LogMIP 2.0
USER'S MANUAL

October 2011

By Aldo Vecchietti
aldovec@santafe-conicet.gov.ar

INDEX

1. Introduction…… 2

2. Disjunctive model formulation…… 3

2.1 SMALL EXAMPLE 1 …… 4

2.2 SMALL EXAMPLE 2…… 7

2.3 NONLINEAR EXAMPLE…… 8

3. How to write a disjunctive model for LogMIP……12

3.1. Controlling disjunctions and constraints domain…….15

3.2. Using a DUMMY Equation……..15

4. Logic Propositions……16

4.1. Declaration Sentence……16

4.2. Definition Sentence……16

5. Solvers……16

5.1. Solution Algorithms for Linear Problems……17

5.2. Solution Algorithms for Non-Linear Problems……17

6. Recommendations and Limitations……19

7. References……19

1. Introduction

LogMIP 2.0 is a program for solving linear and nonlinear disjunctive programming problems involving binary variables and disjunction definitions for modeling discrete choices. While the modeling and solution of these disjunctive optimization problems has not yet reached the stage of maturity and reliability as LP, MIP and NLP: Disjunctive problems have a rich area of applications.

LogMIP 2.0 has been developed by Dr. Aldo Vecchietti from INGAR (Santa Fe-Argentina), Professor Ignacio E. Grossmann from Carnegie Mellon University (Pittsburgh-USA) and the cooperation of GAMS´s staff. It becomes a progress from its previous version (LogMIP 1.0).

PLEASE NOTE:

- LogMIP 1.0, works for 22.6 (December 2007) until 23.6 (December 2010) GAMS releases.
- LogMIP 2.0 is included from 23.7 GAMS release. LogMIP 1.0 does not work anymore from this release.
- Changes in version 2.0 are at the level of language, where LogMIP now uses the EMP syntax and modeltype.
- Solvers for linear disjunctive models (*lmbigm* and *lmchull*) are combined in one new called just *logmip*.
- In this version, non-linear disjunctive models can be solved using Big-M and convex hull relaxations algorithms.
- Non-linear solver *lmlboa* (Logic-Based Outer Approximation) does not work inLogMIP 2.0, a new version is currently developed and will be ready for a next GAMS release.
- LogMIP is composed of:
    - language sentences for the definition of disjunctions and logic constraints, and
    - solvers for linear and non-linear disjunctive models.

  These parts are linked to GAMS, becomes a subset of GAMS language and solvers respectively. LogMIP cannot be executed independently of GAMS system.
- Besides of disjunction definitions, LogMIP needs the declaration and definitions of GAMS´s scalars, sets, tables, variables, constraints, equations, etc.; for the specifications and solution of a disjunctive problem.

2. Disjunctive model formulation

The models for LogMIP have the following general formulation (Generalized Disjunctive Programming –GDP):

$$\min \ Z = \sum_{k} c_k + f(x) + d^t y$$

s.t.

$$g(x) \le 0$$

$$r(x) + D\,y \le 0$$

$$A\,y \le a$$

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

$$\Omega(Y) = True$$

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

x, c_k are continuous variables,

y	are binary variables (0-1),
Y_{ik}	are Boolean variables, to establish whether a disjunction term is true or false
$\Omega(Y)$	logic relationships between Boolean variables,
$f(x)$	objective function, which can be linear or non-linear,
$g(x)$	linear or non-linear inequalities/equalities independent of the discrete choices,
$r(x)+Dy \le 0$	mixed-integer inequalities/equalities that can contain linear or non-linear continuous terms (integer terms must be linear),
$Ay \le a$	linear integer inequalities/equalities
$d^T y$	linear fixed cost terms.

Before explaining the details about the sentences to pose a disjunctive problems and its solvers, in the next sections three small examples are presented in order to illustrate the meaning of the previous GDP formulation. The first two corresponds to linear models, the later to a nonlinear one.

2.1. SMALL EXAMPLE 1

$$\min Z = T$$

$$s.a. \quad T \geq x_A + 8$$
$$T \geq x_B + 5$$
$$T \geq x_C + 6$$

$$\begin{bmatrix} Y_1 \\ x_A - x_C + 5 \leq 0 \end{bmatrix} \vee \begin{bmatrix} \neg Y_1 \\ x_C - x_A + 2 \leq 0 \end{bmatrix}$$
$$\begin{bmatrix} Y_2 \\ x_B - x_C + 1 \leq 0 \end{bmatrix} \vee \begin{bmatrix} \neg Y_2 \\ x_C - x_B + 6 \leq 0 \end{bmatrix}$$
$$\begin{bmatrix} Y_3 \\ x_A - x_B + 5 \leq 0 \end{bmatrix} \vee \begin{bmatrix} \neg Y_3 \\ x_B - x_A \leq 0 \end{bmatrix}$$

$$T, x_A, x_B, x_C \geq 0$$
$$Y_k \in \{true,\ false\},\ k = 1,2,3.$$

This example corresponds to a Jobshop (Jobshop scheduling) problem, having three jobs (A,B,C) that must be executed sequentially in three steps (1,2,3), but not all jobs require all the stages, meaning that the jobs will be executed in a subset of stages. The processing time for each stage is given by the following table:

Job/stage	1	2	3
A	5	-	3
B	-	3	2
C	2	4	-

The objective is to obtain the sequence of task, which minimizes the completion time T. In order to obtain a feasible solution the clashes between the jobs must be eliminated.

For more details about this formulation see Raman y Grossmann (1994).

LogMIP input file for this example

First Version – Using 3 explicit binary variables

```
BINARY VARIABLES Y1, Y2, Y3;
POSITIVE VARIABLES XA, XB, XC, T;
VARIABLE Z;
EQUATIONS
EQUAT1, EQUAT2, EQUAT3, EQUAT4, EQUAT5,
EQUAT6, EQUAT7, EQUAT8, EQUAT9, DUMMY, OBJECTIVE;

EQUAT1..  T =G= XA + 8;
EQUAT2..  T =G= XB + 5;
EQUAT3..  T =G= XC + 6;
EQUAT4..  XA – XC + 5 =L= 0;
EQUAT5..  XC – XA + 2 =L= 0;
EQUAT6..  XB – XC + 1 =L= 0;
EQUAT7..  XC – XB + 6 =L= 0;
EQUAT8..  XA – XB + 5 =L= 0;
EQUAT9..  XB - XA =L= 0;
OBJECTIVE.. Z =E= T;
XA.UP=20.; XB.UP=20.; XC.UP=20.;
DUMMY.. Y1+Y2+Y3 =G= 0;

$ONECHO > "%lm.info%"
* by default the convex hull reformulation is used
disjunction Y1 equat4 else equat5
disjunction Y2 equat6 else equat7
disjunction Y3 equat8 else equat9

* optional, if not set EMP will find the modeltype suitable
modeltype mip
$OFFECHO
```

Constraint independent of discrete choices

Constraints for discrete choices (disjunctions)

GAMS´s variable and equation declarations

GAMS equation and constraint definitions. Constraint definitions corresponding to disjunction must be defined here.

Dummy equation just to avoid the elimination of variable Y from the model, which handles disjunction terms.

Disjunction definitions according to the new (EMP) syntax rules. Note that constraints belonging to a disjunction term are declared (given its name) in this section.

```
OPTION EMP = LOGMIP;
OPTION OPTCR=0.0;
MODEL SMALL11 /ALL/;
SOLVE SMALL11 USING EMP MINIMIZING Z;
```

Calls LogMIP solvers now belonging to the EMP environment

EMP must be in the SOLVE sentence which includes LogMIP

Second Version – Using default Boolean variables in disjunction definitions

```
POSITIVE VARIABLES XA, XB, XC, T;
VARIABLE Z;
EQUATIONS
    EQUAT1, EQUAT2, EQUAT3, EQUAT4, EQUAT5,
    EQUAT6, EQUAT7, EQUAT8, EQUAT9, OBJECTIVE;
```

GAMS´s variable and equation declarations. NOTE THAT no binary variables are defined here because * is defined in disjunction´s section

```
EQUAT1..  T =G= XA + 8;
EQUAT2..  T =G= XB + 5;
EQUAT3..  T =G= XC + 6;
EQUAT4..  XA – XC + 5 =L= 0;
EQUAT5..  XC -  XA + 2 =L= 0;
EQUAT6..  XB – XC + 1 =L= 0;
EQUAT7..  XC – XB + 6 =L= 0;
EQUAT8..  X('A')-X('B')+ 5 =L= 0;
EQUAT9..  X('B')-X('A') =L= 0;
OBJECTIVE.. Z =E= T;
XA.UP=20.; XB.UP=20.; XC.UP=20.;
```

Constraints independent of discrete choices (disjunctions)

Constraints for discrete choices (disjunctions)

Note that a constraint belonging to a disjunction term is declared (given its name) in this section.

```
$ONECHO > "%lm.info%"
default BigM
disjunction * equat4 else equat5
disjunction * equat6 else equat7
disjunction * equat8 else equat9

* optional, if not set EMP will find the modeltype suitable
modeltype mip
$OFFECHO
```

By means of this sentence LogMIP is forced to executed the BIGM relaxation method with default values of the M parameter

to the new (EMP) syntax rules. The * symbol is replaced by a default binary variable names in this case

```
OPTION EMP = LOGMIP;
OPTION OPTCR=0.0;
MODEL SMALL11 /ALL/;
SOLVE SMALL11 USING EMP MINIMIZING Z;
```

Call to LogMIP solvers, now belonging to the EMP environment

EMP must be in the SOLVE sentence. EMP includes LogMIP

Third Version – Alex Meeraus compact version

```
SETS J JOBS   / A, B, C /
     S STAGES / 1*3 /
     GG(J,J) Upper Triangle
ALIAS (J,JJ),(S,SS);

TABLE P(J,S)  Processing Time
    1   2   3
A   5       3
B       3   2
C   2   4

PARAMETER
     C(J,S)          Stage Completion Time
     W(J,JJ)         Maximum Pairwise Waiting Time
     PT(J)           Total Processing Time
     BIG             The Famous Big M;

GG(J,JJ) = ORD(J) < ORD(JJ);

C(J,S)   = SUM(SS$(ORD(SS)<=ORD(S)), P(J,SS));
W(J,JJ) = SMAX(S, C(J,S) - C(JJ,S-1));
PT(J)    = SUM(S, P(J,S));
BIG      = SUM(J, PT(J));

VARIABLES  T        Completion Time
           X(J)     Job Starting Time
           Y(J,JJ)  Job Precedence

POSITIVE VARIABLE X;
BINARY VARIABLE Y;

EQUATIONS COMP(J)   Job Completion Time
          SEQ(J,JJ) Job Sequencing
          DUMMY;

COMP(J).. T =G= X(J) + PT(J);
SEQ(J,JJ)$(ORD(J) <> ORD(JJ)).. X(J) + W(J,JJ) =L= X(JJ);
DUMMY.. SUM(GG(J,JJ), Y(J,JJ)) =G= 0;

X.UP(J) = BIG;

MODEL SMALL13 / ALL /;

file lg / '%lm.info%' /; put lg '* problem %gams.i%';
loop(gg(j,jj)$(ord(j) <> ord(jj)),
  put / 'disjunction ' Y(j,jj)  seq(j,jj) 'else' seq(jj,j));
putclose;
```

NOTE THAT when a disjunction is defined over a domain, the elements expansion must be done by using *file, put* and *loop* or *while* GAMS sentences. The control over de domain is done through de the dollar sign *$*. Do not forget to include a line feed (/) when you write a new disjunction

```
OPTION EMP = LOGMIP;
OPTION OPTCR=0.0;
MODEL SMALL11 /ALL/;
SOLVE SMALL11 USING EMP MINIMIZING Z;
```

Calls LogMIP solvers, now belonging to the EMP environment

EMP must be in the SOLVE sentence which includes LogMIP

2.2. SMALL EXAMPLE 2

$$\min c + 2x_1 + x_2$$

s.a.:

$$\begin{bmatrix} Y_1 \\ -x_1 + x_2 + 2 \le 0 \\ c = 5 \end{bmatrix} \vee \begin{bmatrix} Y_2 \\ 2 - x_2 \le 0 \\ c = 7 \end{bmatrix}$$

$$\begin{bmatrix} Y_3 \\ x_1 - x_2 \le 1 \end{bmatrix} \vee \begin{bmatrix} \neg Y_3 \\ x_1 = 0 \end{bmatrix}$$

$$Y_1 \wedge \neg Y_2 \Rightarrow \neg Y_3$$
$$Y_2 \Rightarrow \neg Y_3$$
$$Y_3 \Rightarrow \neg Y_2$$

$$0 \le x_1 \le 5, 0 \le x_2 \le 5, c \ge 0$$

$$Y_j \in \{\text{true}, \text{false}\}, j = 1,2,3$$

Small example for illustration purpose. It is composed by two disjunctions each one with two terms.

Each term of the first disjunction is handled by different variables. The first term is true if Y_1 is true; the second term of the first disjunction is true if Y_2 is true.

The second disjunction is handled just for one variable: Y_3. The first term apply if Y_3 is true, the second if Y_3 is false.

The logic propositions indicates that:

1. **If Y_1 is true and Y_2 false it implies that Y_3 must be false.**
2. **Y_2 and Y_3 cannot be both true at the same time.**

LogMIP input file for this example

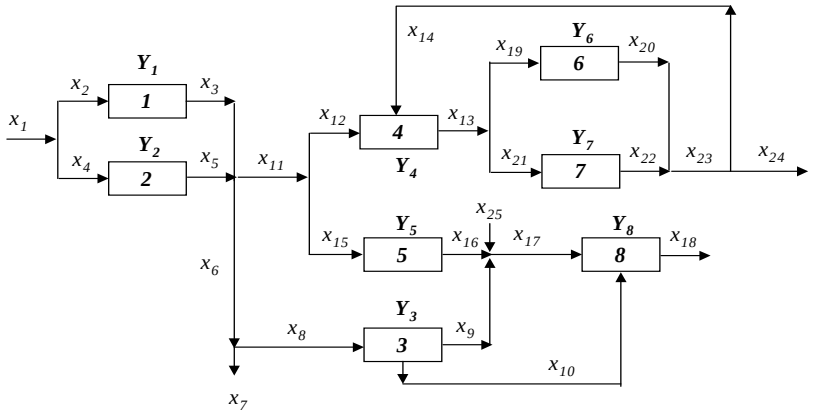
```
SCALAR M /100/;
BINARY VARIABLES Y1,Y2,Y3;
POSITIVE VARIABLES X1,X2,X3, C;
VARIABLE Z;
EQUATIONS EQUAT1, EQUAT2, EQUAT3, EQUAT4, EQUAT5, EQUAT6,
        OBJECTIVE;

EQUAT1..        X2 =L= X1 - 2;
EQUAT2..        C =E= 5;
EQUAT3..        X2 =G= 2;
EQUAT4..        C =E= 7;
EQUAT5..        X1-X2 =L= 1 ;
EQUAT6..        X1 =E= M * Y3;
Logic Equation  L1,L2,L3;
L1.. y1 and not y2 -> not y3;
L2.. y2 -> not y3;
L3.. y3 -> not y2;

OBJECTIVE.. Z =E= C + 2*X('1') + X('2');
X.UP(J)=5; C.UP=7;

$ONECHO > "%lm.info%"

disjunction y1 equat1 equat2 elseif y2 equat3 equat4
disjunction y3 equat5 else equat6

$OFFECHO

OPTION OPTCR=0.0, EMP=LOGMIP;
MODEL SMALL2 /ALL/;
SOLVE SMALL2 USING EMP MINIMIZING Z;
```

NOTE THAT Logic Constraints are NOW declared (*Logic Equation*) and defined in the GAMS Section. GAMS has extended its Language Syntax to define this type of constraints

OBSERVE the different syntax used to pose a two term disjunction; the first one where the terms are handled by two different variables (*y1* and *y2*); while the second one is handled by just one variable (*y3*) one term is satisfied by the TRUE value and the other with FALSE.

2.3. NON-LINEAR EXAMPLE

Synthesis of 8 processes



$$\min\ Z = \sum_{k=1}^{8} c_k + a^T x + 122$$

s.t

Mass Balances

$x_1 = x_2 + x_4,\ x_6 = x_7 + x_8$

$x_3 + x_5 = x_6 + x_{11}$

$x_{11} = x_{12} + x_{15},\ x_{13} = x_{19} + x_{21}$

$x_9 + x_{16} + x_{25} = x_{17}$

$x_{20} + x_{22} = x_{23},\ x_{23} = x_{14} + x_{24}$

Specifications

$x_{10} - 0.8\,x_{17} \le 0,\ x_{10} - 0.4\,x_{17} \ge 0$

$x_{12} - 5\,x_{14} \le 0,\ x_{12} - 2\,x_{14} \ge 0$

Disjunctions:

$$\begin{bmatrix} Y_1 \\ \exp(x_3) - 1 - x_2 \le 0 \\ c_1 = 5 \end{bmatrix} \vee \begin{bmatrix} \neg Y_1 \\ x_3 = x_2 = 0 \\ c_1 = 0 \end{bmatrix}$$

$$\begin{bmatrix} Y_2 \\ \exp(x_5/1.2) - 1 - x_4 \le 0 \\ c_2 = 5 \end{bmatrix} \vee \begin{bmatrix} \neg Y_2 \\ x_4 = x_5 = 0 \\ c_2 = 0 \end{bmatrix}$$

$$\begin{bmatrix} Y_3 \\ 1.5x_9 + x_{10} - x_8 = 0 \\ c_3 = 6 \end{bmatrix} \vee \begin{bmatrix} \neg Y_3 \\ x_9 = 0,\ x_8 = x_{10} \\ c_3 = 0 \end{bmatrix}$$

$$\begin{bmatrix} Y_4 \\ 1.25(x_{12} + x_{14}) - x_{13} = 0 \\ c_4 = 10 \end{bmatrix} \vee \begin{bmatrix} \neg Y_4 \\ x_{12} = x_{13} = x_{14} = 0 \\ c_4 = 0 \end{bmatrix}$$

$$\begin{bmatrix} Y_5 \\ x_{15} - 2x_{16} = 0 \\ c_5 = 6 \end{bmatrix} \vee \begin{bmatrix} \neg Y_5 \\ x_{15} = x_{16} = 0 \\ c_5 = 0 \end{bmatrix}$$

$$\begin{bmatrix} Y_6 \\ \exp(x_{20}/1.5) - 1 - x_{19} \le 0 \\ c_6 = 7 \end{bmatrix} \vee \begin{bmatrix} \neg Y_6 \\ x_{19} = x_{20} = 0 \\ c_6 = 0 \end{bmatrix}$$

$$\begin{bmatrix} Y_7 \\ \exp(x_{22}) - 1 - x_{21} \le 0 \\ c_7 = 4 \end{bmatrix} \vee \begin{bmatrix} \neg Y_7 \\ x_{21} = x_{22} = 0 \\ c_7 = 0 \end{bmatrix}$$

$$\begin{bmatrix} Y_8 \\ \exp(x_{18}) - 1 - x_{10} - x_{17} \le 0 \\ c_8 = 5 \end{bmatrix} \vee \begin{bmatrix} \neg Y_8 \\ x_{10} = x_{17} = x_{18} = 0 \\ c_8 = 0 \end{bmatrix}$$

Logic Propositions:

$Y_1 \Rightarrow Y_3 \vee Y_4 \vee Y_5$
$Y_2 \Rightarrow Y_3 \vee Y_4 \vee Y_5$
$Y_3 \Rightarrow Y_1 \vee Y_2$
$Y_3 \Rightarrow Y_8$
$Y_4 \Rightarrow Y_1 \vee Y_2$
$Y_4 \Rightarrow Y_6 \vee Y_7$
$Y_5 \Rightarrow Y_1 \vee Y_2$
$Y_5 \Rightarrow Y_8$
$Y_6 \Rightarrow Y_4$
$Y_7 \Rightarrow Y_4$
$Y_8 \Rightarrow Y_3 \vee Y_5 \vee (\neg Y_3 \wedge \neg Y_5)$
$Y_1 \veebar Y_2$
$Y_4 \veebar Y_5$
$Y_6 \veebar Y_7$

LogMIP INPUT FILE for this example

```
$TITLE APPLICATION OF THE LOGIC-BASED MINLP ALGORITHM IN EXAMPLE #3
* THE FORMULATION IS DISJUNCTIVE
$OFFSYMXREF
$OFFSYMLIST
*  SELECT OPTIMAL PROCESS FROM WITHIN GIVEN SUPERSTRUCTURE.
*
SETS   I      PROCESS STREAMS                / 1*25 /
       J      PROCESS UNITS                  / 1*8 /

PARAMETERS   CV(I)     VARIABLE COST COEFF FOR PROCESS UNITS - STREAMS
       /  3 = -10  ,  5 = -15  ,  9 = -40,   19 =  25  , 21 =  35  , 25 = -35
         17 = 80  , 14 =  15  , 10 =  15,    2 =  1   ,  4 =  1   , 18 = -65
         20 = -60  , 22 = -80  /;

 VARIABLES  PROF     PROFIT ;

 BINARY VARIABLES    Y(J)         ;
 POSITIVE VARIABLES  X(I) , CF(J);

 EQUATIONS
* EQUATIONS Independent of discrete choices
* ---------------------------------------------------------
 MASSBAL1, MASSBAL2, MASSBAL3, MASSBAL4, MASSBAL5, MASSBAL6, MASSBAL7, MASSBAL8
 SPECS1, SPECS2, SPECS3, SPECS4

* EQUATIONS allowing flow just IFF the unit EXISTS
* ------------------------------------------------
 LOGICAL1, LOGICAL2, LOGICAL3, LOGICAL4, LOGICAL5, LOGICAL6, LOGICAL7, LOGICAL8

* DISJUNCTION'S CONSTRAINTS and EQUATIONS
* ---------------------------------------
 INOUT11, INOUT12, INOUT13, INOUT14  INPUT-OUTPUT RELATIONS FOR PROCESS UNIT 1
 INOUT21, INOUT22, INOUT23, INOUT24  INPUT-OUTPUT RELATIONS FOR PROCESS UNIT 2
 INOUT31, INOUT32, INOUT34           INPUT-OUTPUT RELATIONS FOR PROCESS UNIT 3
 INOUT41, INOUT42, INOUT43, INOUT44, INOUT45                FOR PROCESS UNIT 4
 INOUT51, INOUT52, INOUT53, INOUT54  INPUT-OUTPUT RELATIONS FOR PROCESS UNIT 5
 INOUT61, INOUT62, INOUT63, INOUT64  INPUT-OUTPUT RELATIONS FOR PROCESS UNIT 6
 INOUT71, INOUT72, INOUT73, INOUT74  INPUT-OUTPUT RELATIONS FOR PROCESS UNIT 7
 INOUT81, INOUT82, INOUT83, INOUT84, INOUT85, INOUT86       FOR PROCESS UNIT 8
 OBJETIVO          OBJECTIVE FUNCTION DEFINITION ;

* BOUNDS SECTION:
 X.UP('3')  =  2.0 ;
 X.UP('5')  =  2.0 ;
 X.UP('9')  =  2.0 ;
 X.UP('10') =  1.0 ;
 X.UP('14') =  1.0 ;
 X.UP('17') =  2.0 ;
 X.UP('19') =  2.0 ;
 X.UP('21') =  2.0 ;
 X.UP('25') =  3.0 ;

*DEFINITIONS of  EQUATIONS Independent of discrete choices
 MASSBAL1..   X('13')          =E=  X('19') + X('21')           ;
 MASSBAL2..   X('17')          =E=  X('9') + X('16') + X('25') ;
 MASSBAL3..   X('11')          =E=  X('12') + X('15')           ;
 MASSBAL4..   X('3') + X('5') =E=  X('6') + X('11')            ;
 MASSBAL5..   X('6')           =E=  X('7') + X('8')             ;
 MASSBAL6..   X('23')          =E=  X('20') + X('22')           ;
 MASSBAL7..   X('23')          =E=  X('14') + X('24')           ;
 MASSBAL8..   X('1')           =E=  X('2') + X('4')             ;
 SPECS1..     X('10')  =L=  0.8 * X('17')                       ;
 SPECS2..     X('10')  =G=  0.4 * X('17')                       ;
 SPECS3..     X('12')  =L=  5.0 * X('14')                       ;
 SPECS4..     X('12')  =G=  2.0 * X('14')                       ;

* DEFINITION of EQUATIONS allowing flow just IFF the unit EXISTS
 LOGICAL1..   X('2') + X('3')   =L=  10. * Y('1')           ;
 LOGICAL2..   X('4') + X('5')   =L=  10. * Y('2')           ;
 LOGICAL3..   X('9')   =L=  10. * Y('3')                    ;
 LOGICAL4..   X('12') + X('14')  =L=  10. * Y('4')          ;
 LOGICAL5..   X('15')  =L=  10. * Y('5')                    ;
 LOGICAL6..   X('19')  =L=  10. * Y('6')                    ;
```

```
 LOGICAL7..   X('21')  =L=  10. * Y('7')                  ;
 LOGICAL8..   X('10') + X('17')  =L=  10. * Y('8')        ;

*DEFINITIONS of DISJUNCTION's EQUATIONS
 INOUT11..   EXP(X('3')) -1. =E= X('2')           ;
 INOUT14..   CF('1') =E= 5                        ;
 INOUT12..   X('2') =E= 0                         ;
 INOUT13..   X('3') =E= 0                         ;
 INOUT21..   EXP(X('5')/1.2) -1. =E= X('4')       ;
 INOUT24..   CF('2') =E= 8                        ;
 INOUT22..   X('4') =E= 0                         ;
 INOUT23..   X('5') =E= 0                         ;
 INOUT31..   1.5 * X('9') + X('10') =E= X('8')    ;
 INOUT34..   CF('3') =E= 6                        ;
 INOUT32..   X('9') =E= 0                         ;
 INOUT41..   1.25 * (X('12')+X('14')) =E= X('13') ;
 INOUT45..   CF('4') =E= 10                       ;
 INOUT42..   X('12') =E= 0                        ;
 INOUT43..   X('13') =E= 0                        ;
 INOUT44..   X('14') =E= 0                        ;
 INOUT51..   X('15') =E= 2. * X('16')             ;
 INOUT54..   CF('5') =E= 6                        ;
 INOUT52..   X('15') =E= 0                        ;
 INOUT53..   X('16') =E= 0                        ;
 INOUT61..   EXP(X('20')/1.5) -1. =E= X('19')     ;
 INOUT64..   CF('6') =E= 7                        ;
 INOUT62..   X('19') =E= 0                        ;
 INOUT63..   X('20') =E= 0                        ;
 INOUT71..   EXP(X('22')) -1. =E= X('21')         ;
 INOUT74..   CF('7') =E= 4                        ;
 INOUT72..   X('21') =E= 0                        ;
 INOUT73..   X('22') =E= 0                        ;
 INOUT81..   EXP(X('18')) -1. =E= X('10') + X('17');
 INOUT86..   CF('8') =E= 5                        ;
 INOUT82..   X('10') =E= 0                        ;
 INOUT83..   X('17') =E= 0                        ;
 INOUT84..   X('18') =E= 0                        ;
 INOUT85..   X('25') =E= 0                        ;

 OBJETIVO  .. PROF  =E= SUM(J,CF(J)) + SUM(I , X(I)*CV(I)) + 122 ;

 LOGIC EQUATION ATMOST1; ATMOST1.. Y('1') xor Y('2');
 LOGIC EQUATION ATMOST2; ATMOST2.. Y('4') xor Y('5');
 LOGIC EQUATION ATMOST3; ATMOST3.. Y('6') xor Y('7');

 LOGIC EQUATION IMP0; IMP0..  Y('1') -> Y('3') or Y('4') or Y('5');
 LOGIC EQUATION IMP1; IMP1..  Y('2') -> Y('3') or Y('4') or Y('5');
 LOGIC EQUATION IMP2; IMP2..  Y('3') -> Y('8')                     ;
 LOGIC EQUATION IMP3; IMP3..  Y('3') -> Y('1') or Y('2')           ;
 LOGIC EQUATION IMP4; IMP4..  Y('4') -> Y('1') or Y('2')           ;
 LOGIC EQUATION IMP5; IMP5..  Y('4') -> Y('6') or Y('7')           ;
 LOGIC EQUATION IMP6; IMP6..  Y('5') -> Y('1') or Y('2')           ;
 LOGIC EQUATION IMP7; IMP7..  Y('5') -> Y('8')                     ;
 LOGIC EQUATION IMP8; IMP8..  Y('6') -> Y('4')                     ;
 LOGIC EQUATION IMP9; IMP9..  Y('7') -> Y('4')                     ;

* BEGIN DECLARATIONS AND DEFINITIONS OF DISJUNCTIONS (LOGMIP Section)
$ONECHO > '%LM.INFO%'
 DISJUNCTION Y('1') INOUT11 INOUT14 ELSE INOUT12 INOUT13
 DISJUNCTION Y('2') INOUT21 INOUT24 ELSE INOUT22 INOUT23
 DISJUNCTION Y('3') INOUT31 INOUT34 ELSE INOUT32
 DISJUNCTION Y('4') INOUT41 INOUT45 ELSE INOUT42 INOUT43 INOUT44
 DISJUNCTION Y('5') INOUT51 INOUT54 ELSE INOUT52 INOUT53
 DISJUNCTION Y('6') INOUT61 INOUT64 ELSE INOUT62 INOUT63
 DISJUNCTION Y('7') INOUT71 INOUT74 ELSE INOUT72 INOUT73
 DISJUNCTION Y('8') INOUT81 INOUT86 ELSE INOUT82 INOUT83 INOUT84 INOUT85

* optional, if not set LOGMIP will find the modeltype suitable
 MODELTYPE MINLP
$OFFECHO* end logmip section

 OPTION EMP=LogMIP;
 OPTION OPTCR=0.0;
 MODEL EXAMPLE3  / ALL / ;
 SOLVE EXAMPLE3 USING EMP MINIMIZING PROF   ;
```

This is the new way of writing the Logic Propositions to establish relationships between the disjunctions terms

Each line of this section defines a new disjunction. For this purpose, explicit binary variables are used. The use of those is mandatory when logic propositions are defined

OBSERVE the model in this case is non-linear, it is optional to use the *modeltype* sentence to describe the model nature

3. How to write in GAMS a disjunctive model for LogMIP

The steps to write a problem into GAMS are the following:

a) Write in a GAMS input file (extension .gms) the sets, scalars, parameters, variables, equations and constraints, and any other component necessary for the problem, like if you were writing an algebraic problem.

- You must be familiar with GAMS notation to do so.
- You must also declare and define in this section the equations and constraints employed in disjunction terms.

b) If you are not going to define disjunctions over a domain, write in the same GAMS input file the sentences:

$ONECHO > "%lm.info%"

$OFFECHO

the dollar sign must be in the 1st column. You must write all three keywords, between these two sentences you must include disjunction definitions according to the rules of LogMIP language which have changed to be included in the EMP Environment.

The syntax to write a disjunction is the following:

DISJUNCTION [chull [chull eps] | bigM [bigM Mvalue] | indic]
[Not] Var|* { equ }
{ELSEIF [Not] Var | * {equ}}
[ELSE {equ}]

According to the GAMS syntax rules the meanings of some symbols are the following:

[ ] enclosed construct are optional
{ } enclosed construct may be repeated zero or more times
| or

DISJUNCTION is a mandatory word, after that you have three optional constructs:

[chull [chull eps] | bigM [bigM Mvalue] | indic]

Which are related to the relaxation and transformation of disjunctions:

- chull (convex hull)
- bigM (big M relaxation)
- indic (indicator constraint)

In this version, you can choose among different relaxations for each disjunction.
The default option is the convex hull.
The convex hull and the bigM relaxations also have additional optional definitions:

- For the convex hull, the epsilon (eps) parameter is an upper bound value to check for constraint satisfaction (it has a default value).
- In the case of BigM relaxation, the Mvalue to be defined is the value of the M parameter, which should be large enough to relax the constraint. It also has a default value, but it is important to change to avoid infeasible solutions, for those cases where that value is not appropriate.

The next specification is the variable to handle the disjunction term, by means of the following construct:

[Not] Var | *

You must specify **Var** **or** ***. The first option is to replace Var with a binary variable name defined in the GAMS section; * is employed when the variable name is assigned by GAMS.
NOTE THAT: when using the Var option, make sure to write at least a **dummy equation** that uses them in order to avoid the GAMS compiler take out from the model **if they are not used in other equation/constraint of the model**.
[Not] is the negation of the variable, by means of this sentence the disjunction term is satisfied using the FALSE value, instead of the TRUE value of the variable.

{ equ } represents a set of constraint names (previously defined in the GAMS section) that must be satisfied if the FIRST disjunction term is selected.

For the definition of a several terms disjunction you must also add the following construct:

{ELSEIF [Not] Var | * {equ}}

where **ELSEIF** is a mandatory word and then for each term you must specify a binary variable name (**Var**) or ***, and also the constraints set to be satisfied ({ equ})

For the definition of a two terms disjunction using just one variable to handle both terms, you must also add the following construct:

[ELSE {equ}]

where **ELSE** is a obligatory word, followed by the set of constraint to be satisfied if the term is selected.

NOTE THAT since one variable is used to handle both terms the construct **Var** or ***** is not needed in the ELSE sentence.

Examples:
From Small Example 1 – First version:

disjunction Y1 equat4 else equat5

Corresponds to a two term disjunction having the following syntax rule:

DISJUNCTION Var { equ } ELSE {equ}

From Small Example 1 – Second version:

disjunction * equat4 else equat5

Corresponds to a two term disjunction having the following syntax rule:

DISJUNCTION * { equ } ELSE {equ}

From Small Example 2:

disjunction y1 equat1 equat2 elseif y2 equat3 equat4

Corresponds to a two term disjunction having the syntax rule of a several term, with the following syntax:

DISJUNCTION Var { equ } ELSEIF Var {equ}

c) Defining disjunctions over a domain

The definition of a disjunction over a domain is performed using the put writing facilities of GAMS (for more references about this topic read chapter "The Put Writing Facility" in GAMS User's Guide). The domain expansion is made via FILE and PUT sentences in combination with LOOP and/or WHILE GAMS sentences. To control disjunction´s domain, you must use the dollar sign ($)(for more references read section "The Dollar Condition" in GAMS User's Guide). In the following paragraphs, the definition of disjunctions over a domain is illustrated using some examples.

The following sentences are extracted from Third version of Small Example 1 in page 6 of this manual:

```
1  file lg / '%lm.info%' /;
2  put lg '* problem %gams.i%';
3  loop(lt(j,jj) )$(ord(j) <> ord(jj),
4       put / 'disjunction ' Y(j,jj)  seq(j,jj) 'else' seq(jj,j));
5  putclose;
```

The first line defines a **FILE** where lg is an internal name for GAMS to refer to an external file, the name of the external file is **%lm.info%** which is the default name used for LogMIP info file.

The second line (**put lg**) writes on the file a comment (*** problem %gams.i%**).

The third line is a **loop** control sentence, the controlling domain is given by **lt(j,jj)**, meaning that the loop will be executed over each member of the subset lt(j,jj) but just for those where the order of j is different to the order of jj, which is specified by the sentence **$(ord(j) <> ord(jj)).**

Line 4 writes in file lg a sentence containing the disjunction definition, suppose j is defined over domain 1,2 and 3, and that the order of j is 1 and 2, the consequence of executing line 4, the FILE lg will have the following line written:

```
Disjunction Y('1', '2')  seq ('1', '2') else seq('2','1')
```

In the next iteration of the loop sentence, the order of j changes to 3, then it writes a new line in the file,

```
Disjunction Y('1', '3')  seq ('1', '3') else seq('3','1')
```

This process continue until each one the elements of lt(j,jj) is covered

Please note that a new line character (**/)** is inserted in line 4; if that character is not placed in the sentence, the previous lines would be written one next to the other, in this way:

```
Disjunction Y('1', '2')  seq ('1', '2') else seq('2','1') Disjunction Y('1', '3') seq ('1', '3') else seq('3','1')
```

So, to avoid errors writing long lines it is important to include the line feed sentence (/) at the end of each line.

The next example corresponds to a mid-term contracts signature with suppliers, for more references about this problem visit www.minlp.org in the GDP problems section. The following sentences are extracted from that example.

```
1   file logMIP /"%LM.info%"/;
2   put logmip '* input=%gams.input%'
3   put / 'default  BigM' /
4
5   loop((j,t)$(ord(j)=1 or ord(j)=4 or ord(j)=6),
6      put /' disjunction *' ubb1(j,t) costbct1(j,t)
7           ' elseif * '  ubb2(j,t)   costbct2(j,t)
8           ' elseif * ' ubbno(j,t)  costbno(j,t)      ;
9    );
10
11  loop(j$(ord(j)=1 or ord(j)=4 or ord(j)=6),
12      put /' disjunction *'  lb1(j,'1')   costplct1(j,'1')
13           ' elseif * '  lb21(j,'1')  lb22(j,'2')  costplct21(j,'1') costplct22(j,'2')
14           ' elseif * '  lb31(j,'1')  lb32(j,'2')  lb33(j,'3')  costplct31(j,'1')
15                        costplct32(j,'2') costplct33(j,'3')
16           ' elseif * '  lengno1(j,'1')  lengno2(j,'2')  lengno3(j,'3') costno1(j,'1')
17                        costno2(j,'2')  costno3(j,'3')  ;
18    );
```

In line 1, it can be seen that the GAMS internal name of the file in this case is LogMIP. The third line define that this problem is solved using the bigM relaxation.
The first disjunction definition starts in line 5, it is defined over the domain of sets j and t. Set j is controlled by the elements of order 1, 4 and 6. The disjunction is expanded for the complete set t. Suppose that the domain of t is defined over 1 and 2, then the loop sentence posed from lines 5 to 9 writes in the file the following sentences:

```
disjunction * ubb1('1', '1') costbct1('1', '1') elseif * ubb2('1', '1') costbct2('1', '1')  elseif * ubbno('1', '1')  costbno('1', '1');
disjunction * ubb1('1', '2') costbct1('1', '2') elseif * ubb2('1', '2') costbct2('1', '2')  elseif * ubbno('1', '2')  costbno('1', '2');
disjunction * ubb1('4', '1') costbct1('4', '1') elseif * ubb2('4', '1') costbct2('4', '1')  elseif * ubbno('4', '1')  costbno('4', '2');
disjunction * ubb1('4', '2') costbct1('4', '2') elseif * ubb2('4', '2') costbct2('4', '2')  elseif * ubbno('4', '2')  costbno('4', '2');
disjunction * ubb1('6', '1') costbct1('6', '1') elseif * ubb2('6', '1') costbct2('6', '1')  elseif * ubbno('6', '1')  costbno('6', '1');
disjunction * ubb1('6', '2') costbct1('6', '2') elseif * ubb2('6', '2') costbct2('6', '2')  elseif * ubbno('6', '2')  costbno('6', '2');
```

The second disjunction definition starts in line 11, although it is also delimited by sets j and t, in this case only j is controlled in the loop sentence, the elements of set t are explicitly included in the constraints enumeration inside the disjunctions. Again the lines that this sentence writes in the file are the following:

```
disjunction * lb1('1','1')  costplct1('1','1')  elseif *lb21('1','1')  lb22('1','2')  costplct21('1','1') costplct22('1','2')
elseif  *   lb31('1','1')   lb32('1','2')   lb33('1','3')   costplct31('1','1')   costplct32('1','2')  costplct33('1','3')  elseif  *
lengno1('1','1')  lengno2('1','2')  lengno3('1','3') costno1('1','1')  costno2('1','2')  costno3('1','3')  ;
disjunction * lb1('2','1')  costplct1('2','1') elseif *  lb21('2','1')  lb22('2','2')  costplct21('2','1') costplct22('2','2')
elseif  *   lb31('2','1')   lb32('2','2')   lb33('2','3')   costplct31('2','1')   costplct32('2','2')  costplct33('2','3')  elseif  *
lengno1('2','1')  lengno2('2','2')  lengno3('2','3') costno1('2','1')  costno2('2','2')  costno3('2','3')  ;
disjunction * lb1('3','1')  costplct1('3','1') elseif *  lb21('3','1')  lb22('3','2')  costplct21('3','1') costplct22('3','2')
elseif  *   lb31('3','1')   lb32('3','2')   lb33('3','3')   costplct31('3','1')   costplct32('3','2')  costplct33('3','3')  elseif  *
lengno1('3','1')  lengno2('3','2')  lengno3('3','3') costno1('3','1')  costno2('3','2')  costno3('3','3')  ;
```

3.1. Controlling the disjunction's and constraint domains.

The domain where disjunctions and constraints must be satisfied must be controlled via a loop sentence in combination with dollar operator ($), this operator must be used in the same way than in GAMS constraints definition.

3.2. Use of a DUMMY equation

Although it is not mandatory, we recommend to write a dummy equation into the GAMS section for the binary variables that handle disjunction terms (disjunction conditions) previously defined in the GAMS section. The purpose of this dummy equation is to avoid that GAMS compiler eliminate them from the model (and from the matrix). It occurs when some or all variables are not used in other constraints in the model. Suppose the following variables handling disjunction's terms defined in GAMS section:

```
Binary variables Y(J);
```

If some or all variables of Y are not included in any equation or constraint defined in GAMS section, they will be eliminated from the model, and LogMIP compiler will show an error even when they handle disjunction terms. To avoid that, you must write the following constraint:

```
DUMMY.. SUM(J, Y(J)) =G= 0;
```

which should be always satisfied. Another example:

```
Binary variables y, w, z;
DUMMY ..  y + w + z =G=0;
```

NOTE THAT this is not needed when you use the * option (default variable names) to handle disjunction terms.

4. Logic Propositions

Logic propositions are used to pose relationships between the Boolean (Binary) variables handling the disjunctive terms. Logic Propositions must be declared and defined in the GAMS Section.

4.1. Declaration Sentence

LOGIC EQUATION name

LOGIC EQUATION is a reserved word to specify a logic proposition, name must be provided by the user, it must follow the rules of any constraint name.

4.2. Definition Sentence

The definition of a LOGIC EQUATION is similar to any other equation in the GAMS model (see Chapter 8. Equations in GAMS Users Guide), with the difference that is must include only the following operators:

Operator Symbol	Operation
->	Implication
<->	equivalence
not	negation
and	logical and
or	logical or
xor	exclusive or

Examples:

Declaration Sentence

LOGIC EQUATION ATMOST1, ATMOST2, ATMOST3, IMP0, IMP1, IMP2;

Definition Sentences

ATMOST1.. Y('1') xor Y('2');
ATMOST2.. Y('4') xor Y('5');
ATMOST3.. Y('6') xor Y('7');
IMP0.. Y('1') -> Y('3') or Y('4') or Y('5');
IMP1.. Y('2') -> Y('3') or Y('4') or Y('5');
IMP2.. Y('3') -> Y('8')

5. SOLVERS

✓ **LogMIP can solve linear/nonlinear disjunctive hybrid models that follow the formulation showed in section 2 of this manual. Disjunctive models are those where discrete decisions are written only in the form of disjunctions, while hybrid models involve both disjunctions and mixed-integer constraints.**

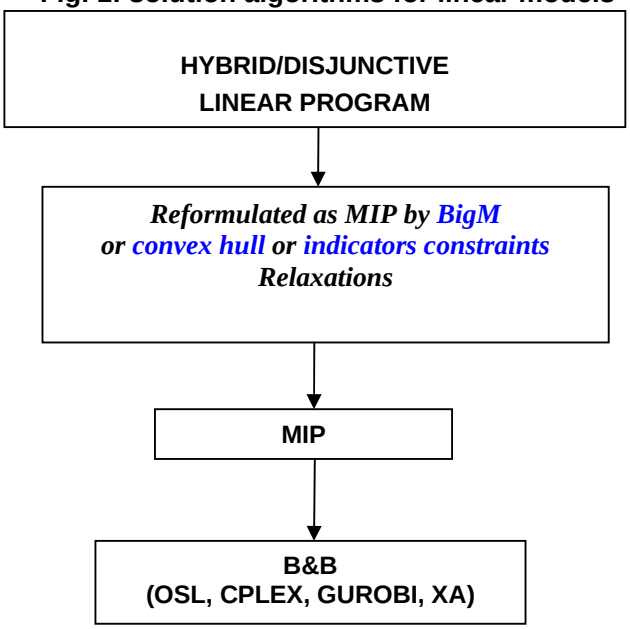
According to the user model type (linear or non-linear), by default, LogMIP decides the solver to run. The user can also specify the model type by including the following sentence in the LogMIP section:

modeltype [MIP|MINLP]

5.1. Solution algorithm for linear problems

Figure 1 shows how the solution for linear hybrid/disjunctive models is driven.

Fig. 1: solution algorithms for linear models



The disjunctions defined in the model are transformed into mixed integer formulations by using one of the relaxations proposed: BigM or convex hull or indicators constraints. The complete set of disjunctions can be transformed by one of those relaxations, or you can choose a different one for each disjunction in the model. Then, the problem is converted into a Mixed Integer Program (MIP) which is later solved by a Branch and Bound algorithm. References about the relaxations can be found in Balas(1979), Vecchietti and Grossmann(2002).

The default relaxation is the convex hull. You can change it by introducing in the LogMIP section the following sentence:

DEFAULT Big-M

By means of this sentence disjunctions are relaxed using the Big-M relaxation.

Since LogMIP 2.0 belongs to the EMP environment, to solve the problem you must write in the GAMS input file the following two sentences:

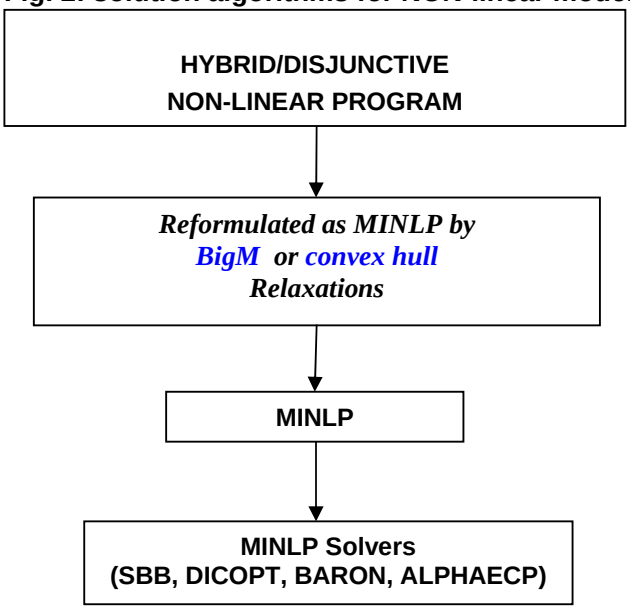
OPTION EMP=LOGMIP;

SOLVE *modelname* USING EMP [MINIMIZING | MAXIMIZING] *variablename*

5.2. Solution algorithms for NON-Linear problems

Figure 2 shows the flowchart to solve non-linear hybrid/disjunctive models.

Fig. 2: solution algorithms for NON-linear models



The disjunctions defined in the model are transformed into mixed integer formulations by using one of the relaxations proposed: BigM or convex hull for non-linear problems. The complete set of disjunctions can be transformed by one of those relaxations, or you can choose a different one for each disjunction in the model. Then the problem is converted into a Mixed Integer Non-Linear Program which is later solved by a MINLP solver such that: SBB, DICOPT, BARON, AlphaECP, etc.

Logic Based Outer Approximation algorithm (Turkay and Grossmann, 1996a) does not work anymore for this LogMIP version. A new system is developed, it will be ready for a next release.

The default relaxation is the convex hull. You can change it by introducing in the LogMIP section the following sentence:

DEFAULT Big-M

By means of this sentence disjunctions are relaxed using the Big-M relaxation.

Since LogMIP 2.0 belongs to the EMP environment, to solve the problem you must write in the GAMS input file the following two sentences:

OPTION EMP=LOGMIP;

SOLVE *modelname* USING EMP [MINIMIZING | MAXIMIZING] *variablename*

6. Recommendations and Limitations.

- Write the GAMS file in a single way following a sequence: declare SETS, VARIABLES and EQUATIONS at the beginning of the file, then pose the constraint, objective function and disjunctions definitions. Finally write the options, model and solution sentences.

- Although GAMS is flexible about the declarations of the equation and variable domains (you can declare them or not), it is strongly recommended to explicitly declare all domains for every variable and constraint defined in the model.

- If possible, write your entire model in a single file, do not use the INCLUDE sentence to import an external file in the model.

- Note that constraints defined in the disjunctions are related with your declaration and definitions in the GAMS section. In this sense you cannot include in the disjunction the name of a constraint not previously defined. This is especially important for constraints defined in the GAMS section over a domain controlled by the dollar sign ($). Disjunctive constraints must be in concordance with those defined in GAMS section.

- A similar advice is necessary for variables handling disjunction´s terms.

7. References.

The following is a list of articles where you can get a more complete material about disjunctive/hybrid models and the algorithms to solve them.

Balas, E.
"Disjunctive programming". Discrete Optimizations II, Annals of Discrete Mathematics, 5, North Holland, Amsterdam, **1979**.

Balas, E.
"*Disjunctive Programming and a hierarchy of relaxations for discrete optimization problems",* SIAM J. Alg. Dis. Meth., 6 (3), 466-485, **1985**.

Balas, E.
"*Disjunctive Programming: Properties of the convex hull of feasible points",* Discrete Applied Mathematics 89 (1), 3-44, **1998**.

Brooke A., Kendrick D. and Meeraus A.
"GAMS a User's Guide". Gams Development Corporation, **1996**.

Gil J.J. and Vecchietti A.
"Issues about the development of a disjunctive program solver". Proceedings of Enpromer , I ,403-409, **2000.**

Gil J.J. and Vecchietti A.
"Using design patterns for a compiler modeling for posing disjunctive optimization programs".
Proceedings of 31 JAIIO, September 2002, Santa Fe Argentina.

Grossmann I.E.
"Mixed-Integer Optimization Techniques for Algorithmic Process Synthesis", Advances in Chemical Engineering, Vol. 23, Process Synthesis, pp.171-246, **1996**.

Lee S. and Grossmann I.E.
*"New algorithm for Nonlinear Generalized Disjunctive Programming".*Comp. Chem. Eng. , 24 (9-10), 2125-2141, **2000**.

Lee, S. and I.E. Grossmann,
"Logic-based Modeling and Solution of NonlinearDiscrete/Continuous Optimization Problems," Annals of Operations Research: State of the Art and Recent Advances in IntegerProgramming*,* 139, 267-288*,* **2005***.*

Raman R. and Grossmann I.E.
"Modeling and Computational Techniques for Logic Based Integer Programming". Comp. Chem. Eng., 18 (7), 563-578, **1994**.

Sawaya, N.W. and Grossmann I.E.
"A Cutting Plane Method for Solving Linear Generalized Disjunctive Programming Problems," Computers and Chemical Engineering, 29, 1891-1913, **2005***.*

Sawaya, N.W. and Grossmann I.E.
"Computational Implementation of Non-Linear Convex Hull Reformulation," Computers & Chemical Engineering, 31, 856-866*,* **2007***.*

Turkay M. and Grossmann I.E.
"Logic-Based Algorithms for the Optimal Synthesis of Process Networks". Comp. Chem. Eng., 20 (8), 959-978, **1996**.

Vecchietti A. and Grossmann I.E.
"*LOGMIP: A Disjunctive 0-1 Nonlinear Optimizer for Process System Models"*. Comp. Chem. Eng., 23,. 555-565, **1999**.

Vecchietti A. and Grossmann I.E.
"Modeling issues and implementation of language for disjunctive programming". Comp. & Chem. Eng, 24, 2143-2155, **2000**.

Vecchietti, A., S. Lee and I.E. Grossmann
"Modeling of Discrete/ContinuousOptimization Problems: Characterization and Formulation of Disjunctions and their Relaxations," Computers and Chemical Engineering 27, 433-448 **2003.**

Vecchietti A, and Grossman I.E.,
"Computational Experience with LogMIP Solving Linear and Nonlinear Disjunctive Programming Problems," Proceedings of the Sixth International Conference on Foundation of Computer Aided Process Design (FOCAPD 2004), p. 587-590 **2004**.