be written after the disjunction definitions

Other options:

INIT TRUE ALL;
or
INIT FALSE ALL;
Hybrid and Disjunctive Programming provide advantages in modeling and solution techniques that complements Mixed Integer Non Linear Programming (MINLP)

LogMIP extends the capabilities of the mathematical modeling systems by means of a language for the expression of disjunctions and logic propositions

Starting with a linear hybrid/disjunctive model it is reformulated into a MIP (by Convex Hull or BigM relaxation). For nonlinear problems with special two terms disjunctions Logic-Based Outer Approximation is used.

LogMIP becomes an alternative modeling and solving continuous/discrete linear/nonlinear program problem