LogMIP

Logical Mixed Integer Programming

Chemical Engineering

Carnegie Mellon

ingar

USER'S MANUAL

Aldo Vecchietti
aldovec@santafe-conicet.gov.ar

INDEX

1. Introduction

LogMIP 1.0 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 modeling, these problems have a rich area of applications.

LogMIP 1.0 has been developed by A. Vecchietti, J.J. Gil and L. Catania at INGAR (Santa Fe-Argentina) and Ignacio E. Grossmann at Carnegie Mellon University (Pittsburgh-USA).

LogMIP is composed of:

- a **language** compiler for the declaration and definition of disjunctions and logic constraints,
- **solvers** for linear and non-linear disjunctive models.

Those components are linked to **GAMS**. Both parts are supersets of GAMS language and solvers respectively. LogMIP is not independent of GAMS. Besides the disjunction and logic constraints declaration and definition, LogMIP needs the declaration and definitions of scalars, sets, tables, variables, constraints, equations, etc. made in GAMS language for the specifications and solution of a disjunctive problem.

2. Disjunctive model formulation

The models for LogMIP have the following general formulation:

$$
\begin{aligned}
& \min \ Z = \sum_k c_k + f(x) + d^t y \\
& s.t. \\
& \qquad g(x) \le 0 \\
& \qquad r(x) + D\,y \le 0 \\
& \qquad A\,y \le a \\
& \qquad \bigvee_{i \in D_k} \begin{bmatrix} Y_{ik} \\ h_{ik}(x) \le 0 \\ c_k = \gamma_{ik} \end{bmatrix} \quad k \in SD \\
& \qquad \Omega(Y) = True \\
& x \in R^n,\ y \in \{0,1\}^q,\ Y \in \{True, False\}^m, c_k \ge 0
\end{aligned}
$$

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.

We present three small examples in order to illustrate the meaning of the previous disjunctive/hybrid formulation. The first two corresponds to linear models, the later to a nonlinear disjunctive model.

2.1. SMALL EXAMPLE 1

$$\min Z = T$$

$$\begin{aligned} s.a. \quad & T \geq x_A + 8 \\ & T \geq x_B + 5 \\ & T \geq x_C + 6 \end{aligned}$$

$$\begin{bmatrix} Y_1 \\ x_A - x_C + 5 \leq 0 \end{bmatrix} \vee \begin{bmatrix} \neg Y_1 \\ x_C - x_A + 2 \leq 0 \end{bmatrix}$$

$$\begin{bmatrix} Y_2 \\ x_B - x_C + 1 \leq 0 \end{bmatrix} \vee \begin{bmatrix} \neg Y_2 \\ x_C - x_B + 6 \leq 0 \end{bmatrix}$$

$$\begin{bmatrix} Y_3 \\ x_A - x_B + 5 \leq 0 \end{bmatrix} \vee \begin{bmatrix} \neg Y_3 \\ x_B - x_A \leq 0 \end{bmatrix}$$

$$T, x_A, x_B, x_C \geq 0$$

$$Y_k \in \{true, false\}, k = 1,2,3.$$

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 6 binary variable

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

EQUAT1..  T =G= X('A') + 8;
EQUAT2..  T =G= X('B') + 5;
EQUAT3..  T =G= X('C') + 6;
EQUAT4..  X('A')-X('C')+ 5 =L= 0;
EQUAT5..  X('C')-X('A')+ 2 =L= 0;
EQUAT6..  X('B')-X('C')+ 1 =L= 0;
EQUAT7..  X('C')-X('B')+ 6 =L= 0;
EQUAT8..  X('A')-X('B')+ 5 =L= 0;
EQUAT9..  X('B')-X('A') =L= 0;
DUMMY..  SUM(I, Y(I)) =G= 0;
OBJECTIVE.. Z =E= T;

X.UP(J)=20.;
```

GAMS components declaration section.

Constraints independent of discrete choices (disjunctions) (EQUAT1–EQUAT3)

Constraints for discrete choices (disjunctions) (EQUAT4–EQUAT9)

GAMS equations and constraints definition.

Constraint definitions corresponding to disjunction terms are defined here.

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

```
$ONECHO > "%lm.info%"
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;
$OFFECHO

OPTION MIP=LMCHULL;
MODEL PEQUE1 /ALL/;
SOLVE PEQUE1 USING MIP MINIMIZING Z;
```

**In this section are defined the disjunctions according to the syntax defined for LogMIP.
This section is compiled by LogMIP and ignored by GAMS.**

LMCHULL is the solver, which generates a MIP problem by applying the convex hull relaxation of a disjunctive set. Then a conventional B&B algorithm solves the MIP GAMS Input file generated by the application.

Second Version – Using 3 binary variable

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

GAMS components declaration section.

```
EQUAT1..  T =G= X('A') + 8;
EQUAT2..  T =G= X('B') + 5;
EQUAT3..  T =G= X('C') + 6;
EQUAT4..  X('A')-X('C')+ 5 =L= 0;
EQUAT5..  X('C')-X('A')+ 2 =L= 0;
EQUAT6..  X('B')-X('C')+ 1 =L= 0;
EQUAT7..  X('C')-X('B')+ 6 =L= 0;
EQUAT8..  X('A')-X('B')+ 5 =L= 0;
EQUAT9..  X('B')-X('A') =L= 0;
DUMMY..  SUM(I, Y(I)) =G= 0;
OBJECTIVE.. Z =E= T;

X.UP(J)=20.;
```

Constraints independent of discrete choices (disjunctions) (EQUAT1–EQUAT3)

Constraints for discrete choices (disjunctions) (EQUAT4–EQUAT9)

GAMS equations and constraints definition.

Constraint definitions corresponding to disjunction terms are defined here.

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

```
$ONECHO > "%lm.info%"
DISJUNCTION D1,D2,D3;

 D1 IS
IF Y('A') THEN
       EQUAT4;
ELSE
       EQUAT5;
ENDIF;

 D2 is
IF Y('B') THEN
       EQUAT6;
ELSE
       EQUAT7;
ENDIF;

 D3 IS IF Y('C') THEN
       EQUAT8;
ELSE
       EQUAT9;
ENDIF;

$OFFECHO

OPTION MIP=LMBIGM;
OPTION LIMCOL=0;
OPTION LIMROW=0;
OPTION OPTCR=0.0;
MODEL PEQUE1 /ALL/;
SOLVE PEQUE1 USING MIP MINIMIZING Z;
```

LMBIGM is the solver, which generates a MIP problem by applying the Big-M relaxation of a disjunctive set. Then a conventional B&B algorithm solves the MIP GAMS Input file generated by the application.

Third Version – Alex Meeraus compact version

```
$TITLE LOGMIP USER'S MANUAL SMALL EXAMPLE 1
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 J BEORE JJ
                        DUMMY      FORCE NAMES INTO MODEL;


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 M / ALL /;

$ONECHO > "%lm.info%"
DISJUNCTION D(J,JJ);

D(J,JJ) WITH ORD(J) < ORD(JJ) IS
IF Y(J,JJ)
   THEN SEQ(J,JJ);
   ELSE SEQ(JJ,J);
ENDIF;

$OFFECHO
OPTION MIP=LMBIGM;

SOLVE M USING MIP MINIMIZING T;
```

2.2. SMALL EXAMPLE 2

$\min c + 2x_1 + x_2$ s.a.: $\begin{bmatrix} Y_1 \\ -x_1 + x_2 + 2 \le 0 \\ c = 5 \end{bmatrix} \vee \begin{bmatrix} Y_2 \\ 2 - x_2 \le 0 \\ c = 7 \end{bmatrix}$ $\begin{bmatrix} Y_3 \\ x_1 - x_2 \le 1 \end{bmatrix} \vee \begin{bmatrix} \neg Y_3 \\ x_1 = 0 \end{bmatrix}$ $Y_1 \wedge \neg Y_2 \Rightarrow \neg Y_3$ $Y_2 \Rightarrow \neg Y_3$ $Y_3 \Rightarrow \neg Y_2$ $0 \le x_1 \le 5, 0 \le x_2 \le 5, c \ge 0$ $Y_j \in \{true, false\}, j = 1,2,3$	**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

```
SET I /1*3/;
SET J /1*2/;
SCALAR M /100/;
BINARY VARIABLES Y(I);
POSITIVE VARIABLES X(J), C;
VARIABLE Z;
EQUATIONS EQUAT1, EQUAT2, EQUAT3, EQUAT4, EQUAT5, EQUAT6,
        DUMMY, OBJECTIVE;

EQUAT1..        X('2') =L= X('1') - 2;
EQUAT2..        C =E= 5;
EQUAT3..        X('2') =G= 2;
EQUAT4..        C =E= 7;
EQUAT5..        X('1')-X('2') =L= 1 ;
EQUAT6..        X('1') =E= M * Y('3');

DUMMY.. SUM(I, Y(I)) =G= 0;

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

```
$ONECHO > "%lm.info%"
        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') -> not Y('3');
        Y('2') -> not Y('3') ;
        Y('3') -> not Y('2') ;
$OFFECHO
```

OBSERVE the different syntax used to pose a two term disjunction where each term must satisfy a TRUE condition (handled by two different variables) against a two term disjunction with one TRUE term condition and the other with a FALSE one (handled by the same variable).

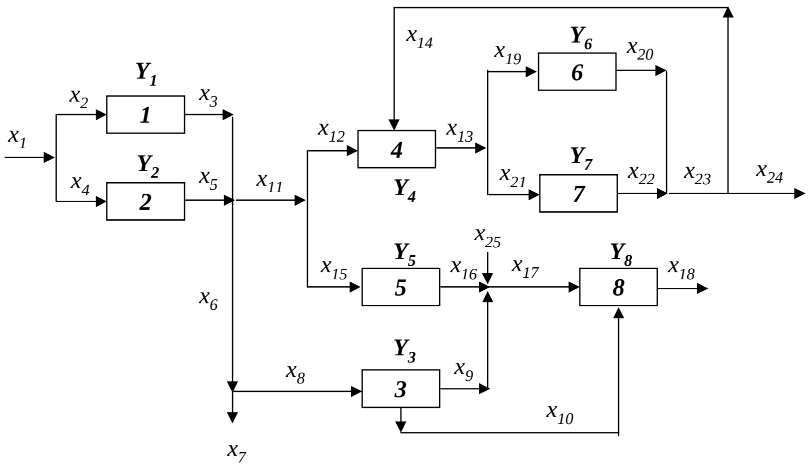
Logic Propositions to establish relationships between the disjunctions terms

```
OPTION MIP=LMBIGM
MODEL PEQUE2 /ALL/;
SOLVE PEQUE2 USING MIP MINIMIZING Z;
```

LMBIGM is the solver, which generates a MIP problem by applying the <u>BigM relaxation of a disjunctive set</u>. Then a conventional B&B algorithm solves the MIP GAMS Input file generated by the application.

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

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

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

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

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

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

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

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

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

Logic Propositions:

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

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 ;

 OPTIONS LIMCOL = 0   ;
 OPTION LIMROW = 0    ;
 OPTION OPTCR = 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 ;

* BEGIN DECLARATIONS AND DEFINITIONS OF DISJUNCTIONS (LOGMIP Section)
$ONECHO > "%lm.info%"

disjunction  d1, d2, d3, d4, d5, d6, d7, d8;

       d1 is  if Y('1') then
               INOUT11;
               INOUT14;
              else
               INOUT12;
               INOUT13;
              endif;

       d2 is  if Y('2') then
               INOUT21;
               INOUT24;
              else
               INOUT22;
               INOUT23;
              endif;

       d3 is if Y('3') then
               INOUT31;
               INOUT34;
              else
               INOUT32;
              endif;


       d4 is if Y('4') then
               INOUT41;
               INOUT45;
              else
               INOUT42;
```

```
                INOUT43;
                INOUT44;
               endif;

       d5 is if Y('5') then
                INOUT51;
                INOUT54;
               else
                INOUT52;
                INOUT53;
               endif;

       d6 is if Y('6') then
                INOUT61;
                INOUT64;
               else
                INOUT62;
                INOUT63;
               endif;

       d7 is if Y('7') then
                INOUT71;
                INOUT74;
               else
                INOUT72;
                INOUT73;
               endif;

       d8 is  if Y('8') then
                INOUT81;
                INOUT86;
               else
                INOUT82;
                INOUT83;
                INOUT84;
                INOUT85;
               endif;

 atmost(Y('1'), Y('2'));
 atmost(Y('4'), Y('5'));
 atmost(Y('6'), Y('7'));

 Y('1') -> Y('3') or Y('4') or Y('5');
 Y('2') -> Y('3') or Y('4') or Y('5');
 Y('3') -> Y('8');
 Y('3') -> Y('1') or Y('2');
 Y('4') -> Y('1') or Y('2');
 Y('4') -> Y('6') or Y('7');
 Y('5') -> Y('1') or Y('2');
 Y('5') -> Y('8');
 Y('6') -> Y('4');
 Y('7') -> Y('4');
```

Special sentences to establish relationships between Boolean variables

Logic Propositions to establish relationships between the disjunctions terms

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

$OFFECHO
* end logmip section

 option minlp=LMLBOA;

 MODEL EXAMPLE3  / ALL / ;
 SOLVE EXAMPLE3 USING MINLP MINIMIZING PROF   ;
```

This is a special section for non-linear problems. This is an initialization section needed by the LOGIC BASED OA algorithm. You must specify it. More references about this will be explained later, you can read also Turkay and Grossmann (1996a).

LMLBOA is the solver for non-linear problems, which applies the LOGIC_BASED OA algorithm. You need a NLP and a MIP solver installed together with GAMS to solve this problem.

3. How to write a disjunctive model for LogMIP

The algorithm to write a problem into GAMS is 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 a mathematical problem.

   You must be familiar with the GAMS notation to do so.

   You must even declare and define in this section the equations and constraints for the disjunction terms.

   You must declare and define binary variables to handle disjunction terms, these variables will work as Boolean variables. 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**.

b) 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 declarations and definitions.

   The complete section including the sentences above is a comment for GAMS compiler, so it is ignored by it. LogMIP language compiler compiles this section.

c) Write between the sentences of section b) the disjunction declaration and definitions according to the rules of **LogMIP language.**

4. Declaration and Definition of Disjunctions

To write disjunctions there are two types of sentences:

- declaration sentence
- definition sentence.

Disjunction can be <u>single or no-domain</u> or can have a <u>domain</u> such that you declare and define a set of disjunctions over the domain.
The domain of the disjunction can be limited by applying the sentence <u>with</u>.

4.1. Declaration Sentence

The declaration sentence use de word DISJUNCTION as token. The syntax is:

DISJUNCTION disjunction_identifier [domain_identifier, …, domain_identifier],
… , disjunction_identifier [domain_identifier, …, domain_identifier];

A disjunction name as well as the domain name can have up to 32 characters long and must start with a letter.

It follows GAMS rules about naming. You can have several entries for disjunction declarations.

Cannot use LogMIP reserved words, which are: disjunction, bu, card, else, elsif, eq, ge, if, initial, le, lt, ord, then, with

The disjunction declaration/definition over a domain is optional.

The domain must be previously defined in GAMS section declared as a SET or ALIAS.

✓ **You cannot define a domain inside the LOGMIP section. The reason is that the disjunction's domain must be in concordance of the constraint's domain, which is defined in the GAMS section.**

Disjunction identifiers must be unique it cannot be equal to any other identifier in the GAMS or LogMIP section.

Examples:

DISJUNCTION a, b(i,j), disjunctionnamelong, d2(j), D2_d2;

a, D2_d2 and disjunctionnamelong are single domain disjunctions.

b(i,j) and d2(j) are disjunctions defined over a domain, i and j are SETS or ALIAS defined in the GAMS section.

4.2. Definition sentence for single domain disjunctions

Given the following disjunction, which is a

single domain disjunction with two terms that must satisfy only one condition (first term:TRUE – second term: FALSE).

$$\begin{bmatrix} Y \\ CONSTRAINT3 \end{bmatrix} \vee \begin{bmatrix} \neg Y \\ CONSTRAINT4 \end{bmatrix}$$

The syntax for the definition is:

```
disjunction_identifier IS
        IF term_condition THEN
                constraint_identifier_1;
                …
                constraint_identifier_i;
                …
                constraint_identifier_n;
        ELSE
                constraint_identifier_n+1;
                …
                constraint_identifier_r;
                …
                constraint_identifier_z;
        ENDIF;
```

term_condition is the identifier of a binary variable defined in GAMS section which takes the role of a Boolean variable, (1:TRUE, 0:FALSE).

Declaration and definition of ***constraint_identifier_1...constraint_identifier_z*** must be performed in GAMS section.

Another example of single domain disjunction can be the following:

Single domain disjunction with two terms. Each term must satisfy a condition.

$$\begin{bmatrix} Y_1 \\ CONSTRAINT\,1 \\ EQUATION1 \end{bmatrix} \vee \begin{bmatrix} Y_2 \\ CONSTRAINT\,2 \\ EQUATION2 \end{bmatrix}$$

The syntax for this case is:

```
disjunction_identifier IS
        IF term_condition_1 THEN
                constraint_identifier_1;
                …
                constraint_identifier_n;
        ELSIF term_condition_2 THEN
                constraint_identifier_n+1;
                …
                constraint_identifier_z;
        ENDIF;
```

term_condition_1 and ***term_comdition_2*** are binary variables identifiers defined in GAMS section which takes the role of a Boolean variable, (1:TRUE, 0:FALSE). The identifiers must be different.

Declaration and definition of ***constraint_identifier_1...constraint_identifier_z*** are performed in GAMS section.

The same syntax can be applied for **single domain disjunction with several terms. Each term must satisfy a condition.**

The main difference is that having a disjunction with more than two terms implies that you must have several ELSIF sections for this case.

For both examples presented the following declarations and definitions apply.

```
$ONECHO > "%lm.info%"
Disjunction D1, D2;

D1 IS
        IF Y THEN
                CONSTRAINT3;
        ELSE
                CONSTRAINT4;
        ENDIF;
```

```
D2 IS
        IF Y1 THEN
                CONSTRAINT1;
                EQUATION1;
        ELSIF Y2 THEN
                CONSTRAINT2;
                EQUATION2;
        ENDIF;
$OFFECHO
```

✓ **It is not allowed in this release the capability of defining a disjunction with several terms (with each term satisfying a condition), and having an ELSE term, which applies when none of the previous term is TRUE.**

It is not allowed a disjunction defined like:
IF..THEN..ELSIF..THEN..ELSIF..THEN..**ELSE**..ENDIF

4.3. Definition sentence for disjunctions defined over a domain

As was mentioned in section 4.1. to define a disjunction over a domain you must first declare the domain (as a SET or ALIAS in GAMS section) and then declare the disjunction identifier over that domain.

The simplest definition can be a disjunction defined over the complete domain, for example, suppose the following disjunction:

$$\begin{bmatrix} Y_{ij} \\ CONSTRAINT\,1_{ij} \\ EQUATION1_{ij} \end{bmatrix} \vee \begin{bmatrix} \neg Y_{ij} \\ CONSTRAINT\,2_{ij} \\ EQUATION2_{ij} \end{bmatrix}$$

In order to define it **you must have in the GAMS section**:

Declaration of the domain:

```
SET I /1*3/
    J /1*4/;
```

Declaration of the constraints:

```
EQUATION
  CONSTRAINT1(I,J), CONSTRAINT2(I,J), EQUATION1(I,J), EQUATION2(I,J)
  DUMMY;
```

Declaration of variables for disjunction terms:

```
BINARY VARIABLES Y(I,J);
```

Definition of constraints:

```
CONSTRAINT1(I,J).. define constraint1 here;
CONSTRAINT2(I,J).. define constraint2 here;
EQUATION1(I,J).. define equation1 here;
EQUATION2(I,J).. define equation2 here;
```

Dummy equation if needed:
DUMMY.. SUM(I, SUM(J, Y(I,J))) =G= 0;

In the **LogMIP section you must have**:

```
$ONECHO > "%lm.info%"
Disjunction 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;

$OFFECHO
```

The previous define a total of 12 (3 times 4) disjunctions with two terms according to all possible combination of I times J.

Another illustrative example could be:

$$\begin{bmatrix} Y_{ij} \\ CONSTRAINT_{ij1} \\ CONSTRAINT_{ij2} \end{bmatrix} \vee \begin{bmatrix} \neg Y_{ij} \\ CONSTRAINT_{ij3} \\ CONSTRAINT_{ij4} \end{bmatrix}$$

In order to define it **you must have in the GAMS section**:

Declaration of domains:
```
SET I /1*3/
    J /1*4/;
ALIAS(J,K);
```

Declaration of constraints:
```
EQUATION
  CONSTRAINT(I,J,K), DUMMY;
```

Declaration of variables for the condition of disjunction terms:
BINARY VARIABLES Y(I,J);

Definition of constraints:
CONSTRAINT1(I,J,K).. define constraint here;

Dummy equation if needed:
DUMMY.. SUM(I, SUM(J, Y(I,J))) =G= 0;

In the **LogMIP section you must have**:

```
$ ONECHO > "%lm.info%"
Disjunction D(I,J);
D(I,J) IS
        IF Y(I,J) THEN
                CONSTRAINT(I,J,'1');
                CONSTRAINT(I,J,'2');
        ELSE
                CONSTRAINT(I,J,'3');
                CONSTRAINT(I,J,'4');
        ENDIF;

$OFFECHO
```

In the same way than the previous example, with the sentences above a total of 12 (3 times 4) disjunctions with two terms are defined according to all possible combinations of I times J.

✓ **You must be aware that when constraint's domains are expanded together the disjunction's domains, constraints must be previously defined in GAMS section. If the constraints are not defined over a particular domain, LogMIP reports an error.**

4.4. Controlling the disjunction's domain

The constraint and disjunction's domain can be controlled by the sentence **with** in conjunction with other operators:

- Relational operators:

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

- Logic 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**

- Using subsets.

If you have defined a disjunction over a domain and you have also the disjunction's variables and constraints defined over the same or different domain, you can have two situations, which are:

- Constraints and variables whose domain is under the control of the disjunction domain.
- Constraints and variables with uncontrolled domains, which must be defined in order to avoid semantic errors.

If you need to control a set domain that is already controlled by the disjunction definition, you must use an ALIAS for that set and redefine the domain over that ALIAS. See example 3 for this purpose.

In the following four sections some illustrative examples are presented with the intention of clarifying these matters.

4.4.1. Example 1

In this example, the disjunction domains are controlled using the *with*, *ord* and *card* clauses. The disjunction's variables and constraints domains do not need an extra control.

Given the following set definition in the GAMS Section:

SET I /1*3/ , J /1*4/;

and the following disjunction declaration and definition in LogMIP Section:

```
$ ONECHO > "%lm.info%"
Disjunction D(I,J);

D(I, J) with (ord(J) lt card (J)) IS
        IF Y(I,J) THEN
                CONSTRAINT(I,J);
                CONSTRAINT(I,J);
        ELSE
                CONSTRAINT(I,J);
                CONSTRAINT(I,J);
        ENDIF;
$OFFECHO
```

Only the domain of set J is controlled. The values of J must be less than the cardinality of J, meaning that only 1, 2 and 3 are permitted. The expansion of the previous definition renders 9 disjunctions (3 times 3): D('1','1'), D('1','2'), D('1','3'), D('2','1'), D('2','2'), D('2','3'), D('3','1'), D('3','2') and D('3','3').

✓ **Please note that although GAMS allows declaring variables and equations without a domain, and then in the definition use them with domains, LogMIP compiler is not aware about this situation and gives you an error or get stacked. We strongly suggest to explicitly declare all domains for every variable and constraint defined in the model.**

4.4.2. Example 2

In this example, the disjunction domains are controlled using the *with* and *ord* clauses. There exists a constraint with a domain not controlled by the disjunction that must be done in order to avoid semantic errors.

Given the following set definition in the GAMS Section:

SET I /1*3/ , J /1*4/, K/1*2/;

and the following disjunction declaration and definition in LogMIP Section:

```
$ ONECHO > "%lm.info%"
Disjunction D(i,j);

D(i,j) with (ord(i) < ord(j)) IS
IF Y(i,j) THEN
        CONSTR1(j);
        CONSTR2(i,j);
ELSE
        CONSTR3(j);
        CONSTR4(j,k) with (ord(k) ge 1);
ENDIF;

$OFFECHO
```

The previous example will generate the following disjunctions:

D('1', '2'), D('1', '3'), D('1','4'), D('2', '3'), D('2', '4') y D('3', '4').

✓ **Observe that CONSTR4 is defined over sets j and k. In this case j does not have problems since its domain is controlled by disjunction D. This is not the case of k, where you must include a with sentence to control the domain and avoid a semantic error.**

For the second term of the first disjunction D('1', '2') the following constraints must be satisfied: CONSTR3('2'), CONSTR4('2','1'), CONSTR4('2','2'), CONSTR4('2', '3').

4.4.3. Example 3

This example is a modification of the previous, for this case we need to control a domain set which is already controlled. An ALIAS definition in the GAMS section is needed.

Given the following definition in the GAMS Section

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

and the following disjunction declaration and definition in LogMIP Section:

```
$ ONECHO > "%lm.info%"
Disjunction D(i,j);

D(i,j) with (ord(i) < ord(j)) IS
IF Y(i,j) THEN
        CONSTR1(j);
        CONSTR2(i,jj) with (ord(jj) le 2);
ELSE
        CONSTR3(j);
        CONSTR4(j,k) with (ord(k) lt card(k));
ENDIF;

$OFFECHO
```

Since the domain j is controlled by the disjunction, and for CONSTR2 you need a different control, the ALIAS definition allow that you can define a *with* clause over it.
The previous example will generate the following disjunctions:

D('1', '2'), D('1', '3'), D('1','4'), D('2', '3'), D('2', '4') y D('3', '4').

Note that:

- for the first term of disjunction D('1','2') the following constraints must be satisfied: CONSTR1('2'), CONSTR2('1', '1') and CONSTR2('1', '2').
- for the first term of disjunction D('2', '3') the following constraints must be satisfied: CONSTRAINT1('3'), CONSTRAINT2('2', '1'), CONSTRAINT2('2', '2').
- and so on...

✓ **If you control the domain of a particular constraint with the sentence with you must be aware which domain expansion is assumed in order to avoid errors on it**

Observe that in the previous example it has been compared the order of a set item against an integer number, because the order is related to a position into the set, the position in which the item has been defined in the set.

4.4.4. Example 4 – Using a subset and the GAMS GDX facilities

Sometimes it is useful to restrict the disjunction domain by using a subset of the original set's domain. This can be done using the definition of a subset and the GDX facilities in the GAMS Section. This possibility is available for GAMS versions 20.5 and up. There exist two versions of this implementation depending on the GAMS version that is running in the system.

This method for controlling the set domain has the advantage of being the most general way of doing it, because you explicitly define the domain applicable for the disjunction.

Suppose the example 1 of section 4.4.1 that must be defined over the following domain of I,J: ('1','2') , ('2','3') , ('3','4') , to specify this you must define in the GAMS section the following:

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

and the following disjunction declaration and definition in LogMIP Section:

```
$ ONECHO > "%lm.info%"
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;
$OFFECHO
```

Note the sentence using the clause *with* in the definition of D(I,J) it is referencing the subset to define disjunctions D(‘1’,’2’), D(‘2’,’3’) and D(‘3’,‘4’).

If you are using GAMS IDE versions 20.5, 20.6, 20.7, 21.0, and 21.1 you must do the following:

1. Include in GAMS INPUT file the following sentences:
   $ GDXOUT filename
   $ UNLOAD
2. In the IDE GAMS parameters box (in the upper right section of the IDE window) write the following:
   gdx= filename

The first two sentences generate in GAMS compilation time a filename.gdx file containing the symbols of the subsets needed by LogMIP to proceed with the compilation. The sentence included in the IDE GAMS parameters box tells LogMIP the filename and file location in order to open it and read those symbols.

Both filenames must be the same.

For versions 21.2 and up the previous items 1. and 2. are not needed.

A larger example

The following example corresponds to a jobshop scheduling problem (Raman and Grossmann, 1994). In this problem, there is a set of jobs *i∈I* that must be processed in a sequence of stages but not all jobs require all stages. Zero wait transfer policy is assumed between stages. To obtain a feasible solution is necessary to eliminate all clashes between jobs. It requires that no two jobs be performed at any stage at the same time. This is expressed by the following disjunction:

$$\begin{bmatrix} Y_{ik} \\ t_i + \sum\limits_{\substack{m\in J(i)\\ m\le j}} \tau_{im} \le t_k + \sum\limits_{\substack{m\in J(k)\\ m< j}} \tau_{km} \end{bmatrix} \vee \begin{bmatrix} \neg Y_{ik} \\ t_k + \sum\limits_{\substack{m\in J(k)\\ m\le j}} \tau_{km} \le t_i + \sum\limits_{\substack{m\in J(k)\\ m< j}} \tau_{im} \end{bmatrix}$$

where t_i is the starting time of job i and τ_{ij} the processing time of job i in stage j. The meaning of (1) is that either the job i precede job k or viceversa in the stage j where a clash can occur. The objective is to minimize the makespan.

The subset is used to prevent clashes at stage j between job i and k. In the following we include GAMS file for a jobshop scheduling problem to illustrate how to use a subset to control the disjunction domain.

The LogMIP input file corresponding to this example is the following:

```
SETS  I  jobs  / A, B, C, D, E, F, G / ;
ALIAS(I,K);
SET      J  stages    / 1*5 /;
ALIAS(J,M);
*
* Subset L to prevent clashes at stage j between stage i and k
*
SET L(I,K,J) /A.B.3, A.B.5, A.C.1, A.D.3, A.E.3, A.E.5, A.F.1, A.F.3,
      A.G.5, B.C.2, B.D.2, B.D.3, B.E.2, B.E.3, B.E.5, B.F.3, B.G.2,
      B.G.5, C.D.2, C.D.4, C.E.2, C.F.1, C.F.4, C.G.2, C.G.4, D.E.2,
      D.E.3, D.F.3, D.F.4, D.G.2, D.G.4, E.F.3, E.G.2, E.G.5,
      F.G.4 /    ;


TABLE TAU(I,J) processing time of job i in stage j

             1       2       3       4       5
      A      3               5               2
      B              3       4               3
      C      6       3               6
      D              8       5       1
      E              4       6               2
      F      2               5       7
      G              8               5       4     ;

VARIABLES    MS      makespan ;
BINARY VARIABLES  Y(I,K,J)  sequencing variable between jobs i and k ;
POSITIVE VARIABLES T(I) ;

EQUATIONS
           FEAS(I)          makespan greater than all processing times
           NOCLASH1(I,K,J)   when i precedes k
           NOCLASH2(I,K,J)   when k precedes i
           DUMMY      ;

FEAS(I).. MS =G= T(I) + SUM(M,TAU(I,M))  ;

NOCLASH1(I,K,J)$((ORD(I) LT ORD(K)) AND L(I,K,J)) ..
               T(I) + SUM(M$(ORD(M) LE ORD(J)), TAU(I,M)) =L=
               T(K) + SUM(M$(ORD(M) LT ORD(J)), TAU(K,M));


NOCLASH2(I,K,J)$((ORD(I) LT ORD(K)) AND L(I,K,J)) ..
               T(K) + SUM(M$(ORD(M) LE ORD(J)),TAU(K,M)) =L=
               T(I) + SUM(M$(ORD(M) LT ORD(J)), TAU(I,M));

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

MODEL JOBSHOP / ALL / ;
```

```
$ ONECHO > "%lm.info%"

DISJUNCTION D1(I,K,J);

      D1(I,K,J) with ((ord(I) lt ord(K)) and L(I,K,J)) IS
      IF Y(I,K,J) THEN
        NOCLASH1(I,K,J);
      ELSE
        NOCLASH2(I,K,J);
      ENDIF;

$OFFECHO
T.up(I)=100.;

OPTION MIP      = LMBIGM;
OPTION OPTCR    = 0.0 ;
OPTION OPTCA    = 0.0 ;

SOLVE JOBSHOP MINIMIZING MS USING MIP ;
```

In the example shown above note that in LogMIP section disjunction D1 is defined over sets I,K,J their domain is controlled by the clause *WITH* using *ord* and *card* operators and the subset L, this is done in the same way than the definition of NOCLASH1 and NOCLASH2 constraints in GAMS section.

4. 5. Using the operator IN

Using the operator IN as follows can perform the same constraints domain limitation for the Example 3:

```
$ ONECHO > "%lm.info%"

Disjunction D(i,j);

D(i,j) with (ord(i) < ord(j)) IS
IF Y(i,j) THEN
        CONSTR1(j);
        CONSTR2(i,j);
ELSE
        CONSTR3(j);
        CONSTR4(j,k) with k IN ('1', '2');
ENDIF;

$OFFECHO
```

✓ The operator IN expands the domain just for the set items included between the parentheses after the operator. The set items must be enclosed by single quotation marks and separated by commas.

Another example of the IN operator is when you define the domain over a value range as follows:

```
$ ONECHO > "%lm.info%"

Disjunction D(i,j);

D(i,j) with (ord(i) < ord(j)) IS
IF Y(i,j) THEN
        CONSTR1(j);
        CONSTR2(i,j);
ELSE
        CONSTR3(j);
        CONSTR4(j,k) with k IN ('1'..'2');
ENDIF;

$OFFECHO
```

✓ **The above examples are just a few samples of what you can do for limiting the disjunction's domain. You can define more difficult sentences by using the operators mentioned and the logical operators and/or.**

Some other examples could be:

with (ord(j) lt card(i) and ord(k) not 1)

with (ord(l) lt ord(K)) and ((ord(l) eq 1 and ord(K) eq 3 and ord(j) eq 1) or

(ord(l) eq 1 and ord(K) eq 4 and ord(j) eq 4))

4.6. Use of a DUMMY equation

Although it is not mandatory, we recommend the user to write a dummy equation into the GAMS section for the binary variables that handle disjunction terms (disjunction conditions) in order to avoid that GAMS compiler eliminate those variables from the model (and from the matrix). It occurs when some or all variables are not used in other equations/constraints of 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 could be:

Binary variables y, w, z;

DUMMY .. y + w + z =G=0;

A combination of the previous examples could be:

DUMMY.. y + w + z + SUM(j, Y(j)) =G= 0;

5. Logic Propositions

Logic propositions are used to pose relationships between the Boolean (Binary) variables handling the disjunctive terms.

The operators defined for writing these sentences are:

Symbol	Meaning
->	Implication
<->	equivalence
not	negation
and	logical and
or	logical or

Every logic proposition must have an implication or equivalence operator, otherwise a syntax error will occur.

LogMIP transforms the logical propositions into a set of mathematical integer inequalities. Given the following set of logic propositions in LogMIP Section:

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

They are transformed into the following set of inequalities written in GAMS language syntax:

```
LOGPROP1.. -Y(1) +Y(2) -Y(3) =G= -1;
LOGPROP2.. -Y(2) -Y(3) =G= -1;
LOGPROP3..  -Y(1) +Y(3) +Y(4) +Y(5) =G= 0;
LOGPROP4..  -Y(2) +Y(3) +Y(4) +Y(5) =G= 0;
LOGPROP5..  -Y(3) +Y(8) =G= 0;
LOGPROP6..  -Y(3) +Y(1) +Y(2) =G= 0;
LOGPROP7..  -Y(5) +Y(8) =G= 0;
LOGPROP8..  -Y(8) +Y(5) =G= 0;
```

Then the disjunctive problem including this mathematical inequalities set is solved.

The user can avoid the writing of logic propositions into LogMIP section by introducing directly the mathematical equivalent inequalities in the GAMS Section, it depends on how confident it is with one methodology or the other.

6. Special Sentences

The special sentences are sentences to establish single relationships between the Boolean (binary variables) handling the disjunction terms. They represent an easy and more natural way than logic propositions to express the most common relationships between the Boolean/binary variables.

There are three kind of special sentences:

- atmost
- atleast
- exactly

The syntax of these sentences are:

atmost (Boolean/binary variable list separated by comma, [number])
atleast (Boolean/binary variable list separated by comma, [number])
exactly (Boolean/binary variable list separated by comma, [number])

- The meaning of **atmost** is that the summation of the variable list must be less than or equal to number.
- The meaning of **atleast** is that the summation of the variable list must be greater than or equal to number.
- The meaning of **exactly** is that the summation of the variable list must be equal to number.

[number] is optional, if no number is specified 1 is assumed.

The Boolean/binary variable list can include single variables and also a variable defined over a domain.

Example:
Given the following set of special sentences in LOGMIP section

```
atmost( Y('1'), Y('2'), Y('3'),2);
atleast(Y('1'), Y('2'), Y('3'), 3);
exactly(Y('1'), Y('2'), Y('3'));
```

They are transformed into the following set of mathematical inequalities:

```
LOGPROP1.. +Y(1) +Y(2) +Y(3) =L= 2;
LOGPROP2.. +Y(1) +Y(2) +Y(3) =G= 3;
LOGPROP3.. +Y(1) +Y(2) +Y(3) =E= 1;
```

7. LogMIP compilation errors

✓ **The syntax, semantic and some other errors detected during the disjunction's compilation phase are shown in the execution window of the IDE. They are not included in the listing file (.lst file).**

8. 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.**

8.1. Linear Solvers.

In figure 1 it is shown the solution algorithms for linear hybrid/disjunctive models.

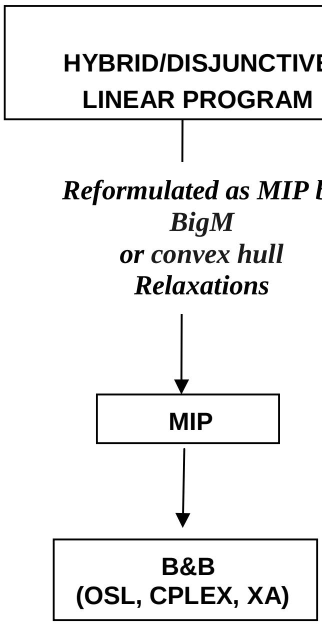

HYBRID/DISJUNCTIVE LINEAR PROGRAM
Reformulated as MIP by BigM or convex hull Relaxations
MIP
B&B (OSL, CPLEX, XA)

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. The original model is transformed into a Mixed Integer Model that it is later solved by a Branch and Bound algorithm. References about the relaxations can be found in Balas(1979), Vecchietti and Grossmann(2002).

Two solvers are available for linear solvers, which are:

- **LMBIGM**: applies the BigM relaxation of a disjunctive set.
- **LMCHULL**: applies the convex hull relaxation of a disjunctive set.

To select one solver you must write in the GAMS input file one of the following sentence:

OPTION MIP=LMBIGM;

or

OPTION MIP=LMCHULL;

See the small examples provided in Section 2 of this manual.

8.1.1. OPTIONS in LMBIGM solver

You have two options when selecting the BigM relaxation solver (LMBIGM). Both options are related to value of the parameter M in the relaxations. This value is very important in order to reach the solution. The parameters are set up by the keywords **DEFAULT** and **DETERMINEM** in the file "LMBIGM.opt".

The way LMBIGM uses this parameters is same than the other solvers in GAMS:

- In GAMS section and after defined the model you want to solve, you must write:
  model_name.optfile =1;
- In the directory where GAMS files are included, write a file whose name is **LMBIGM.opt** with the options you have selected.
- In LMBIGM.opt you must write the options according to the following rules: DEFAULT is the default value for the M parameter value. After the keyword DEFAULT you must write a real number representing the value of M of your choice. Examples: **DEFAULT 1000**, **DEFAULT 5.e5.** DETERMINEM is the option that can be turned off or turned on by writing 0 or 1 respectively, after the keyword DETERMINEM. If you turned it on, then **the solver can calculate the best value of M. For doing that it is very important to provide good bounds for all continuous variables** included in the model. Example: **DETERMINEM 0**.

The default values for this option are DEFAULT 1.E4 and DETERMINEM 0.

8.2. Nonlinear SOLVER

In figure 2 it is shown the solution algorithms for nonlinear hybrid/disjunctive models.

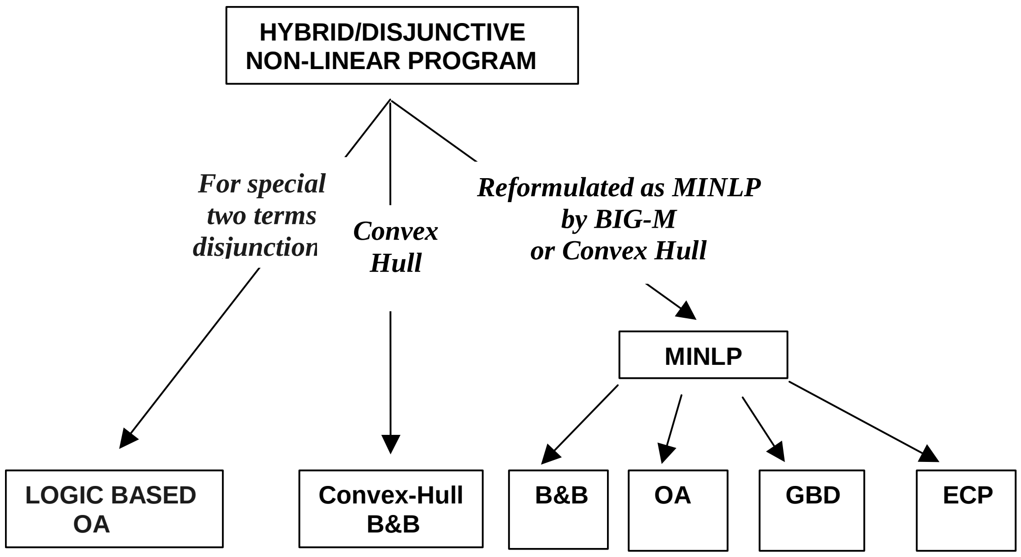

HYBRID/DISJUNCTIVE NON-LINEAR PROGRAM
For special two terms disjunction
Convex Hull
Reformulated as MINLP by BIG-M or Convex Hull
MINLP
LOGIC BASED OA
Convex-Hull B&B
B&B
OA
GBD
ECP

Fig. 2: solution algorithms for nonlinear models

From nonlinear hybrid/disjunctive models you have three paths to arrive to the solution the first one is for models that have special two terms disjunctions that have the following formulation:

$$\begin{bmatrix} Y_i \\ h_i(x) \le 0 \\ c_i = \gamma_i \end{bmatrix} \vee \begin{bmatrix} \neg Y_i \\ B^i x = 0 \\ c_i = 0 \end{bmatrix} \quad i \in D$$

This special disjunction involves two terms handled by one variable, the first term apply when the Yi is true; while the second when it is false. In the second term a subset of variables must set to zero.

The **LOGIC BASED OA** algorithm (Turkay and Grossmann, 1996a) was mainly generated to solve the synthesis of chemical processes. It was applied to examples other than these types of problems. **This is the only solution method working in this release of LOGMIP**. The solver name is LMLBOA.

The other two paths are not implemented yet.

To select the LOGIC BASED OA algorithm you must write in the GAMS input file the following sentence:

```
OPTION MINLP=LMLBOA;
```

Observe that since this corresponds to a nonlinear model you must select the option for MINLP models. See the nonlinear example provided in section 2 of this manual.

8.2.1. INITIALIZATIONS

For some nonlinear models in order to solve a particular problem the user should provide initializations. The meaning of the initializations is to fix the disjunctions terms that are true or false. The algorithm runs as many NLP subproblems as initializations is provided. By running these subproblems it is possible to generate the first MASTER MIP. More details about this initialization step can be found in Turkay and Grossmann (1996a).

There is a special LogMIP language construction for this purpose. The initialization entries must be provided in LogMIP section. The keyword **INIT** is used. After that you must provide the list of disjunctions terms that are TRUE or FALSE by adding those words, and the list of variables that handle those terms, separated by commas and ended by a semicolon. Each **INIT** entry corresponds to an initialization set (a new NLP subproblem). In the Nonlinear example showed in section 2, this initialization is provided:

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

The first initialization set specifies that the first term of disjunction 1, 3, 4, 7 and 8 must be true, the second 1, 3, 5, y 8 and the third 2, 3, 4, 6 y 8. The algorithm executes three Non Linear Program problems by considering in the model the constraints of the disjunction terms that are TRUE. The resulting problem must be feasible, so the correct initialization sets must be provided.

The initialization entries must be written after the disjunction definitions.

The same specification can be done by the following sentences:

```
INIT FALSE Y('2'),   Y('5'),   Y('6');
INIT FALSE Y('2'),   Y('4'),   Y('6'),   Y('7');
INIT FALSE Y('1'),   Y('5'),   Y('7');
```

Instead of initializing by the TRUE terms, we can initialize by those that are FALSE. We can have also a combination of both initializations previously presented:

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

The order of the sentences is not important.
If you want to specify all disjunctions terms you can use the word ALL. For example:
INIT TRUE ALL;
or
INIT FALSE ALL;

8.2.2 TERMINATION OPTIONS

Since the OA Logic-Based algorithm is similar to the OA MINLP algorithm based on dividing the original problem into two subproblems: the NLP subproblem and the master MILP subproblem, and according to this, two termination options for the algorithm exist for the LogMIP non-linear solver. These options are similar to those implemented in DICOPT++ (MINLP GAMS solver).

1. STOP on CROSSOVER
2. STOP on NLP worsening

Option 1 is appropriated for convex models. The meaning of this option is that the objective function obtained in the Master MILP subproblem is lower/upper the solution obtained in the NLP subproblem depending if the optimization direction is maximize/minimize the objective function.
Option 2 is recommended for non-convex model, and the algorithm stops after the objective function obtained in the solution of consecutive NLP subproblems starts to deteriorate.
The default is option 1.
These options are implemented in the same way than DICOPT++, you must specify at the option file (LMLBOA.opt) STOP 1 or STOP 2 depending on the choice selected.

9. 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 start with the constraint and objective function 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. LogMIP compiler can not deal with variables and constraint not declared over a domain and then defined with a domain.

- Write your entire model in a single file, do not use the include directive 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 ($).

- A similar advice is necessary for variables handling disjunction terms. Do not forget to include the dummy equation for them.

10. 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**.