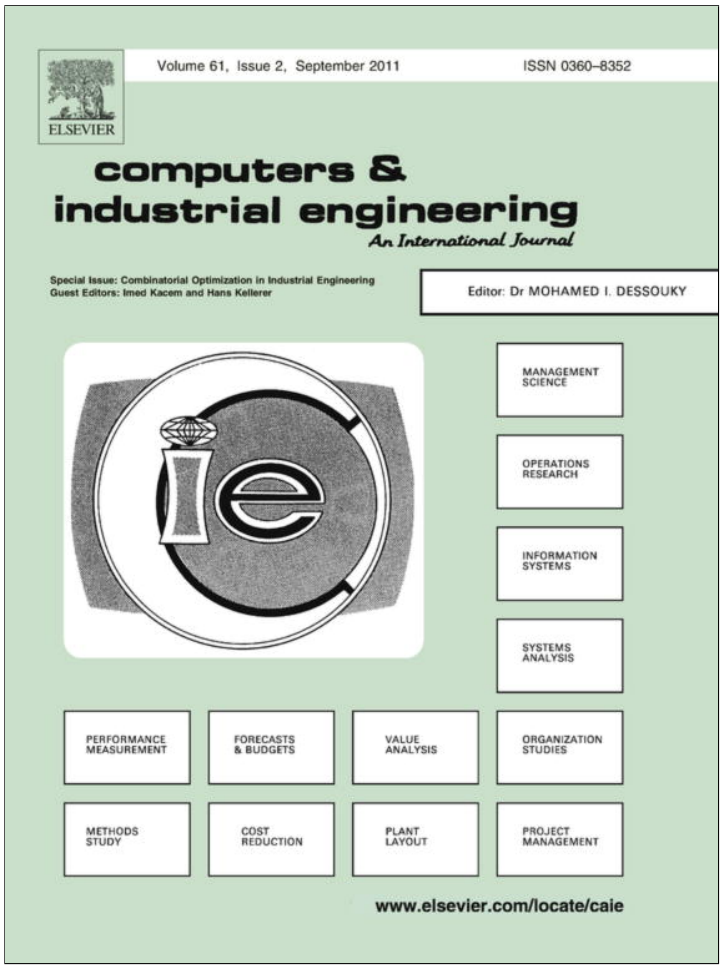

Computers & Industrial Engineering 61 (2011) 422–429

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Combinatorial optimization model and MIP formulation for the structural analysis of conditional differential-algebraic systems

Mathieu Lacroix [a,b], A. Ridha Mahjoub [a,*], Sébastien Martin [a]

[a] LAMSADE, Université Paris-Dauphine, Place du Maréchal De Lattre De Tassigny, 75775 Paris Cedex 16, France
[b] LIMOS, Université Blaise Pascal Clermont-Ferrand II, Complexe Scientifique des Cézeaux, 63177 Aubière Cedex, France

ARTICLE INFO

Article history:
Available online 9 December 2010

Keywords:
Differential-algebraic system
Structural analysis
Graph
Integer linear program
Matching
Branch-and-Cut algorithm

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 structurally nonsingular for every state, and if not in finding a state in which the system is structurally singular. We give a formulation for this problem as an integer linear program. This is based on a transformation of the problem into 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 and present some experimental results.

1. Introduction

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

$$\begin{aligned} &f_1(x,\dot{x},p)=0,\\ &f_2(x,\dot{x},p)=0,\\ &\vdots\\ &f_n(x,\dot{x},p)=0, \end{aligned} \tag{1}$$

where $x=(x_1,x_2,\ldots,x_n)$ is the variable vector, $\dot{x}=(\dot{x}_1,\dot{x}_2,\ldots,\dot{x}_n)$ is the *derivative* vector associated to x with respect to time and $p=(p_1,p_2,\ldots,p_m)$ is the vector of the *input values* of the problem. We will mean by input values what is called in DAS literature as *input* vector and *parameter* vector.

As example, consider for instance the electrical circuit *RL* of Fig. 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.

For this circuit we can associate the following differential system:

$$\begin{aligned} &\dot{i}-\frac{1}{L}U=0,\\ &Ri+U-E(t)=0. \end{aligned}$$

Here the DAS has two equations where U and i are the variables and $E(t)$ is a known time function.

Establishing that a DAS definitely is not solvable can be helpful. A typical problem that appears, for instance, in physical system modeling and simulation is when too many or too few equations are specified in the system, thus leading to inconsistent states of the model. In consequence, a necessary (but not sufficient) condition for solvability is that the number of variables and equations must agree. Object-oriented modeling languages like Modelica (Fritzson, 2003) enforce this as simulation is not possible if this is not the case.

The index matrix M of a DAS (Poulsen, 2001) is the $n\times n$ matrix given by

$$M_{i,j}=\begin{cases} 1 & \text{if } \dot{x}_i \text{ appears in equation } j,\\ 0 & \text{if } x_i \text{ appears in equation } j \text{ and not } \dot{x}_i,\\ -\infty & \text{if both } \dot{x}_i \text{ and } x_i \text{ do not appear in equation } j. \end{cases}$$

Note that $M_{i,j}$ consists of the leading derivative of variable x_i in equation j.

Definition 1.1 Poulsen (2001). A DAS is called structurally nonsingular if there exists a permutation of rows and columns of its index matrix M so that the entire diagonal of the resulting matrix, say $\overline{M}$, only contains non-infinity elements, i.e.,

$$\overline{M}_{i,i}\neq -\infty, \text{ for } i=1,\ldots,n.$$

Otherwise, the system is called structurally singular.

As it is mentioned in Pantelides (1988), Poulsen (2001), Bunus and Fritzson (2008) structurally singular systems have no solution since it is impossible to find consistent initial conditions. Thus, a

* Corresponding author.
E-mail addresses: lacroix@lamsade.dauphine.fr (M. Lacroix), mahjoub@lamsade.dauphine.fr (A. Ridha Mahjoub), martin@lamsade.dauphine.fr (S. Martin).

0360-8352/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cie.2010.12.002

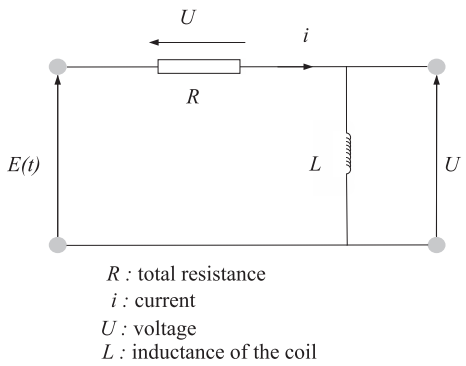

Fig. 1. Electrical circuit.

necessary condition for a DAS to be solvable is that it is structurally nonsingular. The nonsingularity does not however guarantee its solvability. The *structural analysis problem* (SAP) of a DAS consists in verifying whether or not the system is structurally nonsingular.

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. In more formal way, we call *condition* a boolean depending on the values of a particular set of input values among $p_1, \ldots, p_n$. An *embedded condition* is a condition which depends on several different *input values*. A *conditional equation* is an equation whose form depends on the values of a condition. An equation without conditions is called *nonconditional* or *simple*.

Physical systems generally yield DASs with conditional equations. A conditional equation may take different *forms*, *simple* or *embedded*. Moreover, it may generate one or more nonconditional equations. A conditional DAS is a DAS with conditional equations.

In the sequel, we will consider conditional DASs with simple conditional equations such that each equation may take two possible values, depending on whether the associated condition is true or false, and may generate only one nonconditional equation. We suppose that all the conditions are independent. That is to say the value of a condition does not depend on the value of any other condition. As it will turn out, the results we will obtain for this case can be easily extended to DAS's with arbitrary structure.

A conditional DAS may then have different forms depending on the assignment of the *true* and *false* values to the conditions. Each assignment yields a nonconditional system called a *state* of the system. Consider for example the following DAS with three equations and three variables:

$$\begin{aligned}
&C_1 : \text{if } a > 0\\
&\qquad \text{then} \quad 0 = 4x_1^2 + 2\dot{x}_1 + 4x_2 + 2,\\
&\qquad \text{else} \quad 0 = \dot{x}_2 + 2x_3 + 4,\\
&C_2 : \text{if } b > 0\\
&\qquad \text{then} \quad 0 = 6\dot{x}_2 + 2\dot{x}_3 + 2,\\
&\qquad \text{else} \quad 0 = x_1 + \dot{x}_2 + 1,\\
&C_3 : \text{if } c > 0\\
&\qquad \text{then} \quad 0 = 6\dot{x}_1 + x_2 + 2,\\
&\qquad \text{else} \quad 0 = 3\dot{x}_2 + x_3 + 3.
\end{aligned} \tag{2}$$

For conditions $a > 0$, $b > 0$, $c > 0$, system (2) is nothing but the following.

$$\begin{aligned}
&C_1 : 0 = 4x_1^2 + 2\dot{x}_1 + 4x_2 + 2,\\
&C_2 : 0 = 6\dot{x}_2 + 2\dot{x}_3 + 2,\\
&C_3 : 0 = 6\dot{x}_1 + x_2 + 2.
\end{aligned} \tag{3}$$

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

$$\begin{aligned}
&C_1 : 0 = 4x_1^2 + 2\dot{x}_1 + 4x_2 + 2,\\
&C_2 : 0 = 6\dot{x}_2 + 2\dot{x}_3 + 2,\\
&C_3 : 0 = 3\dot{x}_2 + x_3 + 3.
\end{aligned} \tag{4}$$

Definition 1.2. Given a conditional DAS, the SAP consists in verifying if the system is structurally nonsingular in every state, and if not to determine a state in which the system is structurally singular.

In Lacroix, Mahjoub, and Martin (2010), it has been shown that the SAP for DAS with conditional equations is NP-complete. To the best of our knowledge, this is the first and only work that has been done on the SAP for DASs 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. In what follows we will give some definitions needed in the sequel.

A *matching* of a graph is a subset of edges such that no two edges share a common node. Matchings have been shown to be useful for modeling various discrete structures (Lovasz & Plummer, 1986). 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 (Edmonds, 1965; Lovasz & Plummer, 1986). A *stable set* of a graph is a set of vertices no two of which are adjacent. A graph is called *bipartite* if the set of vertices can be divided into two disjoint sets U and V such that every edge connects a node in U to a node in V, that is, U and V are stable sets. A matching M of a bipartite graph $G = (U \cup V, E)$ such that $|U| = |V| = n$ is called *perfect* if $|M| = n$.

The paper is organized as follows. In Section 2 we present some related works. In particular, we discuss the relation between matchings and SAP in bipartite graphs. In Section 3 we give a graph model for the SAP for conditional DAS and an integer programming formulation. A polynomial time algorithm for solving the linear relaxation of this formulation is discussed in Section 4. In Section 5 we devise a Branch-and-Cut algorithm based on these results and present some experimental results. Sections 6 is dedicated to extensions of our approach to DASs with embedded conditions.

2. Related works and matchings

The SAP has been considered in the literature for nonconditional DASs.

Given a DAS, one can associate a bipartite graph $G = (U \cup V, E)$ where U corresponds to the variables, V to the equations, and there is an edge $u_i v_j \in E$ between a node $u_i \in U$ and a node $v_j \in V$ if the variable x_i, in the form x_i or $\dot{x}_i$, corresponding to u_i appears in the equation corresponding to v_j. Graph G is called the *incidence graph* of the DAS. Note that a DAS is structurally singular if and only if there exists a perfect matching in its incidence graph G.

As the perfect matching problem in bipartite graphs can be solved in polynomial time, the SAP for nonconditional DAS's can also be solved in polynomial time.

Consider, for instance, the following DAS with seven equations and seven variables:

$$\begin{aligned}
&C_1 : 0 = x_5 + 4x_6,\\
&C_2 : 0 = 2x_2 + x_7 + \dot{x}_3,\\
&C_3 : 0 = 3x_1 + 3x_3 + \dot{x}_7,\\
&C_4 : 0 = x_5 + 4x_2,\\
&C_5 : 0 = 2x_4 + x_2 + \dot{x}_6,\\
&C_6 : 0 = 3x_4 + 3x_2 + \dot{x}_7,\\
&C_7 : 0 = 3x_2 + \dot{x}_4.
\end{aligned} \tag{5}$$

Let (5′) be the DAS obtained from (5) by replacing the equation C_6 by the equation:

$$C_6' : 0 = 3x_1 + 3x_2.$$

The incidence graphs corresponding to DASs (5) and 5′ are shown in Fig. 2 (a) and (b), respectively. Here nodes $u_1, \ldots, u_6, u_6', u_7$ are associated with equations $C_1, \ldots, C_6, C_6', C_7$ and nodes $v_1, \ldots, v_7$ are associated with variables $x_1, \ldots, x_7$, respectively. A perfect matching is displayed in bold edges in Fig. 2 (a), implying that system (5) is structurally nonsingular. However, the maximum matching of the incidence graph corresponding to system (5′), displayed in Fig. 2 (b), is not perfect, which implies that system (5′) is structurally singular.

Several graph-theoretical techniques (which may be considered as variants or extensions of the above idea) have been proposed for approaching the SAP (Bujakiewicz, 1994; Duff & Gear, 1986; Murota, 2000). Murota (Murota, 2000) also introduces graph decomposition techniques that permit to identify structurally singular and nonsingular subsystems structurally nonsingular and singular. This reduces to decomposing the associated oriented graph into strong connected components. Given a maximal matching M in the bipartite incidence graph $G = (U \cup V, E)$, the associated oriented graph is obtained from G by orienting all the edges of M from U to V and all the other edges from V to U. This decomposition is unique. Such a decomposition can also be obtained using the well known Dulmage and Mendelsohn decomposition (Dulmage & Mendelsohn, 1963). Murota (Murota, 2000) also studies extensions of his approach and further aspects within the framework of matroids and matrices. In Reibig and Feldmann (2002), Reibig and Feldmann propose a method for solving the SAP of nonconditional DAS 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 what is called a *fundamental circuit* of a matroid, induced by a certain bipartite graph. Jian-Wan, Li-Ping, Fan-Li, Yi-Zhong, and Guo-Biao (2006) and Nilsson (2008) consider the SAP in relation with Modelica models. A Modelica source code is first translated into a so-called "flat model" which is nothing but a DAS. In Jian-Wan et al. (2006), the authors propose a method for analysing and detecting minimal structurally singular subsystems. The method uses Dulmage and Mendelsohn decomposition technique in a first step to isolate the structurally singular 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 structurally singular subsystem is detected. The method is applied in a recursive way until all the minimal structurally singular components are localized. Leitold and Hangos (2001) consider the DAS for dynamic process models. They propose a graph-theoretical method for analysing the differential index. The method is an extension of Murota's approach (Murota, 2000), where an incidence graph is considered for each differential index.

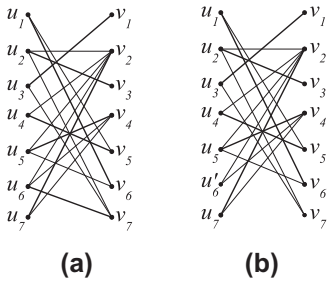

Fig. 2. Incidence graphs of DASs (5) and (5'), respectively.

In Pantelides (1988), Pantelides considers the problem of determining whether differentiation of a subset of the equations of a DAS provides further constraints to be satisfied by the initial values. A graph-theoritical algorithm is proposed to locate those subsets of the system equations which need to be differentiated. Bunus and Fritzson (2008) propose a methodology for detecting and repairing overconstrained and underconstrained DASs based on graph-theoritical approaches. In Poulsen (2001) Poulsen proposes a methodology and approach for analysing general DASs. The methodology is mainly based on structural index analysis. This uses a new and original matrix representation of the structural information of a general DAS instead of a graph representation.

3. The SAP for conditional DAS

In this section we discuss the SAP for conditional DASs. We will give a graph based model and a MIP formulation for the problem.

As explained in the introduction, the SAP for a conditional DAS consists in verifying whether the system is structurally nonsingular in any state, and if not, in finding a state in which the system is structurally singular. The SAP thus reduces to verifying whether or not the bipartite graph related to any state of the system contains a perfect matching. Fig. 3 shows the incidence bipartite graphs for systems (3) and (4). However, a difficulty that arises here is that the number of possible states of a conditional DAS may be exponential. So the approaches used so far for nonconditional DASs cannot, unfortunately, be applied for that problem. A more efficient method is thus needed for solving it. In the rest of this section we shall discuss a graph based model for the problem and purpose a MIP formulation.

Given a conditional DAS with n equations, say $C_1, \ldots, C_n$, and n variables, say $x_1, \ldots, x_n$, we consider a bipartite graph $G = (U \cup V, E)$ where $U = \{u_1, \ldots, u_n\}$ (resp. $V = \{v_1, \ldots, v_n\}$) 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 C_i, when the condition of C_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 $i = 1, \ldots, n$. Then

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

Fig. 4 shows the graph G associated with system (2).

Thus, the SAP reduces to find 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 such a subgraph as a *feasible solution* of the problem and we will refer to this problem as the *perfect matching free subgraph problem* (PMFSP). In consequence, the SAP for conditional DASs can also be presented as follows. Given a bipartite graph $G = (U \cup V, E)$ with $|U| = |V|$ and such that the set of edges incident to u_i for $i = 1, \ldots, n$ is partitioned into two sets E_i^f, E_i^t, determine whether or not there is a subgraph $G' = (U \cup V, E')$ of G containing for each $u_i \in U$ either E_i^t or E_i^f and being perfect matching free. Thus we have the following.

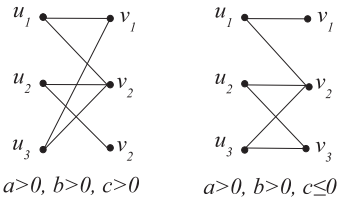

Fig. 3. Incidence graphs.

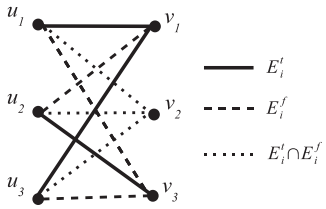

Fig. 4. Graph representing system (2).

Theorem 1. *The SAP for a conditional DAS is equivalent to the PMFSP in the associated bipartite graph.*

In Lacroix et al. (2010), it is proved that the PMFSP is NP-complete. It is shown that the PMFSP is equivalent to the stable set problem in a particular case of tripartite graphs. Which is in turn proved to be NP-complete.

Integer programming (Grötschel, Lovasz, & Schrijver, 1981) is one of the most 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 PMFSP, 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 and only if E_i^t is contained in G' and 0 if and only if E_i^f is contained in G', that is, $x(u_i) = 1$ if the condition of equation C_i is true and 0 if not.

Given $x \in \{0,1\}^U$, one needs to test whether or not the subgraph G' induced by x contains a perfect matching. If G' contains a perfect matching, say M, we then have $x(u_i) = 1$ for all i such that $M \cap E_i^t \neq \emptyset$ and $x(u_i) = 0$ for all i such that $M \cap E_i^f \neq \emptyset$. Thus x satisfies the following equation:

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

Conversely, G' does not contain any perfect matching, if, for every matching M of G,

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

In consequence, the decision problem PMFSP is equivalent to find whether there exists a solution for the following integer program (P).

$$\min 0x$$

$$(P) \quad \sum_{u_i v_j \in M \cap E_i^t} x(u_i) + \sum_{u_i v_j \in M \cap E_i^f} (1 - x(u_i)) \leqslant n - 1 \quad \text{for all } M \in \mathscr{M}, \tag{6}$$

$$0 \leqslant x(u_i) \leqslant 1, \quad \text{for all } u_i \in U, \tag{7}$$

$$x(u_i) \in \{0,1\}, \quad \text{for all } u_i \in U, \tag{8}$$

where $\mathscr{M}$ is the set of perfect matchings of G. Indeed, the PMFSP has a "yes" answer, that is the underlaying DAS is structurally nonsingular, if and only if the program above has no feasible solution. Constraints (6) will be called *matching inequalities*. Constraints (7) are the so-called *trivial inequalities* and constraints (8) are the integrality constraints.

Hence, in order to solve PMFSP, one can use integer programming tools for solving (P). One of the most powerful techniques in integer programming and combinatorial optimization is the so-called *polyhedral 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 formulation (P) is that the polyhedron given by inequalities (6)–(8) may be empty if the PMFSP has no feasible solution, that is if the DAS is structurally nonsingular. 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 the matching inequalities and give an equivalent optimization formulation whose associated polyhedron is always nonempty. Consider the following program (Q), where y is a new non-negative variable.

$$\min y$$

$$\sum_{u_i v_j \in M \cap E_i^t} x(u_i) + \sum_{u_i v_j \in M \cap E_i^f} (1 - x(u_i)) - y \leqslant n - 1, \quad \text{for all } M \in \mathscr{M}, \tag{9}$$

$$(Q) \quad 0 \leqslant x(u_i) \leqslant 1, \quad \text{for all } u_i \in U, \tag{10}$$

$$0 \leqslant y, \tag{11}$$

$$x(u_i) \in \{0,1\}, \quad \text{for all } u_i \in U. \tag{12}$$

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 structurally nonsingular for all the states, and if $y = 0$ then the state induced by x is structurally singular. Inequalities (9) will be also called *matching inequalities*. Inequality (11) expresses the fact that the new variable is non-negative and is also called a trivial inequality. Note that the approach we used to transform the decision model (P) to an optimization one is similar to that used in Phase I in linear programming for detecting a feasible basic solution. As in program (P), the number of inequalities in (Q) may be exponential. In order to solve (Q) using a branch-and-cut approach, one needs an efficient algorithm for separating inequalities (9). In the following section we devise a polynomial time separation algorithm for these inequalities.

4. Separation

The *separation problem* for inequalities (9) consists, given a solution $(x^*, y^*) \in \mathbb{R}_+^U \times \mathbb{R}_+$, in determining whether (x^*, y^*) satisfies inequalities (9), and if not in finding an inequality violated by (x^*, y^*). An algorithm which solves this problem is called a *separation algorithm*. Since inequalities (9) are in exponential number in program (Q), the separation algorithm is a key ingredient for being able to use these inequalities within a cutting plane algorithm. In what follows, we will give a polynomial time separation algorithm for these inequalities. From the equivalence between separation and optimization in combinatorial optimization (Grötschel et al., 1981), this will imply that the linear relaxation of problem (Q) can be solved in polynomial time.

Let $(x^*, y^*) \in \mathbb{R}_+^U \times \mathbb{R}_+$. With an edge $u_i v_j \in E$ associate the weight $w(u_i v_j)$ given by

$$w(u_i v_j) = \begin{cases} 1 & \text{if } u_i v_j \in E_i^t \cap E_i^f, \\ 1 - x^*(u_i) & \text{if } u_i v_j \in E_i^f, \\ x^*(u_i) & \text{if } u_i v_j \in E_i^t. \end{cases}$$

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 (9), corresponding to M, is violated. Otherwise, all the inequalities of type (9) are satisfied.

For instance, consider system (2) 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 Fig. 4 the following weights $w(u_1 v_1) = 0.7$, $w(u_1 v_3) = 0.3$, $w(u_2 v_1) = 0.6$, $w(u_2 v_3) = 0.4$, $w(u_3 v_1) = 0.3$, $w(u_3 v_3) = 0.7$, $w(u_1 v_2) = w(u_2 v_2) = w(u_3 v_3) = 1$. The maximum perfect matching with respect to w is $\{u_1 v_1, u_2 v_2, u_3 v_3\}$ with $w(M) = 2.4$. As $y^* + n - 1 = 2.2$, $w(M) > y^* + n - 1$.

Since $u_1v_1 \in E_1^t$, $u_2v_2 \in E_2^t \cap E_2^f$, $u_3v_3 \in E_3^f$, we have that the inequality

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

is violated with respect to (x^*, y^*).

Thus, the separation problem for inequalities (9) reduces to compute a maximum weight perfect matching in a bipartite graph. Moreover, this can be solved in polynomial time (Lovasz & Plummer, 1986).

5. Branch-and-Cut algorithm

Branch-and-Cut methods are very powerful techniques for solving hard integer programming and combinatorial optimization problems. These methods consist of a combination of a cutting plane technique and a Branch-and-Bound algorithm. In this section, we present a Branch-and-Cut algorithm for the SAP for conditional DAS. Our aim is to address the algorithmic applications of the model and the theoretical results presented in the previous sections. To start the optimization, we consider the following linear program given by the trivial inequalities, that is

$$\begin{aligned} &\min y \\ &0 \leqslant x(u_i) \leqslant 1, \quad \text{for all } u_i \in U, \\ &0 \leqslant y. \end{aligned}$$

An important task in the Branch-and-Cut algorithm is to determine whether or not an optimal solution of the linear relaxation of the SAP is feasible. An optimal solution $(\bar{x}, \bar{y})$ of the linear relaxation is feasible for the SAP if $\bar{x}$ is integer and $(\bar{x}, \bar{y})$ satisfies the matching inequalities. Thus verifying if $(\bar{x}, \bar{y})$ is feasible for SAP can be done in polynomial time. Note that if $\bar{x}$ is integer, then $\bar{y}$ is also integer. If an optimal solution $(\bar{x}, \bar{y})$ of the linear relaxation of the SAP is not feasible, the Branch-and-Cut algorithm generates a matching inequality that is valid for our problem and violated by $(\bar{x}, \bar{y})$. We remark that all inequalities are global (i.e. valid in all the Branch-and-Cut tree). Our separation algorithm is applied on the graph G weighted by the current LP-solution $(\bar{x}, \bar{y})$.

To store the generated inequalities, we create a pool whose size increases dynamically. In each iteration, the new inequalities are added to the pool. The generated inequalities are removed from the current LP when they are not active. We first separate inequalities from the pool. If all the inequalities in the pool are satisfied by the current LP-solution, then we separate the matching inequalities.

The Branch-and-Cut algorithm has been implemented in C++ using ABACUS library (Elf, Gutwenger, Jünger, & Rinaldi, 2001) to manage the Branch-and-Cut tree and CPLEX 11.0 as LP-solver. To separate inequalities (9), we use Galil, Micali, and Gabow algorithm (Galil, Micali, & Gabow, 1986) for solving the maximum weight perfect matching. This algorithm runs in $O(nmlog(n))$ time where n is the number of nodes and m the number of edges of the graph. Actually, we use the version implemented in Lemon Graph Library (Jüttner, Dezsö, & Kovács, 2010).

The algorithm was tested on a Pentium core 2 duo 2.66 GHz with 2 Gb RAM. We fixed the CPU time limit to 1 h. Results are presented here for randomly generated instances and realistic instances. For randomly instances, tests were performed for systems with up to $n = 70$ equations. (Recall that the corresponding bipartite graphs have $2n$ nodes). The systems are considered in such a way that each equation has between $k-1$ and $k+1$ variables where k is fixed. Actually, the systems met in practice, as those that will be tested here, respect that restriction. In consequence, in the incidence graphs of the instances, every node has a degree between $k-1$ and $k+1$. Our tests were performed for k between 5 and 25. Five instances were tested for each problem and we will provide the average results.

Table 1
Randomly generated instances.

n	k	o/p	CPU	No	Match	Gap
20	5	5/5	0	1	20.4	0
20	10	5/5	3.6	18.2	625.8	0.66
20	15	5/5	3.4	7	444.2	0.67
20	20	5/5	2.6	15.4	359	0.65
25	5	5/5	0.2	1.4	13.6	0
25	10	5/5	0	1	1.8	0
25	15	5/5	13.8	18.2	1201.4	0.29
25	20	5/5	29.2	73.4	2189	0.65
30	5	5/5	8.4	6.6	1079.6	0
30	10	4/5	1340	129.4	98373.8	0.54
30	15	4/5	2071.8	99.8	130,632	0.67
30	20	3/5	1826.6	55.8	99637.4	0.63
30	25	2/5	2884.6	46.6	142051.2	0.33
35	5	5/5	4.8	1	720.75	0
35	10	2/5	2643.4	12.6	134,868	0.44
35	15	0/5	3600	1	162749.2	–
40	5	5/5	14	1	932	0
40	7	4/5	883.8	1	36967.8	0
40	9	3/5	1694.2	1	61363.8	0
45	5	5/5	55	1.4	2112.6	0
45	7	3/5	2099.4	1	57603.2	0
45	9	0/5	3600	1	100906.8	–
50	5	5/5	62.8	1	2079.4	0
50	7	5/5	1312.8	1	28117.4	0
50	9	1/5	3068.4	1	61452.6	0
55	5	5/5	628.6	1	11635.2	0
55	7	0/5	3600	1	56792.6	–
55	9	0/5	3600	1	55639.2	–
60	5	5/5	558.8	1	8736.8	0
60	7	2/5	3003.8	1	35038.6	0
60	9	0/5	3600	1	42140.6	–
65	5	4/5	1520.4	1	34,944	0
65	7	0/5	3600	1	34064.6	–
65	9	0/5	3600	1	35083.5	–
70	5	2/5	1819.5	1	25,240	0
70	7	0/5	3600	1	29590.5	–
70	9	0/5	3600	1	29,768	–

Table 2
Realistic instances.

n	k	CPU	No	Match	Gap
51	3	0	1	1	0
75	3	3	63	32	1
81	3	0	1	1	0
111	3	7	510	256	1
135	3	33	2047	1024	1
159	3	227	8190	4096	1
183	3	2399	32,766	16,384	1
255	3	3600	26,945	13,478	–

The results are given in Tables 1 and 2. Table 1 reports the average results obtained for the randomly generated problems, while Table 2 presents the results for the realistic ones.

The entries in the tables are:

- n : the number of equations
- k : the integer indicating that the number of variables in each equation is between $k-1$ and $k+1$
- o/p : the number of problems solved to optimality over the number of instances tested
- CPU : the total CPU time in second
- No : the number of generated nodes in the Branch-and-Cut tree
- Match : the number of generated matching inequalities
- Gap : the gap between the optimal value and the lower bound obtained at the root node of the Branch-and-Cut tree

From Table 1, summarizing the results for the random instances, it appears that the CPU time strongly depends on the integer k. For instance, for the problems with $n = 35$ equations and k equal to 5, all the instances have been solved to optimality. Whereas for the same instances with k greater than or equal to 15, none of the instances have been solved in the time limit. This may be explained by the fact that when k is high, the associated graph is quite dense, and hence contains more perfect matchings. In consequence, verifying if there is a configuration of the system which does not yield to a perfect matching, may need a lot of time. However when the graph is sparse, this can be done much faster. We also remark that a significant number of matching inequalities have been generated for most of the instances with 30 equations and more.

The instances of Table 2 correspond to realistic systems having between 51 and 255 equations. These systems were generated by composition of small electrical circuits. These instances have been much easier to solve. In fact, as it appears from the table, all the instances have been solved to optimality except the system containing 255 equations. We also remark that, as for the random instances, a big number of matching inequalities have been generated for all the large instances. In addition, we observe that, in contrast of the random instances, the algorithm generates much more nodes in the Branch-and-Cut tree.

Finally, let us mention that we have tested an implicit enumeration algorithm for solving the SAP but we could not solve the problem for instances with more than 35 equations. This shows the efficiency of the Branch-and-Cut algorithm we propose.

6. Extension: DASs with embedded conditions

The approach developed above can be extended for integrating embedded conditions. DASs with embedded conditions are more complex to handle. In this section we discuss this generalization.

Each embedded conditional equation can generate several nonconditional equations. If an embedded conditional equation C_d generates q equations with respect to a condition d, then it can be expanded to q conditional equations $C_1, \ldots, C_q$ such that each equation C_r may generate the rth equation of C_d according to condition d whether it is true or false. For example, consider the following embedded DAS with three equations and four variables.

$$
\begin{aligned}
&C_1: \text{if } a>0\\
&\quad \text{then}\\
&\quad\quad \text{if } b>0\\
&\quad\quad\quad \text{then } 0=\dot{x}_2,\\
&\quad\quad\quad \text{else } 0=3x_1,\\
&\quad \text{else}\\
&\quad\quad \text{if } c>0\\
&\quad\quad\quad \text{then } 0=3x_3+\dot{x}_2,\\
&\quad\quad\quad \text{else } 0=x_4,\\
&C_2: \text{if } d>0\\
&\quad \text{then}\\
&\quad\quad \text{if } e>0\\
&\quad\quad\quad \text{then } 0=4x_2+\dot{x}_1,\\
&\quad\quad\quad \text{else } 0=x_3,\\
&\quad\quad \text{if } f>0\\
&\quad\quad\quad \text{then } 0=x_1,\\
&\quad\quad\quad \text{else } 0=x_4+x_3,\\
&\quad \text{else } 0=x_1+\dot{x}_4+1,\\
&\quad\quad 0=x_4+\dot{x}_2+1,\\
&C_3: \text{if } a>0\\
&\quad \text{then } 0=6\dot{x}_1+2,\\
&\quad \text{else } 0=3\dot{x}_2+x_3+3+x_4.
\end{aligned}
\tag{13}
$$

Observe that C_2 can be decomposed into two sub-equations:

$$
\begin{aligned}
&C_2': \text{if } d>0\\
&\quad \text{then}\\
&\quad\quad \text{if } e>0\\
&\quad\quad\quad \text{then } 0=4x_2+\dot{x}_1,\\
&\quad\quad\quad \text{else } 0=x_3,\\
&\quad \text{else } 0=x_1+\dot{x}_4+1,\\
&C_2'': \text{if } d>0\\
&\quad \text{then}\\
&\quad\quad \text{if } f>0\\
&\quad\quad\quad \text{then } 0=x_1,\\
&\quad\quad\quad \text{else } 0=x_4+x_3,\\
&\quad \text{else } 0=x_4+\dot{x}_2+1.
\end{aligned}
$$

Therefore, system (13) can be expressed by (C_1, C_2', C_2'', C_3) Thus any conditional equation of a DAS with embedded conditions may be decomposed into smaller *conditional sub-equations* or *embedded conditional sub-equations*. In the sequel we consider systems that are minimal, that is to say in which the conditional equations cannot be decomposed anymore. The system expressed by (C_1, C_2', C_2'', C_3) is minimal. As for the non-embedded DAS, we suppose that each conditional equation generates one simple equation for each combination of conditions. We also suppose, without loss of generality, that every embedded conditional equation, may generate exactly one nonconditional 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 assigned to an embedded equation yields only one nonconditional equation.

For instance, in system (13), the equation $0 = \dot{x}_2$ in C_1 is the result of the combination $\{a > 0, b > 0\}$ and $0 = x_4 + x_3$ in C_2'' is the result of the combination $\{d > 0, f \leqslant 0\}$. Let $C_1, \ldots, C_m$ be the possible combinations of conditions. Remark that the number of combinations C_k is polynomial in the size of the system. For system (13), we have the following combinations: $C_1 = \{a > 0,\ b > 0\}$, $C_2 = \{a > 0, b \leqslant 0\}$, $C_3 = \{a \leqslant 0, c > 0\}$, $C_4 = \{a \leqslant 0, c \leqslant 0\}$, $C_5 = \{d > 0, e > 0\}$, $C_6 = \{d > 0,\ e \leqslant 0\}$, $C_7 = \{d > 0,\ f > 0\}$, $C_8 = \{d > 0,\ f \leqslant 0\}$, $C_9 = \{d \leqslant 0\}$, $C_{10} = \{a > 0\}$ and $C_{11} = \{a \leqslant 0\}$.

For $k = 1, \ldots, m$, let R_k be the set of combinations of conditions that are implied by C_k, that is the combinations that are satisfied if C_k so is. For instance, for the example above, $C_{10} = \{a > 0\}$ is implied by $C_1 = \{a > 0, b > 0\}$. For $k = 1, \ldots, m$, let T_k be the set of combinations C_l which are incompatible with C_k, that is C_k and C_l cannot occur at the same time. For the example above C_2 and C_4 are incompatible. Note that any two combinations C_k and C_l related to the same equation are compatible.

Consider the bipartite graph $G = (U \cup V, E)$ where $U = \{u_1, \ldots, u_n\}$ corresponds to the minimal conditional equations of the system, $V = \{v_1, \ldots, v_n\}$ corresponds to the variables, and we consider an edge $u_i v_i$ 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 $k = 1, \ldots, m$. Thus

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

Observe that an edge of E may appear in more than one set E_k.

For system (13), the edge sets E_k, generated by combination $C_1, \ldots, C_{11}$, are $E_1 = \{u_1v_2\}$, $E_2 = \{u_1v_1\}$, $E_3 = \{u_1v_3, u_1v_2\}$, $E_4 = \{u_1v_4\}$, $E_5 = \{u_2v_2, u_2v_1\}$, $E_6 = \{u_2v_3\}$, $E_7 = \{u_3v_1\}$, $E_8 = \{u_3v_4, u_3v_3\}$, $E_9 = \{u_2v_1,$

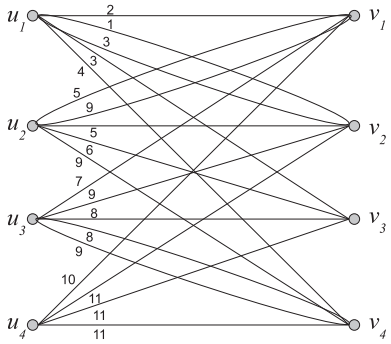

Fig. 5. Graph G.

$u_2v_4, u_3v_4, u_3v_2\}$, $E_{10} = \{u_4v_1\}$ and $E_{11} = \{u_4v_2, u_4v_3, u_4v_4\}$. The incidence graph is given in Fig. 5. Note that nodes u_2, u_3 correspond to the conditional equations C'_2, C''_2. The number on each edge corresponds to the set E_k to which the edge belongs.

Thus the DAS is structurally singular 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 $E_k, E_l \in \mathscr{F}$, then C_k and C_l are compatible, and
(b) if $E_k \in \mathscr{F}$ and $C_l \in R_k$, then $E_l \in \mathscr{F}$.

One can easily verify for graph G of Fig. 5 that the sets E_1, E_6, E_7, E_{10} constitute a feasible family that covers the nodes $u_1, \ldots, u_4$ and verifies the conditions (a) and (b) above. However the subgraph induced by these sets does not contain a perfect matching. In consequence, the family of conditions C_1, C_6, C_7, C_{10} induces a structurally singular system. Given a DAS with embedded conditions we will call *state* of the system any set of combinations C_k covering all the equations of the system and whose edge sets E_k 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 structurally singular or to show that such a state does not exist, that is the system is structurally nonsingular 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 subgraph contains such a matching.

The generalized PMFSP (GPMFSP) consists in finding a perfect matching free subgraph satisfying conditions (a) and (b) and covering U or to show that such a subgraph does not exist. In what follows we will propose an integer programming based model for this problem.

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

$$x_k + x_l \leqslant 1, \quad \text{for all } l \in T_k, \quad \text{for } k = 1, \ldots, m,$$
$$x_k \leqslant x_l, \quad \text{for all } l \in R_k, \quad \text{for } k = 1, \ldots, m.$$

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

$$\sum_{k=1}^{m} |E_k \cap M| x_k = n.$$

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

$$\sum_{k=1}^{m} |E_k \cap M| x_k \leqslant n - 1.$$

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

$$\max \ y \tag{14}$$
$$\sum_{k=1}^{m} |E_k \cap M| x_k \leqslant n - 1, \quad \text{for all } M \in \mathscr{M}, \tag{15}$$
$$x_k + x_l \leqslant 1, \quad \text{for all } l \in T_k, \quad \text{for } k = 1, \ldots, m, \tag{16}$$
$$x_l \leqslant x_k, \quad \text{for all } l \in R_k, \quad \text{for } k = 1, \ldots, m \tag{17}$$
$$\sum_{\substack{k=1,\ldots,m \\ E_k \cap \delta(u_i) \neq \emptyset}} x_k \geqslant y, \quad \text{for all } u_i \in U, \tag{18}$$
$$0 \leqslant x_k \leqslant 1, \quad \text{for all } E_k \subset E, \tag{19}$$
$$0 \leqslant y, \tag{20}$$
$$x_k \in \{0, 1\}, \quad \text{for all } E_k \subset E. \tag{21}$$

where $\mathscr{M}$ is the set of perfect matching in G. We claim that the SAP is equivalent to program $(\widehat{P})$. Actually, we have that GPMFSP has a solution if and only if $(\widehat{P})$ has an optimal solution (x,y) with $y = 1$. In fact, first remark that any two combinations of conditions C_k, C_l related to the same equation are incompatible, and then the corresponding edge sets E_k, E_l cannot be considered together. Then any feasible solution satisfies

$$\sum_{\substack{k=1,\ldots,m \\ E_k \cap \delta(u_i) \neq \emptyset}} x_k \leqslant 1, \quad \text{for all } u_i \in U.$$

This implies that $y \leqslant 1$. Moreover, y is integer. Otherwise, since the left hand side of (18) is integer, the solution $(x, y + \epsilon)$ with $\epsilon > 0$ is also feasible, contradicting the optimality of (x,y). Now it is clear, by constraints (15)–(17), (19)–(21) that every optimal solution (x,y) of $(\widehat{P})$ induces a subgraph which does not contain a perfect matching. Moreover, the edge sets E_k such that $x_k = 1$ satisfy conditions (a) and (b). If $y = 1$, then the subgraph induced by x also covers all the nodes of U thanks to constraints (18), which implies that the subgraph induced by x is a solution of GPMFSP. Suppose now that $y = 0$. Since the objective function of $(\widehat{P})$ maximizes y, there must exist $i \in \{1, \ldots, n\}$ such that $\sum_{\substack{k=1,\ldots,m \\ E_k \cap \delta(u_i) \neq \emptyset}} x_k = 0$. Otherwise, $(x,1)$ would be also a solution of $(\widehat{P})$ which contradicts the optimality of (x,y). But this implies that there does not exist a perfect matching free subgraph covering the nodes of U and satisfying conditions (a) and (b). In consequence, the GPMFSP has no solution.

Due to the equivalence between GPMFSP and SAP for embedded DASs, it follows that an embedded DAS is structurally nonsingular if and only if $(\widehat{P})$ has an optimal solution with $y = 0$.

Clearly, constraints (16)–(20) can be separated in polynomial time (by enumeration). For constraints (15), the separation problem can be reduced to the maximum weight matching problem in a bipartite graph.

7. Concluding remarks

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 non-embedded cases. We have shown that the linear relaxations of these models can be solved in polynomial time. Based on the formulation for the non-embedded version, we have developed a Branch-and-Cut algorithm for solving the problem in this case and presented some computational results. These show that the difficulty of the problem strongly depends on the density of the incidence graph. They also show that the linear relaxation of the problem is not so tight. Further valid inequalities are needed to strengthen the linear relaxation. For this a deep investigation of the associated polytope would be necessary. This is one of the directions of our futur research. A second direction is to develop a Branch-and-Cut algorithm for the embed-

ded case. Here the problem is much harder. However, our polyhedral investigation for the non-embedded problem would be very helpful for the embedded one.

Acknowledgments

We would like to thank the two anonymous referees for their very constructive comments that permitted us to much improve the presentation of the papers. We would also like to thank Sébastien Furic, Bruno Lacabanne and El Djillali Talbi 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

Bujakiewicz, P. (1994). Maximum weighted matching for high index differential algebraic equations. Doctor's dissertation, Delft University of Technology.

Bunus, P., & Fritzson, P. (2008). Automated static analysis of equation-based components. *Simulation, 80*, 321–345.

Duff, I. S., & Gear, C. W. (1986). Computing the structural index. *SIAM Journal on Algebraic and Discrete Methods*, 594–603.

Dulmage, A. L., & Mendelsohn, N. S. (1963). Coverings of bipartite graphs. *Canadian Journal of Mathematics*, 517–534.

Edmonds, J. (1965). Maximum matching and a polyhedron with 0,1-vertices. *Journal of Research of the National Bureau of Standards, 69B*, 125–130.

Elf, M., Gutwenger, C., Jünger, M., & Rinaldi, G. (2001). *Branch-and-cut algorithms for combinatorial optimization and their implementation in ABACUS. Computational combinatorial optimization*. Springer. pp. 157–222.

Fritzson, P. (2003). *Principles of object-oriented modeling and simulation with Modelica 2.1*. Wiley-Interscience.

Galil, Z., Micali, S., & Gabow, H. N. (1986). An O(EV log V) algorithm for finding a maximal weighted matching in general graphs. *SIAM Journal on Computing, 15*, 120–130.

Grötschel, M., Lovasz, L., & Schrijver, A. (1981). The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 169–197.

Jian-Wan, D., Li-Ping, C., Fan-Li, Z., Yi-Zhong, W., & Guo-Biao, W. (2006). An analyzer for declarative equation based models. *Modelica*, 349–357.

Lacroix, M., Mahjoub, A. R., & Martin, S. (2010). Structural analysis for differential-algebraic systems: Complexity, formulation and facets. In *Proceeding ISCO 2010, electronic notes in discrete mathematics* (Vol. 36, pp. 1073–1080).

Leitold, A., & Hangos, K. M. (2001). Structural solvability analysis of dynamic process models. *Computers and Chemical Engineering, 25*, 1633–1646.

Jüttner, A., Dezsö, B., & Kovács, P. (2010). Lemon graph library. <https://lemon.cs.elte.hu/trac/lemon>.

Lovasz, L., & Plummer, M. D. (1986). *Matching theory*. North-Holland.

Murota, K. (2000). *Matrices and matroids for systems analysis*. Springer-Verlag.

Nilsson, H. (2008). Type-based structural analysis for modular systems of equations. In *Proceedings of the 2nd international workshop on equation-based object-oriented languages and tools* (pp. 71–81).

Pantelides, C. C. (1988). The consistent initialization of differential-algebraic systems. *SIAM Journal of Scientific and Statistical Computing*, 213–231.

Poulsen, M. Z. (2001). Structural analysis of DAEs. Ph.D. thesis, Informatics and Mathematical Modelling Technical University of Denmark.

Reibig, G., & Feldmann, U. (2002). A simple and general method for detecting structural inconsistencies in large electrical networks. *Circuits and Systems I: Fundamental Theory and Applications*, 237–240.