Structural Analysis in Differential-Algebraic Systems and Combinatorial Optimization*

Mathieu Lacroix[1◇], A. Ridha Mahjoub[2], Sébastien Martin[2]

[1] LIMOS, Université Blaise Pascal Clermont-Ferrand II, Complexe Scientifique des Cézeaux, 63177 Aubière Cedex, France (lacroix@lamsade.dauphine.fr)

[2]LAMSADE, Université Paris Dauphine, Place du Maréchal De Lattre De Tassigny, 75775 Paris Cedex 16, France (mahjoub@lamsade.dauphine.fr), (martin@lamsade.dauphine.fr)

ABSTRACT

In this paper we consider the structural analysis problem for differential-algebraic systems with conditional equations. This problem consists, given a conditional differential algebraic system, in verifying if the system is well-constrained for every state, and if not to find a state in which the system is bad-constrained. We give a formulation for this problem as an integer linear program. This is based on a transformation of the problem to a matching problem in an auxiliary graph. We also show that the linear relaxation of that formulation can be solved in polynomial time. Using this, we develop a branch-and-cut algorithm for solving the problem.

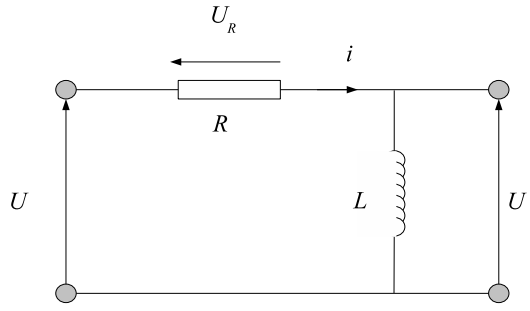
Keywords: Differential algebraic system, structural analysis, graph, integer program, matching, branch-and-cut.

1. Introduction

Differential-algebraic systems (DAS) are used for modeling complex physical systems as electrical networks and dynamic mouvements. Such a system can be given as

$$F(x, \dot{x}, u, p, t) = 0 \tag{1}$$

where x is the variable vector, u is the input vector, p is the parameter vector and t is time. As example, consider for instance the electrical circuit RL of Figure 1. This network contains a coil L and a resistance R in series. The voltage U and current i at the terminal of the self are unknown.



R : total resistance
i : current
U : voltage
L : inductance of the coil

Fig. 1: Electrical circuit.

For this circuit we can associate the following differential system :

$$\frac{di}{dt} = \frac{U}{L},$$

$$E(t) = U + Ri,$$

where $E(t)$ is a time function.

Establishing that a DAS definitely is not solvable can be helpful. A necessary (but not sufficient) criteria for solvability is that the number of variables and equations must agree. Object-oriented modeling langages like Modelica [5] enforce this as simulation is not possible if this is not the case. Thus before solving a DAS, it is utile to verify if there are as many equations as variables, and if there exists a mapping between the equations and the variables in such a way that each equation is related to only one variable and each variable is related to only one equation. If this is satisfied, then we say that the system is *well-constrained*. Otherwise, the system is said to be *bad-constrained*. A bad-constrained system is also refered as *structurally singular system*. The *structural analysis problem* (SAP) of a DAS consists in verifying if the system is well-constrained.

The structural analysis problem has been considered in the literature for nonconditional DASs. In [10, 11], Murota develops a graph theoretical method for the structural analysis of a system of equations. He introduces a formulation of the problem in terms of bipartite graphs, and shows that a system of equations is well-constrained if and only if there exists a perfect matching in the corresponding bipartite graph. He also introduces graph decomposition techniques that permit to identify the well and bad-constrained subsystems. This reduces to decomposing the associated graph into strong connected components. Such a decomposition can also be realized using the well known Dulmage and Mendelsohn decomposition [3]. Murota also studied extensions of his approach and further aspects within the framework of matroids and matrices. In [14], Reibig and Feldmann propose a structural method for solving DASs of the form

$$F(x, \dot{x}) = f(t). \tag{2}$$

which are generated from the description of electrical networks. The method is based on graph and matroid theory. It permits to reduce the problem to the determination of

978-1-4244-4136-5/09/$25.00 2009 IEEE

*Supported by the project ANR-06-TLOG-26-01 PARADE

◇Current address: LAMSADE, Université Paris Dauphine, Place du Maréchal De Lattre De Tassigny, 75775 Paris Cedex 16, France

which is called a fundamental circuit of a matroid, induced by a certain bipartite graph. Jian-Wan et al [7] and Nilsson [13] consider the SAP in relation with Modelica models. A Modelica source code is first translated into a so-called "flat model" which is a system of equations of type 1. In [7], the authors propose a method for analysing and detecting minimal bad-constrained subsystems. The method uses Dulmage et Mendelsohn decomposition techniques in a first step to isolate the bad-constrained subsets of equations. Then for each such subsystem, a set of fictitious equations is formulated. These are related to the underlaying physical system. The resulting system of equations is in turn decomposed and so on until a minimal bad-constrained subsystem is detected. The method is applied in a recursive way until all the minimal bad-constrained components are localized. In [13], the author studies the SAP for modular systems of equations, that is systems which are constructed by composition of individual equation system fragments. Leitold and Hangos [8] consider the DAS for dynamic process models. These are DAS which are sometimes difficult to solve numerically due to index problems [1]. They propose a graph-theoretical method for analysing the differential index and the structural solvability of these models. The method is an extension of Murota's approach [10], where a representation graph is considered for each differential index.

To the best of our knowledge, the SAP has not been considered for DAS with conditional equations. This paper is concerned with this extension of the problem. The purpose of the paper is to propose a model and a resolution approach for the problem in this case.

The paper is organized as follows. In Section 2, we discuss the relation between DAS and matchings. In Section 3, we give a graph representation of SAP for conditional DAS. An integer programming formulation is proposed in Section 4. A polynomial time algorithm for solving the linear relaxation of this formulation is discussed in Section 5. And in Sections 6 and 7, we study an extension of our approach to DAS with embedded conditions.

2. Differential algebraic systems and matchings

A *matching* of a graph is a subset of edges such that no two edges share a common node. Matchings have shown to be useful for modeling various discrete structures [9]. A well known and widely studied problem in combinatorial optimization is the matching problem. this consists, given a graph $G = (V, E)$, in finding a matching with maximum cardinality [9],[4]. A graph is called *bipartite* if the vertices can be divided into two disjoint sets U and V such that every edge connects a node in U to one in V, that is, U and V are independent sets. A matching M of a bipartite graph $G = (U \cup V, E)$ such that $|U| = |V| = n$ is called *perfect* if $|M| = n$. As mentioned above, the structural analysis of a DAS is a first and necessary step before analysing the system by simulating. The aim of this step is to find out whether the system is well-constrained. Given a DAS, one can associate a bipartite graph $G = (U \cup V, E)$ where U corresponds to the equations, V to the variables, and there is an edge $u_i v_i \in E$ between a node $u_i \in U$ and a node $v_i \in V$ if the variable corresponding to v_i appears in the equation corresponding to u_i. Graph G is called *incidence graph*. Therefore checking if the system is well-constrained can be realized by calculating a perfect matching in the associated incidence graph. If such a matching does not exist, then the underlaying DAS is bad-constrained [10].

Consider, for instance, the following DAS :

$$\begin{array}{ll} eq_1: & 0 = x + 4y, \\ eq_2: & 0 = 2u + z + \dot{v}, \\ eq_3: & 0 = 3a + 3z + \dot{z}, \\ eq_4: & 0 = x + 4u, \\ eq_5: & 0 = 2w + v + \dot{y}, \\ eq_6: & 0 = 3w + 3u + \dot{z}, \\ eq_7: & 0 = 3a + 3u + \dot{w}. \end{array} \tag{3}$$

Let (3') be the DAS obtained from (3) by replacing the equation eq_6 by the equation :

$$eq'_6: \quad 0 = 3x + 3u$$

The incidence graphs corresponding to DAS (3) and (3') are shown in Figure 2 (a) and (b), respectively. Here nodes $u_1, ..., u_6, u'_6, u_7$ are associated with equations $eq_1, ..., eq_6, eq'_6, eq_7$ and nodes $v_1, ..., v_7$ are associated with variables a, u, v, w, x, y, z, respectively. A perfect matching is displayed in bold edges in Figure 2 (a), implying that system (3) is well-constrained. However, the maximum matching of the incidence graph corresponding to system (3'), displayed in Figure 2 (b), is not perfect, which implies that system (3') is bad-constrained.

For more details on the relation between simple (nonconditional) DASs and matchings see [10, 11].

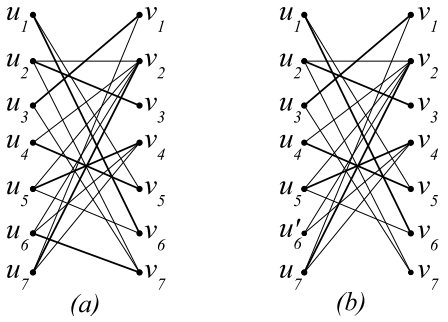


Fig. 2: Incidence graphs.

3. The SAP for conditional DAS

In many practical situations, physical systems have different states generated by some technical conditions. These may be, for instance, related to temperature changements in hydraulic systems. These physical systems generally yield DASs with conditional equations. Each conditional equation can generate several equations. In this section we discuss the SAP for these more general DASs. We will suppose, without loss of generality, that the conditional equations of the system all generate the same number of equations, for each combination of conditions. Otherwise,

the system would be bad-constrained. One can easily detect a conditional equation yielding less or more variables than equations, if there is any. Moreover we will suppose that each conditional equation may generate only one equation. If this is not the case then one can expand the system to a one satisfying this. In fact, if a conditional equation eq_{cond} generates k equations with respect to a condition $cond$, then it can be expanded to k conditional equations $eq_1, ..., eq_k$ such that each equation eq_i may generate the i^th equation of eq_{cond} according to condition $cond$ whether it is true or false. For example the conditional equation

$$\begin{array}{l}\text{if } a > 0\\ \qquad \text{then}\\ \qquad\quad 0 = 2x + \dot{y}^2,\\ \qquad\quad 0 = \dot{x} + y - 4,\\ \qquad \text{else}\\ \qquad\quad 0 = x^2 + \dot{y} + 2,\\ \qquad\quad 0 = 6\dot{x} + \dot{y} + 3,\end{array}$$

can be expanded to

$$\begin{array}{l}\text{if } a > 0\\ \qquad \text{then} \quad 0 = 2x + \dot{y}^2,\\ \qquad \text{else} \quad 0 = x^2 + \dot{y} + 2,\\ \text{if } a > 0\\ \qquad \text{then} \quad 0 = \dot{x} + y - 4,\\ \qquad \text{else} \quad 0 = 6\dot{x} + \dot{y} + 3.\end{array}$$

A conditional DAS may then have different forms depending on the set of conditions that hold each assignment. Here we consider conditional DAS such that any conditional equation may take two possible values, depending on whether the associated condition is true or false and may generate only one equation. The equations generated by a conditional equation will be refered as *simple equation*. Each assignment of the values true and false to the conditions yield a nonconditional system, called a *state* of the system.
Consider for example the following DAS:

$$\begin{array}{ll} eq_1: & \text{if } a > 0\\ & \qquad \text{then} \quad 0 = 4x^2 + 2\dot{x} + 4y + 2,\\ & \qquad \text{else} \quad 0 = \dot{y} + 2z + 4,\\ eq_2: & \text{if } b > 0\\ & \qquad \text{then} \quad 0 = 6\dot{y} + 2\dot{z} + 2,\\ & \qquad \text{else} \quad 0 = x + \dot{y} + 1,\\ eq_3: & \text{if } c > 0\\ & \qquad \text{then} \quad 0 = 6\dot{x} + y + 2,\\ & \qquad \text{else} \quad 0 = 3\dot{y} + z + 3.\end{array} \tag{4}$$

For conditions $a > 0$, $b > 0$, $c > 0$, system (4) is nothing but the following.

$$\begin{array}{ll} eq_1: & 0 = 4x^2 + 2\dot{x} + 4y + 2,\\ eq_2: & 0 = 6\dot{y} + 2\dot{z} + 2,\\ eq_3: & 0 = 6\dot{x} + y + 2.\end{array} \tag{5}$$

And conditions $a > 0$, $b > 0$, $c \leq 0$ yield the system.

$$\begin{array}{ll} eq_1: & 0 = 4x^2 + 2\dot{x} + 4y + 2,\\ eq_2: & 0 = 6\dot{y} + 2\dot{z} + 2,\\ eq_3: & 0 = 3\dot{y} + z + 3.\end{array} \tag{6}$$

Figure 3 shows the incidence bipartite graphs for systems (5) and (6).

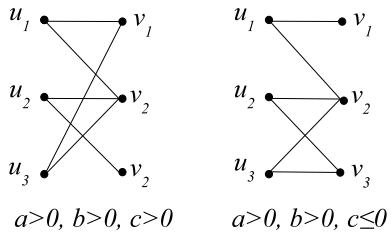


Fig. 3: Incidence graphs.

So given a conditional DAS, the associated SAP consists in verifying whether the system is well-constrained in any state, and if not, in finding a state in which the system is bad-constrained. The SAP for a conditional DAS thus reduces to verifying whether or not the bipartite graph related to any state of the system contains a perfect matching. A difficulty arises here is that the number of possible states may be exponential. So the approaches used so far for simple DASs cannot, unfortunately, be applied for that problem. A more efficient method is this needed for solving it. In the rest of this section we shall discuss a graph based model for the problem. Given a conditional DAS with n equations, say $eq_1, ..., eq_n$, and n variables, say $x_1, ..., x_n$, we consider a bipartite graph $G = (U \cup V, E)$ where $U = \{u_1, ..., u_n\}$ (resp. $V = \{v_1, ..., v_n\}$) is associated with the equations (resp. variables). Between a vertex $u_i \in U$ and a vertex $v_j \in V$ we consider an edge, called *true edge* (resp. *false edge*), if the variable x_j appears in equation eq_i, when the condition of eq_i is supposed true (resp. false). Let E_i^t (resp. E_i^f) be the set of true (resp. false) edges incident to u_i, for all $i = 1, ..., n$. Let

$$E = \bigcup_{i=1,...,n} (E_i^t \cup E_i^f).$$

Figure 4 shows the graph G associated with system (4).

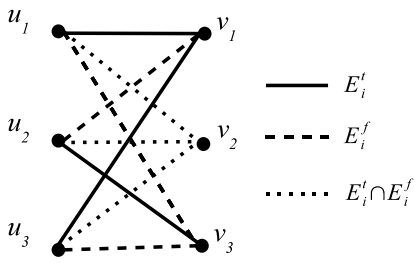


Fig. 4: Graph representing system (4).

Now, the SAP reduces to finding whether or not there exists a subgraph of G, say G', containing, for each node u_i, either E_i^t or E_i^f, and which does not have a perfect matching. We will refer to this problem as the *free perfect matching*

subgraph problem (FPMSP). In the following section we will discuss an integer programming formulation for this problem.

4. Formulation

Integer programming [6] is one of the powerful tools of mathematical programming and combinatorial optimization. Several problems from various domains can be formulated as integer programs. Effective methods have been developed for formulating, analysing and solving these problems. In what follows we will propose an integer programming based model for the FPMSP, and thus for the underlaying SAP.

With a vertex $u_i \in U$ let us associate a binary variable $x(u_i)$ which takes 1 if E_i^t is contained in G' and 0 if E_i^f is contained in G', that is, $x(u_i) = 1$ if the condition of equation eq_i is true and 0 if not.

Let M be a perfect matching of G. Let $x \in \{0,1\}^U$ such that $x(u_i) = 1$ if $M \cap E_i^t \neq \emptyset$ and $x(u_i) = 0$ if $M \cap E_i^f \neq \emptyset$, that is subgraph G' induced by x contains a perfect matching. Therefore x satisfies the following equation

$$\sum_{u_iv_j \in M \cap E_i^t} x(u_i) + \sum_{u_iv_j \in M \cap E_i^f} (1 - x(u_i)) = n.$$

Thus, by considering the following constraint, one can discard this non feasible solution.

$$\sum_{u_iv_j \in M \cap E_i^t} x(u_i) + \sum_{u_iv_j \in M \cap E_i^f} (1 - x(u_i)) \leq n - 1.$$

In consequence, the FPMSP is equivalent to the following integer program.

$$\max 0x \tag{3}$$

$$\sum_{u_iv_j \in M \cap E_i^t} x(u_i) + \sum_{u_iv_j \in M \cap E_i^f} (1 - x(u_i)) \leq n - 1 \quad \text{for all } M \in \mathcal{M}, \tag{4}$$

$$0 \leq x(u_i) \leq 1, \quad \text{for all } u_i \in U, \tag{5}$$

$$x(u_i) \in \{0,1\}, \quad \text{for all } u_i \in U. \tag{6}$$

Here $\mathcal{M}$ is the set of perfect matchings.

Indeed, FPMSP has a "yes" answer, that is the underlaying DAS is well-constrained, if and only if, the program above has no a feasible solution. We will denote the program above by (P). Hence, in order to solve FPMSP, one can use integer programming tools for solving (P). One of the powerful techniques in integer programming and combinatorial optimization is the so called *polyedral approach*. This consists in reducing the resolution of the program to a sequence of linear programs. This approach is based on a deep investigation of the convex hull of the solutions of the problem.

A drawback of the model given by (3)-(6) is that the polyhedron given by inequalities (4)-(6) may be empty (if the system is well-constrained). The polyhedral approach based on that model, would not then be appropriate. In order to avoid this situation and always work with a feasible program, we are going to slightly modify program (P).

Consider the following program (Q), where y is a new non-negative variable.

$$\min y \tag{7}$$

$$\sum_{u_iv_j \in M \cap E_i^t} x(u_i) + \sum_{u_iv_j \in M \cap E_i^f} (1 - x(u_i)) - y \leq n - 1 \quad \text{for all } M \in \mathcal{M}, \tag{8}$$

$$0 \leq x(u_i) \leq 1, \quad \text{for all } u_i \in U, \tag{9}$$

$$0 \leq y, \tag{10}$$

$$x(u_i) \in \{0,1\}, \quad \text{for all } u_i \in U. \tag{11}$$

Clearly, x is a solution of (P) if and only if $(x, 0)$ is a solution of (Q). Thus in order to solve problem (P), one can solve problem (Q). If (x, y) is an optimal solution of (Q) with $y \neq 0$, then the system in question is well-constrained for all states, and if $y = 0$ then the state induced by x is bad-constrained.

As in program (P), the number of inequalities in (Q) may be exponential. in order to solve (Q) using a polyhedral approach, one needs an efficient algorithm for separating inequalities (8). In the following section we devise a polynomial time separation algorithm for these inequalities.

5. Separation

The *separation problem* for inequalities (8) consists, given a solution $(x^*, y^*) \in \mathbb{R}_+^U \times \mathbb{R}_+$, to determine whether (x^*, y^*) satisfies inequalities (8), and if not to find an inequality violated by (x^*, y^*). An algorithm which solves this problem is called a *separation algorithm*. In what follows, we will give a polynomial time separation algorithm for inequalities (8). This will imply that the linear relaxation of problem (Q) can be solved in polynomial time [6].

Let $(x^*, y^*) \in \mathbb{R}_+^U \times \mathbb{R}_+$. With an edge $u_iv_j \in E$ associate the weight $x(u_i)$ if $u_iv_j \in E_i^t \backslash E_i^f$, $1 - x(u_i)$ if $u_iv_j \in E_i^f \backslash E_i^t$ and 1 if $u_iv_j \in E_i^t \cap E_i^f$. If the maximum weight of a perfect matching M in G, with respect to these weights, is greater than $y^* + n - 1$, then the inequality of type (8), corresponding to M, is violated. Otherwise, all the inequalities of type (8) are satisfied.
For instance, consider, system (4) and the solution $x(u_1) = 0,7$, $x(u_2) = 0,4$, $x(u_3) = 0,3$, $y = 0,2$. Therefore we associate with the edges of the corresponding bipartite graph in Figure 4 the following weights $x^*(u_1v_1) = 0.7$, $x^*(u_1v_3) = 0.3$, $x^*(u_2v_1) = 0.6$, $x^*(u_2v_3) = 0.4$, $x^*(u_3v_1) = 0.3$, $x^*(u_3v_3) = 0.7$ $x^*(u_1v_2) = x^*(u_2v_2) = x^*(u_3v_3) = 1$ and set $y^* = 0.2$. The maximum perfect matching with respect to x^* is $\{u_1v_1, u_2v_2, u_3v_3\}$ with weight 2.4. Here we have that the inequality

$$x(u_1) + x(u_2) + 1 - x(u_2) + 1 - x(u_3) - y \leq 2$$

is violated. Thus, the separation problem for inequalities (8) reduces to computing a maximum weight perfect matching in a bipartite graph. Moreover, this can be solved in polynomial time [9].

6. Extension : DASs with embedded conditions

The approach developed above can be extended for integrating embedded conditions. DASs with embedded con-

ditions are more complex to handle. In this section we discuss this generalization.

Consider for example the following DAS.

$$
\begin{array}{lll}
eq_1: & \text{if } a>0 & \\
& \text{then} & \\
& & \text{if } b>0 \\
& & \quad \text{then } \ 0=\dot{y}, \\
& & \quad \text{else } \ 0=3x, \\
& \text{else} & \\
& & \text{if } c>0 \\
& & \quad \text{then } \ 0=3z+\dot{y}, \\
& & \quad \text{else } \ 0=v, \\
eq_2: & \text{if } d>0 & \\
& \text{then} & \\
& & \text{if } e>0 \\
& & \quad \text{then } \ 0=4y+\dot{x}, \\
& & \quad \text{else } \ 0=z, \\
& & \text{if } f>0 \\
& & \quad \text{then } \ 0=x, \\
& & \quad \text{else } \ 0=v+z, \\
& \text{else } & 0=x+\dot{v}+1, \\
& & 0=v+\dot{y}+1, \\
eq_3: & \text{if } a>0 & \\
& \text{then } & 0=6\dot{x}+2, \\
& \text{else } & 0=3\dot{y}+z+3+v.
\end{array}
\qquad (7)
$$

Observe that eq_2 can be decomposed into two sub-equations :

$$
\begin{array}{lll}
eq_2': & \text{if } d>0 & \\
& \text{then} & \\
& & \text{if } e>0 \\
& & \quad \text{then } \ 0=4y+\dot{x}, \\
& & \quad \text{else } \ 0=z, \\
& \text{else } & 0=x+\dot{v}+1,
\end{array}
$$

$$
\begin{array}{lll}
eq_2'': & \text{if } d>0 & \\
& \text{then} & \\
& & \text{if } f>0 \\
& & \quad \text{then } \ 0=x, \\
& & \quad \text{else } \ 0=v+z, \\
& \text{else } & 0=v+\dot{y}+1.
\end{array}
$$

Therefore, system (7) can be expressed by $(eq_1, eq_2', eq_2'', eq_3)$ Thus any conditional equation of a DAS with embedded conditions may decompose into smaller *conditional sub-equations*. In the sequel we consider systems that are minimal, that is to say in which the conditionals equations cannot decompose anymore. As for the non-embedded DAS, we suppose that each conditional equation generates the same number of simple equations for each combination of conditions. We also suppose, without loss of generality, that every conditional equation, may generate exactly one simple equation. Given a DAS with embedded equations we will call *combination of conditions* any assignment of the values true and false to the conditions of an equation of the system. By our hypothesis any combination of condition yields only one simple equation.

For instance, in system (7), the equation $0=\dot{y}$ in eq_1 is the result of the combination $\{a>0, b>0\}$ and $0=v+z$ in eq_2'' is the result of the combination $\{d>0, f\geq 0\}$.

Let $C_1, ..., C_m$ be the possible combinations of conditions. Remark that the number of combinations C_k is polynomial in the size of the system (number of simple equations and variables).

For system (7), we have the following combinations : $C_1 = \{a>0,\ b>0\}$, $C_2 = \{a>0,\ b\leq 0\}$, $C_3 = \{a\leq 0,\ c>0\}$, $C_4 = \{a\leq 0,\ c\leq 0\}$, $C_5 = \{d>0,\ e>0\}$, $C_6 = \{d>0,\ e\leq 0\}$, $C_7 = \{d>0,\ f>0\}$, $C_8 = \{d>0,\ f\leq 0\}$, $C_9 = d\leq 0$, $C_{10} = a>0$ and $C_{11} = a\leq 0$. For $i=1,...,m$, let R_i be the set of combinations of conditions that are implied by C_i, that is the combinations that are satisfied if C_i so is. For instance, for the example above, $C_{10}=\{a>0\}$ is implied by $C_1=\{a>0, b>0\}$. For $i=1,...,m$, let T_i be the set of combinaisons C_j which are incompatible with C_i, that is C_i and C_j cannot occur at the same time. For the example above C_2 and C_4 are incompatible.

Consider the bipartite graph $G=(U\cup V, E)$ where $U=\{u_1,...,u_n\}$ corresponds to the conditional equations of the system, $V=\{v_1,...,v_n\}$ corresponds to the variables, and we consider an edge u_iv_i between two nodes $u_i\in U$ and $v_i\in V$ if the variable v_i appears in a simple equation that may be generated by the conditional equation corresponding to u_i, when a combination C_k of conditions holds.

Let E_k be the set of edges implied by combinations C_k, for $k=1,...,n$. Thus

$$E=\bigcup_{k=1,...,m} E_k.$$

For system (7), the edge sets E_k, generated by combination $C_1,...,C_{11}$, are $E_1=\{\ u_1v_2\ \}$, $E_2=\{\ u_1v_1\ \}$, $E_3=\{\ u_1v_3,\ u_1v_2\ \}$, $E_4=\{\ u_1v_4\ \}$, $E_5=\{\ u_2v_2,\ u_2v_1\ \}$, $E_6=\{\ u_2v_3\ \}$, $E_7=\{\ u_3v_1\ \}$, $E_8=\{\ u_3v_4,\ u_3v_3\ \}$, $E_9=\{\ u_2v_1,\ u_2v_4,\ u_3v_4,\ u_3v_2\ \}$, $E_{10}=\{\ u_4v_1\ \}$ and $E_{11}=\{\ u_4v_2,\ u_4v_3,\ u_4v_4\ \}$. And the incidence graph is given in Figure 5. Note that nodes u_2, u_3 correspond to the conditional equations eq_2', eq_2''. The number on each edge corresponds to the set E_k to which belongs the edge.

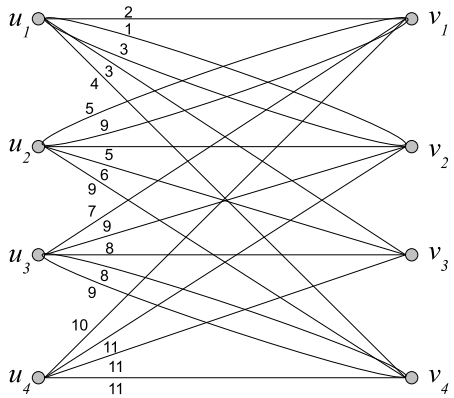


Fig. 5: Graph G.

Thus the DAS is bad-constrained if and only if there exists a family $\mathscr{F}$ of sets E_k whose edges cover the nodes of U

and such that the subgraph induced by these edges does not contain a perfect matching. Moreover, $\mathscr{F}$ must satisfy the following :
a) if $E_i, E_j \in \mathscr{F}$, then C_i and C_j are compatible, and
b) if $E_i \in \mathscr{F}$ and $C_j \in R_i$, then $E_j \in \mathscr{F}$.
One can easily verify, for graph G of Figure 5 that the sets E_1, E_6, E_7, E_{10} constitute a feasible family that covers the nodes $u_1, ..., u_4$ and verifies the conditions a), b) above. However the subgraph induced by there sets does not contain a perfect matching. In consequence, the family of conditions C_1, C_6, C_7, C_{10} induces a bad-constrained system.

Given a DAS with embedded conditions we will call *state* of the system any set of combinations C_i covering all the equations of the system and whose edge sets E_i verify conditions a), b) above.

As for the DASs studied in the previous section, the problem here is to find a state in which the system is bad-constrained or to show that the system is well-constrained in any state. So in graph G this consists in finding a subgraph induced by a state of the system which does not contain a perfect matching, or to show that for any state the corresponding graph contains such a matching.

7. Formulation

With every set of edges E_i we associate a binary variable x_i such that $x_i = 1$ if C_i is considered in the state of the system and $x_i = 0$ if not. Clearly, if x represents a state of the system, then x satisfies the following inequalities

$$\begin{aligned} x_i + x_k &\leq 1, && \text{for all } k \in R_i, \text{ for } i = 1, ..., n, \\ x_k &\leq x_i, && \text{for all } k \in T_i, \text{ for } i = 1, ..., n. \end{aligned}$$

Moreover if M is a perfect matching of G, then x satisfies the equation

$$\sum_{j=1}^{m} |E_j \cap M| \, x_j = n.$$

Thus, by considering the following constraint, one can discard this solution.

$$\sum_{j=1}^{m} |E_j \cap M| \, x_j \leq n - 1.$$

Consider the following integer program $(\widehat{P})$.

$$\max y \tag{12}$$

$$\sum_{j=1}^{m} |E_j \cap M| \, x_j \leq n - 1, \quad \text{for all } M \in \mathcal{M}, \tag{13}$$

$$x_i + x_k \leq 1, \quad \text{for all } k \in R_i, \tag{14}$$

$$x_k \leq x_i, \quad \text{for all } k \in T_i, \tag{15}$$

$$\sum_{\substack{i=1,...,n \\ E_k \cap \, \delta(u_i) \neq \emptyset}} x_k \geq y, \quad \text{for all } u_i \in U, \tag{16}$$

$$0 \leq x_k \leq 1, \quad \text{for all } E_k \in E, \tag{17}$$

$$0 \leq y, \tag{18}$$

$$x_k \in \{0, 1\}, \quad \text{for all } E_k \in E. \tag{19}$$

We claim that the SAP is equivalent to program $(\widehat{P})$. In fact, let (x, y) be a solution of $(\widehat{P})$. If $y \geq 1$, then by constraints (16) each node of U has at least one incident edge in the bipartite graph induced by x. As this subgraph does not contain a perfect matching, this implies that the system is bad-constrained. If $y < 1$, as the program is to maximize y, there should not exist a state of the system such that the corresponding edge sets E_i cover the nodes of U and do not induce a perfect matching. This implies that the system is well-constrained. Clearly, constraints (14)-(18) can be separated in polynomial time (by enumeration). For constraints (13), the separation problem can be solved to the maximum weight matching problem in a bipartite graph.

8. Conclusion

In this paper we have studied the SAP for conditional DASs. We have proposed integer programming formulations for the problem for both the embedded and nonembedded cases. We have shown that the linear relaxations of these models can be solved in polynomial time. Based on these formulations, we are now going to develop branch-and-cut algorithms for solving the problem in both versions with and without embedded conditions.

Acknowledgments

We would like to thank Sébastien Furic, Djilali Talbi and Bruno Lacabanne from LMS-Imagine for stimulating discussions. This work has been supported by the project ANR-06-TLOG-26-01 PARADE. The financial support is much appreciated.

REFERENCES

[1] K. E. Brenan, S. L. Campbel, L. R. Petzold, "Numerical solution of initial value problems in differential-algebraic equations", *New York: North-Holland*, 1989.

[2] I. Duff, "Analysis of sparse systems", *Thesis - Oxford university*, 1972.

[3] A. L. Dulmage, N. S. Mendelsohn, "Coverings of bipartite graphs", *Canadian Journal of Mathematics*, 1963, pp. 517-534.

[4] J. Edmonds, "Maximum matching and a polyhedron with 0,1-vertices", *J. Res. Nat. Bur. Standards 69B*, 1965, pp. 125-130.

[5] P. Fritzson, "Principles of Obeject-Oriented Modeling and Simulation with Modelica 2.1", *Wiley-Interscience*, 2003.

[6] M. Grtschel, L. Lovasz, A. Schrijver, "The ellipsoid method and its consequences in combinatorial optimization", *Combinatorica 1*, 1981, pp. 169-197.

[7] D. Jian-Wan, C. Li-Ping, Z. Fan-Li, W. Yi-Zhong, W. Guo-Biao, "An Analyzer for Declarative Equation Based Models", *Modelica 2006*, 2006, pp. 349-357.

[8] A. Leitold, K. M. Hangos, "Structural solvability analysis of dynamic process models", *Computeurs and Chemical Engineering 25*, 2001, pp. 1633-1646.

[9] L. Lovasz, M. D. Plummer, "Matching Theory", *North-Holland*, 1986.

[10] K. Murota, "Systems Analysis by Graphs and Matroids", *Springer-Verlag*, 1987.

[11] K. Murota, “Matrices and Matroids for Systems Analysis”, *Springer-Verlag*, 2000.

[12] G. L. Nemhauser, L. A. Wolsey, “Integer and Combinatorial Optimization”, *A Wiley-Interscience Publication*, 1988.

[13] H. Nilsson, “Type-Based Structural Analysis for Modular Systems of Equations”, *Proceedings of the 2nd International Workshop on Equation-Based Object-Oriented Languages and Tools*, 2008, pp. 71-81.

[14] G. Reibig, U. Feldmann, “A simple and general method for detecting structural inconsistencies in large electrical networks”, *Circuits and Systems I: Fundamental Theory and Applications*, 2002, pp. 237-240.

[15] J. Unger, A. Kroner, W. Marquardt, “Structural analysis of differential-algebraic equation systems - theory and application”, *Computeurs and Chemical Engineering*, 1995, pp. 867-882.