MODELING ISSUES AND IMPLEMENTATION OF LANGUAGE FOR DISJUNCTIVE PROGRAMMING

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

[(+)]INGAR – Instituto de Desarrollo y Diseño

UTN – Facultad Regional Santa Fe

e-mail: aldovec@alpha.arcride.edu.ar

[(*)]Department of Chemical Engineering, Carnegie Mellon University,

Pittsburgh, PA 15213 - USA

e-mail: grossmann@cmu.edu

Keywords: Generalized Disjunctive Programming (GDP), Mixed-Integer Nonlinear Programming, Constraint Logic Programming, Modeling Language

Abstract

This paper describes a number of key modeling issues for the development of tools for solving nonlinear discrete/continuous problems where logic/disjunctive constraints are included in the formulation. A generalized hybrid representation of these problems is presented. A comparison between Constraint Logic Programming (CLP) and Generalized Disjunctive Programming (GDP) is established together with several constraint transformations from CLP to GDP. The components and expressions of a modeling language for setting up disjunctions and logic constraints are proposed. The language allows the specification of problems with complex logic formulations. A parser is developed for the analysis and translation of the logic sentences into files ready to be used by the solvers. An overview of the solution algorithms is also given together with several aspects about the implementation as a superset of GAMS mathematical programming language. Examples illustrating the capabilities of the proposed system are described.

1. Introduction

In recent years there have been several efforts for incorporating logic in mathematical and optimization programming. The logic is introduced at the level of problem formulation, and at the level of solution techniques. Disjunctive Programming (Raman and Grossmann, 1994; Turkay and Grossmann 1996, Bjorkqvist and Westerlund, 1999) and Constraint Logic Programming (Hajian et al. 1995; Darby-Dowman et al, 1997) are examples of these efforts. A disjunctive program can be regarded as a mixed integer program involving disjunctive constraints. Constraint Logic Programming combines a powerful language to express combinatorial problems with constraint propagation techniques within an implicit enumeration search. Compared to Mixed-Integer Nonlinear Program (MINLP), both approaches offer significantly improved techniques for modeling, and in many cases more effective solutions in the areas of design, synthesis, planning and scheduling of process engineering problems. The introduction of disjunctions and logic into the formulation is in many cases a more direct way of stating the problem. Although, new models, algorithms and solvers have been proposed, challenges remain at the level of modeling, language expressiveness, solution techniques and general tools for solving logic-based problems. In particular, when adaptations of languages conceived for mathematical programming or complex declarations are used for expressing a model with disjunctions or logic constraints, this can lead to ambiguous problem statements or models that are difficult to read or understand.

It is the purpose of this paper to propose a number of ideas and concepts for disjunctive programming regarding the modeling, language syntax and its implementation in a computer code. The motivation is to develop general tools that can address nonlinear/discrete optimization problems with logic in the formulation. For the problem representation we start from a general hybrid formulation including disjunctions, Boolean and 0-1 variables for the discrete choices. Logic propositions are included in the problem formulation for stating relations between the Boolean variables. These propositions involve the logic operators “and”, “or”, “not” and “implication”. The relationship between the disjunctive representation and models from constraint logic programming is then presented with the transformation of logic constraints and embedded disjunctions in the form of general disjunctions. Next, the basic elements of a language to express a general hybrid disjunctive model are proposed, as well as the connection between the models and the algorithms available to solve them. Finally, results obtained in the solution of several examples are presented.

2. Generalized Hybrid representation

The model presented below corresponds to a generalized hybrid formulation for a continuous discrete nonlinear program problem where the discrete choices are represented by disjunctions, Boolean variables and binary variables (Vecchietti and Grossmann, 1999):

$$
\begin{aligned}
&\min \quad Z = \Sigma_k c_k + f(x) + d^T y \\
&st \\
&\qquad g(x) \le 0 \\
&\qquad r(x) + Dy \le 0 \\
&\qquad Ay \ge a \qquad\qquad\qquad\qquad \textbf{(PH)} \\
&\qquad \bigvee_{i \in D_k} \begin{bmatrix} Y_{ik} \\ h_{ik}(x) \le 0 \\ c_k = \gamma_{ik} \end{bmatrix} \quad k \in SD \\
&\qquad \Omega(Y) = True \\
&x \in R^n \ , y \in \{0,1\}^q \ , Y_{ik} \in \{True, False\}^m, c_k \ge 0
\end{aligned}
$$

In problem (PH) x and c_k are continuous variables, y are binary variables (0-1), Y_{ik} are Boolean variables to establish whether a given term in a disjunction is true $[h_{ik}(x) \le 0]$, $\Omega(Y)$ are logical relations between Boolean variables, $f(x)$ represents a linear/nonlinear objective function, $g(x)$ are linear/nonlinear inequalities that hold independent of the discrete choices, $r(x)+Dy \le 0$ corresponds to general mixed integer algebraic equations, $Ay \ge a$ is a set of integer inequalities, $d^T y$ are fixed cost terms.

When constraints involving binary variables are not present in (PH), this problem reduces to the Generalized Disjunctive Program formulation by Raman and Grossmann (1994):

$$
\begin{aligned}
&\min \quad Z = \Sigma_k c_k + f(x) \\
&st \\
&\qquad g(x) \le 0 \\
&\underset{i \in D_k}{\vee} \begin{bmatrix} Y_{ik} \\ h_{ik}(x) \le 0 \\ c_k = \gamma_{ik} \end{bmatrix} \quad k \in SD \qquad\qquad \textbf{(GDP)} \\
&\qquad \Omega(Y) = True \\
&x \in R^n \ , \ Y_{ik} \in \{True, False\}^m, \ c_k \ge 0
\end{aligned}
$$

From (PH) it is also possible to obtain a Mixed Integer Non-Linear Programming problem (MINLP) if disjunctions and Boolean variables are not included in the problem formulation:

$$
\begin{aligned}
&\min \quad Z = f(x) + d^T y \\
&st \\
&\qquad g(x) \le 0 \\
&\qquad r(x) + Dy \le 0 \qquad\qquad \textbf{(PA)} \\
&\qquad Ay \ge a
\end{aligned}
$$

In this way problem (PH) provides the flexibility of modeling a problem as a GDP, MINLP or a hybrid model.

The propositional logic in (PH) is expressed through the conjunction of q different propositions,

$$\Lambda = \{ L_1 \wedge L_2 \wedge ... \wedge L_q \}$$

where L_i is a logical proposition expressed in terms of $\wedge$ (and), $\vee$ (or), $\neg$(negation, not), $\Rightarrow$ (implication) and $\Leftrightarrow$ (equivalence) operators.

The set of clauses Λ can be transformed into the Conjunctive Normal Form (CNF) (Clocksin and Mellish, 1981) which is expressed as follows:

$$\Omega = \left[\underset{i \in P1}{\vee} Y_i \ \underline{\vee}_{i \in P1} Y_i \right] \wedge \left[\underset{i \in P2}{\vee} Y_i \ \underline{\vee}_{i \in P2} Y_i \right] \wedge ..... \wedge \left[\underset{i \in Ps}{\vee} Y_i \ \underline{\vee}_{i \in Ps} Y_i \right]$$

where P_i and $\bar{P}_i$ are subsets of the Boolean variables that correspond to a subset of 0-1 variables and *s* is the number of disjunctive clauses. The CNF form implies that every clause in Ω must be satisfied. Although in previous work the propositional logic was mostly implemented in problems of process synthesis to reflect the structural relationship between the units (Raman and Grossmann, 1993), it can be applied to any type of problem where it is needed. The CNF set Ω can be transformed automatically into an equivalent set of integer inequalities (Tourn, 1995). It should be noted that for the special case where we have the following multiple choice constraint (at most one item),

$$\sum_i Y_i \leq 1 \qquad (1)$$

it is cumbersome to use propositional logic to represent it. However, using a slack Boolean variable *z* with a "exclusive or" for all the Boolean variables Y_i, yields a simple transformation:

$$Y_1 \veebar Y_2 \veebar \ldots \veebar Y_s \veebar z \Rightarrow \sum_i Y_i + z = 1 \qquad (2)$$

In this way if z is true (z=1) all Y_i are false (Y_i= 0), and if z is false (z=0) only one Y_i can be true (Y_i= 1).

3. Constraint Logic Programming (CLP) and Generalized Disjunctive Programming (GDP)

In this section we explore the relationship between Constraint Logic Programming and Generalized Disjunctive Programming. In the past ten years, Constraint Logic Programming (CLP) (Henteryck, 1989; Tsang, 1999; Darby Dowman et al., 1997) has become an important tool for solving scheduling, resource allocation and planning problems, which are difficult combinatorial optimization problems. In CLP the problem is modeled in more expressive logic syntax. For instance, conditional constraints, constraints on all-different and meta-constraints are employed in the problem formulation. CLP solves the model by creating a search tree based on enumeration, and during the search it reduces the domain of the variables by propagating the constraints. On the other hand, Generalized Disjunctive Programming (GDP) has been applied to design, scheduling and synthesis problems, which can leads to improved models compared to MINLP (Raman and Grossmann, 1994; Lee and Grossmann, 1999). Several algorithms have been proposed for linear and nonlinear GDP problems. A brief review of these methods is presented later in the paper.

Below, the transformations of some logic constraints into the GDP form are proposed. The objective is to show that some common constraints of CLP can be reformulated as a GDP. Finally, in this section we present an example of the transformations.

Consider the three following cases of conditional constraints that arise in CLP (Hanteryck, 1989):

case a) $$g(x) \leq 0 \Rightarrow f(x) \leq 0 \qquad (3)$$

where g(x), f(x) are scalar functions. This implication can be transformed to:

$$\neg\, g(x) \leq 0 \ \vee \ f(x) \leq 0 \qquad (4)$$

where $\vee$ is the "or" operator and $\neg$ is the negation of the constraint. The latter implies the non-satisfaction of the constraint, with which (4) can be written as:

$$g(x) \geq \varepsilon \ \vee \ f(x) \leq 0 \qquad (5)$$

For implementation, typical values of ε can be chosen between 0.0001 and 0.001.

Assigning Boolean variables to (5) we obtain a two-term disjunction covered by the model GDP,

$$\begin{bmatrix} Y_1 \\ g(x) \geq \varepsilon \end{bmatrix} \vee \begin{bmatrix} Y_2 \\ f(x) \leq 0 \end{bmatrix} \qquad (6)$$

It should be noted that for the case when g(x) and f(x) are vector functions, the implication in (3) can be written as:

$$\underset{i \in I}{\wedge}\ g_i(x) \leq 0 \Rightarrow \underset{j \in J}{\wedge}\ f_j(x) \leq 0 \qquad (7)$$

Eliminating the implication,

$$\neg\left(\underset{i \in I}{\wedge}\ g_i(x) \leq 0\right) \vee \left(\underset{j \in J}{\wedge}\ f_j(x) \leq 0\right) \qquad (8)$$

moving the negation inward yields,

$$\underset{i \in I}{\vee}\left(\neg g_i(x) \leq 0\right) \vee \left(\underset{j \in J}{\wedge}\ f_j(x) \leq 0\right) \qquad (9)$$

it can be transformed to:

$$\underset{j \in I}{\wedge}\left(\underset{i \in J}{\vee} \neg g_i(x) \leq 0 \ \vee f_j(x) \leq 0\right) \qquad (10)$$

which in turn can be written in GDP form as:

$$\bigvee_{i\in I}\begin{bmatrix} Y_i \\ g_i(x)\geq \varepsilon \end{bmatrix} \vee \begin{bmatrix} Y_j \\ f_j(x)\leq 0 \end{bmatrix} j\in J \qquad (11)$$

case b) $$Y_i \Rightarrow g_i(x) \leq 0 \qquad (12)$$

where Yi is a Boolean variable and $g_i(x)$ a vector function. To cover the negation of this implication we need to expand it to:

$$Y_i \Rightarrow g_i(x) \leq 0 \ \vee \ \neg Y_i \Rightarrow x\in D \qquad (13)$$

allowing x to lie in the relaxed domain D when Y_i is not true. The expression in (13) can be readily be written as a two term disjunction of the model (GDP),

$$\begin{bmatrix} Y_i \\ g(x)\leq 0 \end{bmatrix} \vee \begin{bmatrix} \neg Y_i \\ x\in D \end{bmatrix} \qquad (14)$$

Similarly, when we have the opposite implication:

$$g_i(x) \leq 0 \Rightarrow Y_i \qquad (15)$$

it can be transformed to,

$$\neg\, g_i(x) \leq 0 \ \vee Y_{i,} \qquad (16)$$

which yields the GDP form,

$$\begin{bmatrix} \neg Y_i \\ g(x)\geq \varepsilon \end{bmatrix} \vee \begin{bmatrix} Y_i \\ x\in D \end{bmatrix} \qquad (17)$$

case c)

$$\Omega_i(Y) \Rightarrow h_i(x) \leq 0 \ \vee \ \neg\Omega_i(Y) \Rightarrow g_i(x) \leq 0 \qquad (18)$$

where $\Omega_i(Y)$ is a set of logic propositions which has a value of true if all are satisfied. Introducing a Boolean variable Z_i for $\Omega_i(Y)$, (18) can be expressed as:

$$Zi \Rightarrow h_i(x) \leq 0 \ \vee \ \neg\, Z_i \Rightarrow g_i(x) \leq 0 \qquad (19)$$

The final representation of (19) as a disjunction in GDP form is given by:

$$\begin{bmatrix} Z_i \\ h_i(x) \le 0 \end{bmatrix} \vee \begin{bmatrix} \neg Z_i \\ g_i(x) \le 0 \end{bmatrix} \qquad (20)$$

$$\Omega_i(Y) \Leftrightarrow Z_i$$

case d)

An additional case that is of interest is embedded disjunctions that do not directly fit the GDP form. These arise for instance in multiperiod design problems (Van der Hever and Grossmann, 1999). An example is the following disjunction:

$$\begin{bmatrix} Y_1 \\ \begin{bmatrix} Z_1 \\ h_1(x) \le 0 \end{bmatrix} \vee \begin{bmatrix} Z_2 \\ h_2(x) \le 0 \end{bmatrix} \end{bmatrix} \vee \begin{bmatrix} Y_2 \\ \\ g(x) \le 0 \end{bmatrix} \qquad (21)$$

which implies that if Y_1=True, this is an additional disjunction for Z_1 and Z_2.

Although disjunctions like in (21) are not covered in model GDP, they can be transformed to non-embedded disjunctions as follows:

$$\begin{bmatrix} Y_1 \\ Z_1 \vee Z_2 \end{bmatrix} \vee \begin{bmatrix} Y_2 \\ g(x) \le 0 \end{bmatrix} \qquad (22)$$

$$\begin{bmatrix} Z_1 \\ h_1(x) \le 0 \end{bmatrix} \vee \begin{bmatrix} Z_2 \\ h_2(x) \le 0 \end{bmatrix} \qquad (23)$$

Simplifying the first disjunction in (22) yields,

$$\begin{array}{c} Y_1 \vee Y_2 \\ Z_1 \vee Z_2 \Leftrightarrow Y_1 \\ Y_2 \Rightarrow g(x) \le 0 \end{array} \qquad (24)$$

transforming the implication as in case b) produces:

$$\begin{bmatrix} Y_2 \\ g(x) \le 0 \end{bmatrix} \vee \begin{bmatrix} \neg Y_2 \\ x \in D \end{bmatrix} \qquad (25)$$

From (22)-(25) the final GDP form of an embedded disjunction is given by:

$$\begin{bmatrix} Z_1 \\ h_1(x) \le 0 \end{bmatrix} \vee \begin{bmatrix} Z_2 \\ h_2(x) \le 0 \end{bmatrix} \qquad (26)$$

$$\begin{bmatrix} Y_2 \\ g(x) \le 0 \end{bmatrix} \vee \begin{bmatrix} \neg Y_2 \\ x \in D \end{bmatrix} \qquad (27)$$

$$\begin{gathered} Y_1 \vee Y_2 \\ Z_1 \vee Z_2 \Leftrightarrow Y_1 \end{gathered} \qquad (28)$$

Thus, what the transformations for CLP constraints and the embedded disjunctions have shown is that a rather broad set of logic constraints can be converted into the disjunctions as in (GDP), or for that matter as in (PH).

Example of a CLP model converted to GDP form

We show next how a complex CLP model can be converted into a GDP. The following example corresponds to a CLP model formulated in the ILOG solver:

Continuous variables : $x \ge -5, \quad y \ge 0$
Integer variables : $k=0,1,2,3,4$
Constraints :

$$x^3 + 10\,x = y^x + 2^k$$

$$kx + 7.7\,y = 2.4$$

$$(k-1)^{y+1} \le 10 \qquad (29)$$

$$\{ [\log(y + 2x+12) \le k+5] \vee [\, y \ge k^2 \,] \} \Rightarrow \{ x \le 0 \wedge y \le 1 \}$$

$$x \le 0 \Rightarrow k > 3$$

With the transformations described above, the model can be formulated in GDP form as follows:

Continuous variables : $x \ge -5, \quad y \ge 0$
Integer variables : $k=0,1,2,3,4$
Constraints :

$$x^3 + 10\,x = y^x + 2^k$$

$$kx + 7.7\,y = 2.4$$

$$(k-1)^{y+1} \le 10$$

$$\begin{bmatrix} Y_1 \\ log(y+2x+12)-(k+5)\geq \varepsilon \\ y-k^2 \leq \varepsilon \end{bmatrix} \vee \begin{bmatrix} Y_2 \\ x \leq 0 \\ y \leq 1 \end{bmatrix} \qquad (30)$$

$$\begin{bmatrix} W_1 \\ x \geq \varepsilon \end{bmatrix} \vee \begin{bmatrix} W_2 \\ k \geq 3 \end{bmatrix}$$

4. Language

LOGMIP is the first code that was implemented for solving nonlinear/discrete problems formulated in the form of the hybrid model (PH). In that solver we adopted some constructions of the mathematical programming language of GAMS for expressing disjunctions (Vecchietti and Grossmann, 1999). Writing a disjunction in GAMS language is not natural and not concise since we are using a language created to define a model in algebraic form. Also, logic propositions have to be provided to the model as integer constraints to pose a problem in the form of problem (PH). A Prolog (Tourn, 1995) program developed to transform the logical expression into inequalities is executed separately of LOGMIP. The program output is included in the LOGMIP input file. In this work we describe a new language in LOGMIP for expressing models formulated in the form of PH or GDP. The idea is to provide capabilities for describing in a natural form disjunctions, logic expressions and constraints within a mathematical programming language.

In the past, several constructions have been proposed for expressing conditional models like the one proposed in the PH and GDP formulations. Pantelides (1988) proposed an input language for dynamic simulation in SpeedUp based on IF statements and the logical operators AND, OR, NOT. The language is used for expressing discontinuities in the model and logical conditions of any complexity. Recently, Rico-Ramirez (1998) proposed several constructions based on WHEN...CASE, SELECT...CASE, SWITCH CASE, CONDITIONAL statements for the description of conditional models in the equation oriented software ASCEND. There are also some other solvers for constraint logic programming such as ECLiPSe and ILOG where conditional constraints, logic constraints and special constructs like the all-different can be expressed in high-level languages (Bockmayr and Kasper, 1998).

The symbols, reserved words, components and language syntax to express models in the form of generalized hybrid/disjunctive programming problem (PH) are described bellow.

The language consists of: Boolean variables, continuous and binary variables, mathematical, relational and logical operators, selection statements of the type "IF...THEN...ELSE...ENDIF". The descriptions of these components are as follows:

Operands, operators and statements:

Binary and continuous variables.

Boolean variables, which can take values true or false.

Mathematical operators: +, -, *, /.

Relational operators: = (equal), <= (less than or equal to), >= (greater than or equal to).

Logical operators: $\wedge$ (and), $\vee$ (or), ~(not,!), ->(implication), <->(equivalence).

Selection statements of the form IF...THEN...ELSE...ENDIF.

The expressions that can be formulated with this approach are the following:

Expressions

- *Selection statements* of the form IF...THEN...ELSE...ENDIF are used to formulate a disjunction. It selects the set of constraints to be applied after evaluating a logical expression (or a single Boolean variable) to true or false. Example:

  $IF\ (\ Y_i\ )\ \ THEN\ \ \ h_i(x) \leq 0$

  $ELSE\ \ \ \ g_i(x) \leq 0$

  $ENDIF$

- *Logic propositions*: used to express (by the logical operators previously described) relationships between the Boolean variables. Example:

  $$y1\ \wedge \neg y2 \Rightarrow y3 \vee y4$$

- *Logic expressions*: expressions relating variables (continuous, binary or Boolean) and operators (mathematical and/or logic). Examples: $[(x+3) \Rightarrow (y+4)]$. Although these statements are not contained in problem (PH), from the previous section these expressions can be converted into disjunctions (by the user or by LOGMIP) such that can be processed by the algorithms.

Hence, a rather large class of logic expressions is allowed in our language. Allowing the formulation of complex logical sentences allows the problem to be set up in a more flexible way for the user. For the solution, however, we transform these logic expressions into propositional logic and disjunctions as were discussed in the previous section.

We have chosen the selection statement IF...THEN...ELSE...ENDIF because it states naturally and directly all types of disjunctions. Two term disjunctions are formulated with a single IF sentence. Through nesting sentences we can define disjunction with several terms and/or embedded disjunctions. Below we show how to write the disjunctions in the proposed language.

Two terms disjunctions

$$\left[\begin{array}{c} True \\ \\ \end{array}\right] \vee \left[\begin{array}{c} False \\ \\ \end{array}\right]$$

IF *(logic expression)* **THEN**
    Constraints to be applied when logic expression is TRUE
ELSE
    Constraints to be applied when logic expression is FALSE
ENDIF

Several Terms Disjunction

$$\left[\begin{array}{c} 1 \\ \\ \end{array}\right] \vee \left[\begin{array}{c} 2 \\ \\ \end{array}\right] \vee \left[\begin{array}{c} 3 \\ \\ \end{array}\right]$$

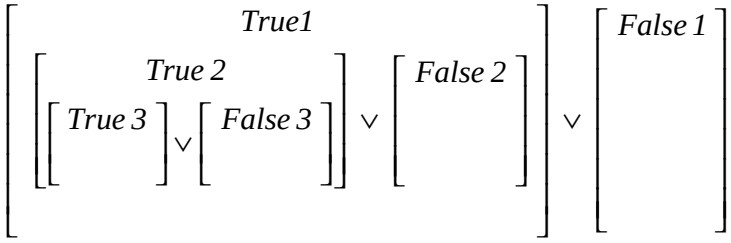
IF *(logic expression$_1$)* **THEN**
    Constraints to be applied when logic expression$_1$ is TRUE
ELSE IF *(logic expression$_2$)* **THEN**
    Constraints to be applied when logic expression$_2$ is TRUE
ELSE IF *(logic expression$_3$)* **THEN**
    Constraints to be applied when logic expression$_3$ is TRUE
ENDIF

Note that the above also applies to disjunctions with two terms if each of them is activated by a different logic expression.

Embedded Disjunction

Case a)

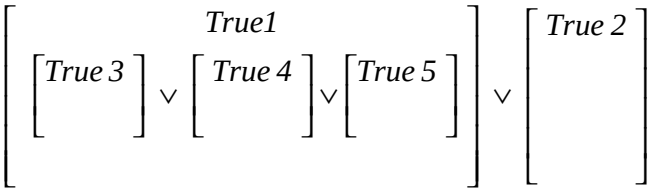
The first case involves disjunctions terms that can be true or false:



IF *(logic expression$_1$)* **THEN**
    Constraints to be applied when logic expression$_1$ is TRUE
    IF *(logic expression$_2$)* **THEN**
        Constraints to be applied when logic expression$_2$ is TRUE
        IF *(logic expression$_3$)* **THEN**
            Constraints to be applied when logic expression$_3$ is TRUE
        ELSE
            Constraints to be applied when logic expression$_3$ is FALSE
        ENDIF
    ELSE
        Constraints to be applied when logic expression$_2$ is FALSE
    ENDIF
ELSE
    Constraints to be applied when logic expression$_1$ is FALSE
ENDIF

Case b)

For this case the disjunction terms are handled by different Boolean variables:



IF *(logic expression$_1$)* **THEN**
    Constraints to be applied when logic expression$_1$ is TRUE
    IF *(logic expression$_3$)* **THEN**
        Constraints to be applied when logic expression$_3$ is TRUE
    ELSE IF *(logic expression$_4$)* **THEN**
        Constraints to be applied when logic expression$_4$ is TRUE
    ELSE IF *(logic expression$_5$)* **THEN**
        Constraints to be applied when logic expression$_5$ is TRUE
    ENDIF
ELSE IF *(logic expression$_2$)* **THEN**
    Constraints to be applied when logic expression$_2$ is TRUE
ENDIF

In figure 1 we present the input file of the problem (30) written with the new language.

5. Overview of the solution algorithms

The tree diagram of Figure 2 shows the different methods that can be used to solve the Hybrid (PH), Disjunctive (GDP) and MINLP (PA) models. The diagram also shows the conditions and transformations needed for these problems to be solved by the algorithms, which are in the lowest level of the tree.

The Logic Based OA (Outer Approximation) and the Logic Based GBD (Generalized Benders Decomposition) methods can be applied for disjunctive problems (GDP) involving two terms disjunctions like the one proposed by Turkay and Grossmann (1996). Extensions of these algorithms can be applied for the case of the hybrid form (Vecchietti and Grossmann, 1999). In the same way the GDP form through the generation of the convex hull of the nonlinear disjunctions can be solved by the Branch and Bound (B&B/CRP) algorithm proposed by Lee and Grossmann (1999). This method can be extended to the hybrid formulation in order to handle both the disjunctions and the binary variables. The Hybrid and GDP representations can be reformulated as MINLP problems replacing the disjunctions by Big-M constraints or generating the Convex Hull (Lee and Grossmann, 1999). Once the transformation has been made, any method for MINLP can be applied. The best known algorithms for solving a MINLP problem are Branch and Bound (B&B) (Gupta and Ravindran (1985), Ryo and Sahinidis (1995), Stubbs and Mehrotra (1996)), Outer-Approximation (OA) (Duran and Grossmann, 1986) and its extensions Outer-Approximation/Equality-Relaxation/Augmented-Penalty (OA/ER/AP) (Viswanathan and

Grossmann, 1990) , Generalized Benders Decomposition (GBD) (Geoffrion, 1972) and Extended Cutting Plane (ECP) (Westerlund and Petterson, 1995).

6. Implementation

In the new version of LOGMIP, nonlinear discrete/continuous programs can be formulated in any of the three representations presented before: PH, GDP and MINLP. Aside from the language, unique capabilities in LOGMIP are the selection between several formulation and solution methods.

For the problem formulation, the language previously presented is extended in GAMS for stating logic constraints, logic propositions and disjunctions. The new language is a superset of the GAMS language. In this way we can get the capabilities and advantages of both the mathematical and the logic expressions for setting up a problem. A parser has been developed for the recognition of logic constructions of the language. In that way, the input file for the new version of LOGMIP includes the mathematical programming language of GAMS and all the logic sentences. A precompiler step is first executed, and the input file is transformed into a file ready to be compiled by GAMS plus another file containing all the information of the logic needed for the solution algorithms. This file is read when the algorithm is executed. Figure 3 shows the flowchart of the new version of LOGMIP.

At the level of the solution methods, the code of DICOPT++ has been used as a base for the implementation of the ECP and GBD MINLP algorithms. For the case of the algorithms that consider directly the disjunctions in the formulation, we have implemented the Logic-Based OA algorithm and the convex hull transformation. We intend to add the disjunctive branch and bound method in the future. As for the solutions methods the defaults are as follows:

a) If the problem is posed as an MINLP the OA method is implemented and DICOPT++ is applied.
b) If the problem is posed as a GDP, the convex hull transformation is used for the disjunctions, and the logic propositions are transformed in equation form. The resulting GDP is solved as an MINLP.
c) If the problem is posed as the hybrid model (PH), then the same transformations as in (b) are used.

It should be noted that in above cases the advantage is that the user no need to supply initial guesses since the default initialization in GAMS is invoked.

For the case when the user wants to apply the Logic-Based OA, which would be for process networks expressed in the form of (GDP) or (PH), the user must supply an initialization through set covering with additional initial values for the variables.

7. Examples

Three examples have been solved for illustration purposes. The first problem corresponds to the transformed CLP example in (30). That PH formulation was converted to an MINLP by the Convex Hull transformation. The OA method was used and the solution was obtained at the first step of the algorithm when the relaxed problem was solved. The solution of this problem is x=-1.285, y= 0.979 and k=4.

The other problems have been solved with LOGMIP with several configurations and algorithms. The data for these examples can be found in Vecchietti and Grossmann (1999).

The second example corresponds to a process network superstructure (see Figure 4) where the optimal configuration has to be found. The input file of the GDP formulation for this problem with the new language can be found in Appendix A. The results obtained in the solution of this problem are presented in Table 1.

The third example is a multiproduct batch plant design where the objective is to determine the unit design, number of units in parallel and the storage tank location and design in order to minimize the investment cost. This example has been also solved considering different algorithms and formulations. The results obtained are shown in Table 2.

From the results presented in the previous tables it can be seen that no formulation outperforms the other. Different number of iterations, and hence computational time, are needed to reach the solution and also the optimal values of the starting points differ. Therefore, it is important to have a general and flexible tool where not only the formulation but also the algorithm can be selected for solving a discrete/continuous nonlinear problem.

8. Concluding remarks

This paper has described a generalized modeling framework and solution techniques for nonlinear discrete/continuous problems. The proposed approach allows the representation of the

same problem with different formulations. These can be expressed in terms of equations only, and/or disjunctions and logic constraints. We have presented the transformations between several logic constraints, which are frequent in Constraint Logic Programming (CLP), into the Generalized Disjunctive Programming (GDP) form. The significance of these transformations is that problems with logic constraints can be translated into the disjunctive form of problems PH or GDP. We have also proposed a language for the expression of logic constraints and disjunctions. The selection sentences IF...THEN...ELSE...ENDIF have been chosen for expressing disjunctions. The choice was based on the simplicity and expressiveness for posing disjunctions of different levels of complexity. The proposed language also includes the operators, statements and symbols for posing logic expressions and propositions. The language is a superset of the GAMS mathematical programming language. This combination allows the specification of complex mathematical/logic program optimization problems. A parser for checking the syntax, analysis and transformation of the logic sentences into readable files for the solvers has been developed. A brief overview of the algorithms for the solution of these problems was given, and several examples were presented. We intend to report in the future more extensive results once the development of the new version of LOGMIP has been completed.

Acknowledgments. The authors are grateful for the financial support to NSF International Cooperative Research (grant INT-9724823) and to CONICET for the Argentina counterpart.

REFERENCES

Bjorkqvist, J. and Westerlund T.. Automated Reformulation of Disjunctive Constraints in MINLP Optimization. Comp. and Chem. Eng., **23**,Supp., S11-S14, 1999.

Bockmayr, A. and Kasper T.. Branch and Infer: A Unifying Framework for Integer and Finite Domain Constraint Programming. INFORMS Journal of Computing, **10** (3), 1998.

Darby-Dowman K., Little J., Mitra G. and Zaffalon M.. Constraint Logic Programming and Integer Programming Approaches and Their Collaboration in Solving an Assignment Scheduling Problem. Constraints, **1**, 245-264, 1997.

Duran M. and Grossmann I.E.. An Outer-Approximation Algorithm for a Class of Mixed-Integer Nonlinear Programs. Mathematical Programming, **36**, 307-339, 1986.

Geoffrion, A. M.. Generalized Benders Decomposition. Journal of Optimization Theory and Application, **10** (4), 237-260, 1972.

Gupta O.K. and Ravindran V., Branch and Bound Experiments in Convex Nonlinear Integer Programming, Management Science, **31**(12), 1533-1546, 1985.

Henteryck, P.V.. Constraint Satisfaction in Logic Programming. MIT press, Cambridge, MA, 1989.

Hajian, M. T.; El-Sakkout, H.; Wallace, M.; Lever, J. M. and Richards E.B.. Towards a Closer Integration of Finite Domain Propagation and Simplex-Based Algorithms. IC-Parc Publications (UK) (*http://www.icparc.ic.ac.uk/papers.html*), 1995.

Lee, S. and Grossmann, I.E.. Generalized Disjunctive Programming: Nonlinear Convex Hull Relaxation and Algorithms. Submitted for publication, 1999.

Pantelides, C. C.. SpeedUp recent advances in process simulation. Comp. and Chem. Eng., **12** (7), 745-755, 1988.

Raman R. and Grossmann I.E.. Symbolic Integration for Logic in Mixed-Integer Linear Programming Techniques for Process Synthesis. Comp. Chem. Eng., **17** (9), 909-927, 1993.

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

Rico-Ramirez V.. Representation, Analysis and Solution of Conditional Models in an Equation-Based Environment, Ph.D. Thesis, Dept. Chemical Eng., Carnegie Mellon University, Pittsburgh, PA, 1998.

Ryoo, H. S., and N. V. Sahinidis, Global Optimization of Nonconvex NLPs and MINLPs with Applications in Process Design. Comp. Chem. Eng., **19** (5), 551-566, 1995.

Stubbs R. and Mehrotra S.. Branch and Cut Methods for Mixed 0-1 Convex Nonlinear Programming. Paper presented on Fifth SIAM Conference on Optimization, Victoria, British Columbia, Canada, 1996.

Tourn Michel. PROLOG Program for converting propositions into linear inequalities. Chemical Engineering Department , Carnegie Mellon University, Pittsburgh, PA, 1995.

Tsang, E.P.. Formulations at Constraint Satisfaction. Academic Press, 1993.

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

Van der Heever S. and Grossmann I.E.. Disjunctive multiperiod Optimization Methods for Design and Planning at Process Systems. Comp. Chem. Eng., **23**, 1075-1095, 1999.

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

Viswanathan J.V. and Grossmann I.E.. A Combined Penalty Function and Outer-Approximation method for MINLP Optimization. Comp. Chem Eng., **14** (7), 769-778, 1990.

Westerlund and Pettersson. An Extended Cutting Plane Method for Solving Convex MINLP Problems. Comp. Chem. Eng., **19**, S131-36, 1995.

Appendix A. Input file of Processes Superstructure example with new language

```
$TITLE APPLICATION OF THE LOGIC-BASED MINLP ALGORITHM IN EXAMPLE #3
* FOR THIS PROBLEM THE FORMULATION IS DISJUNCTIVE
$OFFSYMXREF
$OFFSYMLIST
*  SELECT OPTIMAL PROCESS FROM WITHIN GIVEN SUPERSTRUCTURE.
*
*  REFERENCE: MARCO DURAN , PH.D. THESIS , 1984.
*             CARNEGIE-MELLON UNIVERSITY , PITTSBURGH , PA.

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

* BALANCES, DESIGN SPECIFICATIONS
* HOLDS INDEPENDENT OF DISCRETE CHOICES
* ------------------------------------------------------------
   MASSBAL1, MASSBAL2, MASSBAL3, MASSBAL4, MASSBAL5, MASSBAL6,
   MASSBAL7, MASSBAL8
   SPECS1, SPECS2, SPECS3, SPECS4

* LOGIC PROPOSITIONS DECLARATIONS
*------------------------------------------------------------
   LOGIC1, LOGIC2, LOGIC3, LOGIC4, LOGIC5, LOGIC6, LOGIC7, LOGIC8,
   LOGIC9, LOGIC10, LOGIC11, LOGIC12, LOGIC13


* DISJUNCTIVE CONSTRAINTS:
* ------------------------------------
* Process 1
          INOUT11, INOUT12, INOUT13, INOUT14
* Process 2
          INOUT21, INOUT22, INOUT23, INOUT24
* Process 3
          INOUT31, INOUT32, INOUT34
* Process 4
          INOUT41, INOUT42, INOUT43, INOUT44, INOUT45
* Process 5
          INOUT51, INOUT52, INOUT53, INOUT54

* Process 6
```

```
          INOUT61, INOUT62, INOUT63, INOUT64
* Process 7
          INOUT71, INOUT72, INOUT73, INOUT74
* Process 8
          INOUT81, INOUT82, INOUT83, INOUT84, INOUT85, INOUT86
*
          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 ;

*
* SET COVERING INITIALIZATION
*
INITIAL BU 3;

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

* EQUATIONS DEFINITIONS
* ---------------------------------------------------------------------
 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')                     ;


* LOGIC PROPOSITIONS
*---------------------------------------------------------------
 LOGIC1..  (y('1') and ~y('2')) or (~y('1') and y('2'))          ;
 LOGIC2..  y('1') -> (y('3') or y('4') or y('5'))                ;
 LOGIC3..  y('2') -> (y('3') or y('4') or y('5'))                ;
 LOGIC4..  y('3') -> (y('1') or y('2'))                          ;
 LOGIC5..  y('3') -> y('8')                                      ;
 LOGIC6..  (y('4') and ~y('5')) or (~y('4') & y('5'))            ;
 LOGIC7..  y('4') -> (y('6') or y('7'))                          ;
 LOGIC8..  y('5') -> (y('1') or y('2'))                          ;
 LOGIC9..  y('5') -> y('8')                                      ;
```

```
 LOGIC10.. y('6') -> y('4')                                          ;
 LOGIC11.. ~y('6') <-> y('7')                                        ;
 LOGIC12.. y('7') -> y('4')                                          ;
 LOGIC13.. y('8') -> (y('3') or y('5')  or ~y('3') or y('5'))        ;

* DISJUNCTION SPECIFICATIONS
* -------------------------------------------------
IF (Y('1'))THEN
 INOUT11..   EXP(X('3')) -1. =E= X('2')            ;
 INOUT14..   CF('1') =E= 5                         ;
ELSE
 INOUT12..   X('2') =E= 0                          ;
 INOUT13..   X('3') =E= 0                          ;
ENDIF

IF (Y('2')) THEN
 INOUT21..   EXP(X('5')/1.2) -1. =E= X('4')        ;
 INOUT24..   CF('2') =E= 8                         ;
ELSE
 INOUT22..   X('4') =E= 0                          ;
 INOUT23..   X('5') =E= 0                          ;
ENDIF

IF(Y('3')) THEN
 INOUT31..   1.5 * X('9') + X('10') =E= X('8')     ;
 INOUT34..   CF('3') =E= 6                         ;
ELSE
 INOUT32..   X('9') =E= 0                          ;
ENDIF

IF(Y('4')) THEN
 INOUT41..   1.25 * (X('12')+X('14')) =E= X('13')  ;
 INOUT45..   CF('4') =E= 10                        ;
ELSE
 INOUT42..   X('12') =E= 0                         ;
 INOUT43..   X('13') =E= 0                         ;
 INOUT44..   X('14') =E= 0                         ;
ENDIF

IF(Y('5'))THEN
 INOUT51..   X('15') =E= 2. * X('16')              ;
 INOUT54..   CF('5') =E= 6                         ;
ELSE
 INOUT52..   X('15') =E= 0                         ;
 INOUT53..   X('16') =E= 0                         ;
ENDIF

IF(Y('6')) THEN
 INOUT61 ..   EXP(X('20')/1.5) -1. =E= X('19')      ;
 INOUT64 ..   CF('6') =E= 7                         ;
ELSE
 INOUT62 ..   X('19') =E= 0                         ;
 INOUT63 ..   X('20') =E= 0                         ;
ENDIF

IF(Y('7'))THEN
 INOUT71 ..   EXP(X('22')) -1. =E= X('21')          ;
```

```
 INOUT74 ..   CF('7') =E= 4                          ;
ELSE
 INOUT72 ..   X('21') =E= 0                          ;
 INOUT73 ..   X('22') =E= 0                          ;
ENDIF

IF(Y('8'))THEN
 INOUT81..   EXP(X('18')) -1. =E= X('10') + X('17');
 INOUT86 ..   CF('8') =E= 5                          ;
ELSE
 INOUT82..   X('10') =E= 0                           ;
 INOUT83..   X('17') =E= 0                           ;
 INOUT84..   X('18') =E= 0                           ;
 INOUT85..   X('25') =E= 0                           ;
ENDIF

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

 OPTION LIMCOL = 0   ;
 OPTION LIMROW = 0   ;

 MODEL LOGIC /ALL/;
 LOGIC.optfile=1;
 SOLVE LOGIC USING LOGMIP MINIMIZING PROF;
```

LIST OF TABLES

LIST OF FIGURES

Table 1. Results of the Processes Superstructure example

Model	MINLP (Big-M)			MINLP (Convex Hull)	PH	GDP
Algorithm	ECP	GBD	OA	OA	Logic Based OA	Logic Based OA
Variables	33	33	33	41	33	33
Discrete variables	8	8	8	8	8	8
Constraints	32	32	32	51	52	52
Objective value at relaxation	-25.1**	15.08	15.08	62.6	73.1*	73.1*
Iterations	6 MIP	1 NLP 17 major	1 NLP 4 major	1 NLP 2 major	2 NLP 1 major	3 NLP 1 major
Optimum value	68	68	68	68	68	68

(**) Initial MIP
(*) Initial NLP

Table 2. Results of the Multiproduct Batch Plant example

Model	MINLP (Big-M)		MINLP (Convex Hull)	PH
Algorithm	ECP	OA	OA	Logic Based OA
Variables	114	113	225	113
Discrete variables	53	53	53	53
Constraints	187	187	277	187
Objective value at relaxation	-134230**	219335	224165	305061*
Iterations	28 MIP	7 major	16 major	3 major
Objective value	261883	261883	261883	261883

(**) Initial MIP
(*) Initial NLP

Figure 1. Input file of problem presented in (30) with the new language

```
$TITLE CLP Problem from ILOG
$OFFSYMXREF
$OFFSYMLIST

scalar EP / 0.000001/;

* Variable declarations
*
variables X, obje;
positive variables  Y;
integer variable K;
binary variables Y1, Y2, Z, W1, W2;

* Equation declarations
*
equations con1, con2, con3, cost, first11, first21, first22, first31
          secnd11, secnd21, secnd22, secnd31, prop1;

* Equations that hold independent of discrete choices
*
con1.. POWER(x,3) + 10.*X - EXP(X*LOG(Y)) + EXP(K*LOG(2)) =E= 0.0     ;
con2.. K*X + 7.7*Y - 2.4 =E=0.0     ;
con3.. EXP((Y+1)*LOG(K-1)) =L= 10   ;

*  DISJUNCTIONS
*
IF(Y1) THEN
      first11.. LOG( Y + 2*X + 12) – (k-5) =G= EP ;
      first12.. Y - K**2 =L= EP;
ELSE IF(Y2) THEN
      secnd21.. X =L= 0;
      secnd22.. Y =L= 1;
ENDIF

IF(W1) THEN
      first31.. X =G= EP;
ELSE IF(W2) THEN
      secnd31.. K =G= 3;
ENDIF


cost.. obje =E= x;

MODEL Logic /all/;

SOLVE Logic USING DISJUNCTIVE minimizing obje;
```

Figure 2. Overview of the solutions algorithms

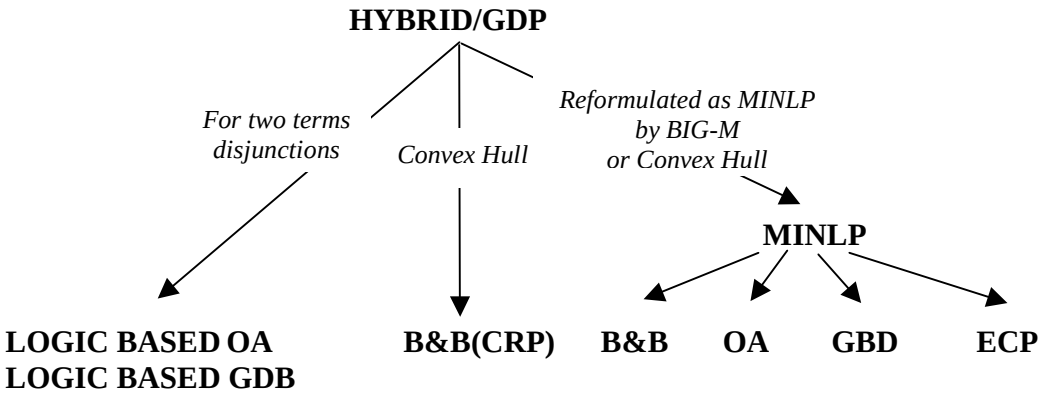


Figure 3. Flowchart of LOGMIP

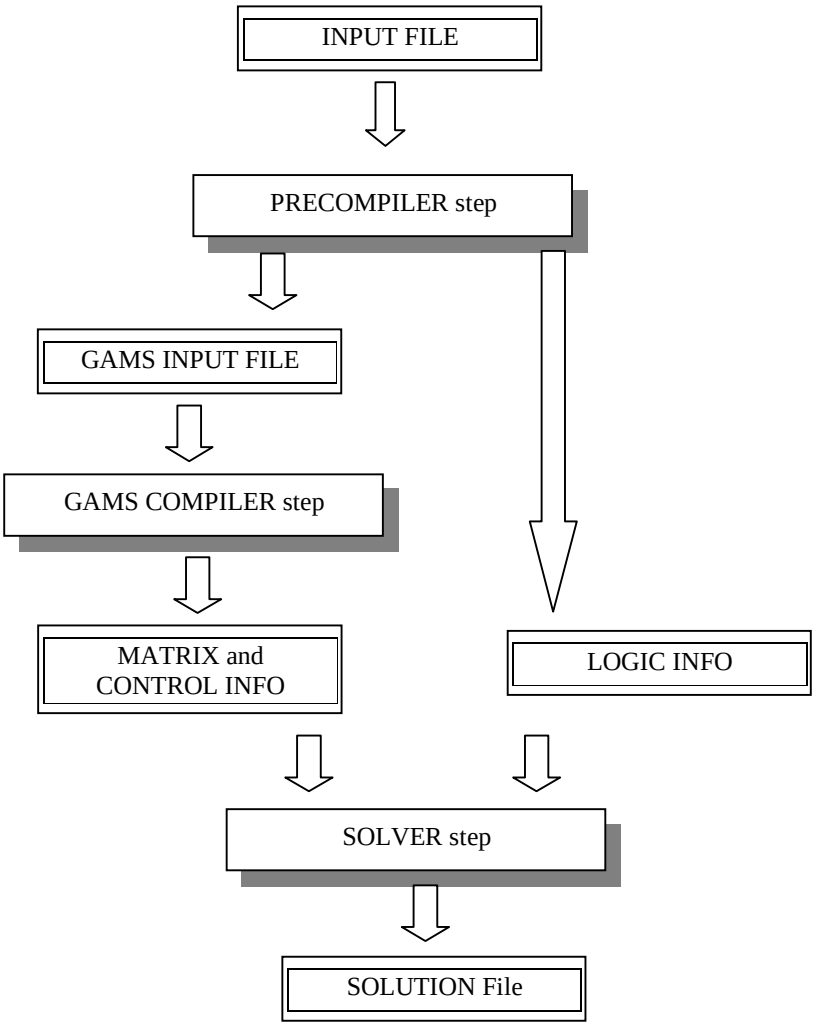


Figure 4. Processes superstructure of second example