Minimal Structurally Overdetermined sets for residual generation: A comparison of alternative approaches

Armengol, J. [*] **Bregon, A.** [**] **Escobet, T.** [***] **Gelso, E.** [*]
Krysander, M. [****] **Nyberg, M.** [****] **Olive, X.** [†] **Pulido, B.** [**]
Travé-Massuyès, L. [‡]

[*] *IIiA, Universitat de Girona, Spain; ({joaquim.armengol,esteban.gelso}@udg.edu)*
[**] *GSI, Universidad de Valladolid, Spain; ({anibal,belar}@infor.uva.es)*
[***] *SAC, Universitat Politècnica de Catalunya, Spain; (teresa.escobet@upc.edu)*
[****] *Dept. of Electrical Engineering, Linköpings universitet, Sweden; ({matkr,matny}@isy.liu.se)*
[†] *Thalès Alenia Space France; (Xavier.Olive@thalesaleniaspace.com)*
[‡] *LAAS-CNRS, Université de Toulouse, France; (louise@laas.fr)*

Abstract: The issue of residual generation using structural analysis has been studied by several authors. Structural analysis does not permit to generate the analytical expressions of residuals since the model of the system is abstracted by its structure. However, it determines the set of constraints from which residuals can be generated and it provides the computation sequence to be used. This paper presents and compares four recently proposed algorithms that solve this problem.

1. INTRODUCTION

Fault detection and diagnosis aims at exhibiting and identifying faulty situations for partially observed systems. In the continuous model based approaches, the subsets of model equations/constraints that underly discrepancies play a fundamental role. In FDI approaches these subsets are the source for residuals, which can be computed following the parity space approach (building testable relations) or the observer approach Blanke and Lorentzen [2006]. Testable relations are computed off-line and can be checked on-line against the observations as they represent redundancies. This is why they are also called Analytical Redundancy Relations (ARR) Staroswiecki [2002]. In the DX logical diagnosis theory Reiter [1987], the supports of these relations, i.e. the behavioral hypotheses, are known as conflicts. It has been shown that ARR supports correspond to potential conflicts Cordier et al. [2004], Nyberg and Krysander [2003], Pulido and Alonso-González [2004], i.e. generating ARRs and their supports can be seen as the compilation of potential conflicts.

Thus, these subsets of equations can be characterized as the overdetermined (or over-constrained) sets, i.e. those that contain more equations than unknown variables. Among them, the minimal overdetermined sets have been shown to be the necessary and sufficient ones to construct a residual.

Since finding minimal overdetermined sets is the core of many approaches to finding testable relations, or the complete set of minimal ARRs, it is important to analyze and compare different solutions to this problem. This work aims at presenting and comparing four alternative structural approaches that determine the *Minimal Structurally Overdetermined (MSO) sets* Krysander et al. [2008], Gelso et al. [2008], or the structure of the minimal ARRs, also called *Structural ARRs (SARRs)* Pulido and Alonso-González [2004], Travé-Massuyès et al. [2006]. The properties of the algorithms analyzed and compared are the correctness of the solution provided, and the computational complexity. One further contribution is a unifying framework in which the four algorithms are presented and compared.

The paper is organized as follows. Section 2 defines a set of terms used through the paper. Then, section 3 compares the four algorithms principles from a theoretical point of view. Afterwards, section 4 presents a brief description of the four algorithms, and section 5 analyses their computational complexity. Next section introduces the CHEM benchmark, which is used as a case study, and presents the results of the four algorithms. Section 7 deals with related work and section 8 concludes the paper.

2. MAIN CONCEPTS

Let the *system description* SD consist of a set of n equations involving a set of variables, and associated to each equation, a *support*. The set of variables is partitioned into a set Z of n_Z known variables and a set X of n_X unknown variables. The known variables Z are typically system inputs and outputs, and the unknown variables X are system internal variables, unknown inputs, or disturbances. We do refer to the vector of known variables z and the vector of unknown variables x. The equations are assumed to be differential or algebraic equations in z and x. The supports are described later on.

We consider a *model*, denoted $M(z,x)$ or M for short, to be any set of equations in knowns z and unknowns x. The total set of equations included in the system description SD is denoted M_{SD}.

MSO sets can be defined as follows:

Definition 1. (Structurally Overdetermined). A set $M \subseteq M_{SD}$ of equations is *structurally overdetermined* (SO) if M has more equations than unknowns.

Definition 2. (Minimal Structurally Overdetermined). A *Minimal Structurally Overdetermined (MSO) set* is a structurally overdetermined set whose no proper subset is structurally overdetermined.

We say that a model $M(z, x)$ is consistent with a given trajectory of z, or for short, consistent with z, if there is a trajectory of x such that the equations $M(z, x)$ are fulfilled.

Definition 3. (ARR for $M(z, x)$). Let $M(z, x)$ be a model, then an equation $r(z, \dot{z}, \ddot{z}, \ldots) = 0$ is an *ARR for* $M(z, x)$ if for each z consistent with the model $M(z, x)$, the equation is fulfilled.

Definition 4. (Residual Generator for $M(z, x)$). A system taking a subset of the variables z as input, and generating a scalar signal r as output, is a *residual generator for the model* $M(z, x)$, if for all z consistent with $M(z, x)$, it holds that $\lim_{t \to \infty} r(t) = 0$.

To each equation $e \in M_{SD}$ there is a corresponding *support* denoted by supp(e). If the support of an equation is valid, then the equation is assumed to be valid. We can write this as

$$\text{supp}(e) \to e$$

In this work, we represent the support by the predicate $OK(C)$, meaning that the equation is valid if component C is fault free. If there are fault modes, the same predicate type of notation can be used.

Definition 5. (Model Support). The support of a model M is defined as

$$\text{supp}(M) = \wedge_{e \in M} \text{supp}(e)$$

Definition 6. (Residual Support). If $r(t)$ is a residual computed by a residual generator for the model $M(z, x) \subseteq M_{SD}(z, x)$, then

$$\text{supp}(r(t)) = \text{supp}(M)$$

is the *residual support* of $r(t)$.

The support of an ARR can be defined in a similar way. Given the above definitions, the support of a residual/ARR is uniquely defined from the set of equations that are involved. This set of equations is called the *equation support*. If not ambiguous, *support* and the notation $supp(.)$ are indifferently used for support or equation support. These definitions also hold in a structural framework.

The structure of a model M can be represented by the bipartite graph $G(M, X \cup Z)$ (or equivalently its biadjacency matrix). There is an edge between an equation $e_i \in M$ and a variable $v_i \in X \cup Z$ if v_i, or any time derivative of v_i, is involved in e_i [1]. A 0 in position (e_i, v_j) indicates that equation e_i does not contain variable v_j, and a 1 that equation e_i contains variable v_j. The vector of components (e_i, v_j), $j = 1, \ldots, n_X + n_Z$, is referred as the structure of equation e_i, also called a *structural equation*. An SARR is defined as the structure of an ARR.

Model equations can be interpreted in a causal way and may have several *causal interpretations*, each pointing at one of the involved variables that can be always computed given the values of all the other variables involved. This information, which is used by two of the presented algorithms, can be accounted for in the structural model.

A minimal SARR is obtained from an MSO set by eliminating the unknown variables and combining the equations structure accordingly. This requires the equations of the MSO set to have consistent causal interpretations [2]. Hence, a minimal SARR corresponds to a MSO set but the inverse is not true.
Let us note that if several causal interpretations are consistent for the equations of a given MSO set, they are considered to produce the same SARR because the resulting ARRs are semantically equivalent, i.e. they just differ syntactically.

[1] Other structural representations could equivalently be used for dealing with differential forms Krysander et al. [2008].

3. COMPARISON OF THE ALGORITHMS PRINCIPLES

The four algorithms presented in the next section follow a structural approach familiar to the FDI community Staroswiecki [2002]. They take as input a structural model. The so-called MSO-Algorithm and CBMSOs-Algorithm only use structural information and they output the MSO sets. The SARR-Algorithm and the PCC-Algorithm make use of additional causal information and they output one causal interpretation, i.e. a minimal SARR, or all the consistent causal interpretations for each MSOs, respectively.

Some characteristic features of the four algorithms are outlined below:

- The MSO-Algorithm determines MSOs, proceeding by eliminating equations from the original set M (more precisely the structurally overdeterminated part of it) until the structural redundancy (difference between number of equations and number of unknown variables) is 1.
- The CBMSOs-Algorithm determines MSOs by first determining a complete matching between equations and unknown variables, leading to a first set of MSOs. These are “combined” in a second step to get the additional MSOs.
- The SARR-Algorithm determines the set of minimal SARRs by successive elimination of the unknown variables. The subset of equations that support the eliminations leading to a SARR are MSOs.
- The PCC-Algorithm first determines MSOs, then searches for all the causally consistent interpretations for each of them. For determining MSOs, it performs successive unknown variable eliminations in a depth-first search manner. Causal interpretations are propagated in the same way for each MSO set. The relation with the SARR-Algorithm is that if a MSO set has several causal interpretations, all lead to semantically equivalent causal structural equations, hence to one SARR.

4. ALGORITHMS

4.1 MSO-Algorithm

We describe the basic version of the MSO-algorithm from Krysander et al. [2008]. The algorithm is based on a top-down approach in the sense that we start with the entire model and then reduce the size of the model step by step until a MSO set remains. The algorithm computes the set of all MSO sets contained in the model.

To present the algorithm we first need the notions of *structurally overdetermined part* and *structural redundancy*. Let $X(M) \subseteq x$ be the subset of unknowns connected to at least

[2] Equations in a set are said to have consistent causal interpretations if the equations can be chained in a pairwise causally consistent way, i.e two equations e_i and e_j linked (through a common variable v) in the chain must be such that e_i can be used to compute v and e_j can be used to compute another variable.

one equation in M. Let $^+$ be a function on and into the power set of M_{SD} such that $M^+ \subseteq M$ is the set of equations $e \in M$ such that for any maximal matching in $G(M, X(M))$ there exists an alternating path in $G(M, X(M))$ between at least one free equation node and e. The set M^+ is uniquely defined and will be called the structurally overdetermined part of M. This part is equal to the vertical tail of the Dulmage-Mendelsohn canonical decomposition Dulmage and Mendelsohn [1958] of $G(M, X(M))$ and can be computed by the command `dmperm` in Matlab®. Next we will quantify redundancy based on the structural description of a model. Given a set of equations M, the structural redundancy $\varphi(M)$ is defined by

$$\varphi(M) = |M^+| - |X(M^+)| \tag{1}$$

With these definitions, the MSO-algorithm can be stated as in Algorithm 1.

Algorithm 1:

```
1  function $MSOs := \text{FindMSO}(M)$ ;
2  if $\varphi(M) = 1$ then
3      $MSOs := \{M\}$ ;
4  else
5      $MSOs := \varnothing$;
6      foreach equation $e$ in $M$ do
7          $M' := (M \setminus \{e\})^+$;
8          $MSOs := MSOs \cup \text{FindMSO}(M')$;
9      end
10 end
```

4.2 *Combination of Basic MSOs (CBMSOs) algorithm*

CBMSOs-Algorithm, presented in Gelso et al. [2008], finds the set of all MSO sets contained in the model. The input of this algorithm is the MSO sets obtained from one complete matching on the unknown variables by using, for instance, the so called ranking algorithm or constraint propagation algorithm Blanke and Lorentzen [2006]. This first set of MSOs will be called Basic MSOs, and is obtained from unmatched constraints by backtracking unknown variables of them through constraints to which they were matched.

The basic MSO sets are combined in order to get more MSOs using Algorithm 2. A structurally overdetermined set can be obtained from the elimination of at least one shared equation from the set of equations of two MSO sets, and this operation is called a combination. The combinations of MSOs can be minimal or not. This algorithm finds only the minimal ones.

Algorithm 2: Algorithm to find all MSOs

```
Input: Complete matching
1  $MSO_1 \leftarrow$ Basic MSOs;
2  $i = 1$;
3  while $i <$ number of $MSO_1$, or, $MSO_i$ is not empty do
4      $MSO_{i+1} := \text{Combine}(MSO_i, MSO_1)$; set i=i+1;
5  end
6  $MSO := (MSO_1 \dots MSO_i)$;
Output: Complete set of MSOs
```

Function 'Combine', which is presented as Algorithm 3, leads to obtain the new MSOs after combining two MSO sets. Step four in the algorithm is very important. It can be tackled in a brute-force way, which can result in a combinatorial explosion. This method avoids this by using the rank information provided by ranking algorithm applied to the biadjacency matrix of $G(M_{SD}, X)$. Then it removes one shared constraint at a time, and the corresponding matched constraints are used only to calculate the unknown variables of the shared constraint.

Algorithm 3: Combine function

```
1  function $MSO_{ab} = \text{Combine}(MSO_a, MSO_b)$
2  foreach set $S_a$ in $MSO_a$ do
3      foreach set $S_b$ in $MSO_b$ do
4          if shared constraints set of $S_a$ and $S_b$ is not void, and, $S_a$ and $S_b$ do
           not share the same unmatched constraint then
5              Remove one or more shared constraints;
6              Check if the new structurally overdetermined set is minimal;
7              if it is minimal then
8                  add to $MSO_{ab}$;
9              end
10         end
11     end
12 end
Output: $MSO_{ab}$
```

4.3 *SARR-Algorithm*

This algorithm is a variant of the method presented in Travé-Massuyès et al. [2006], which was devised for diagnosability assessment. Starting from a structural model augmented by causal information, the idea of the algorithm is to proceed to successive elimination of unknown variables on the structural model equations to obtain the set of minimal SARRs. The equation supports of the resulting SARRs are minimal MSO sets. The main steps to generate the SARRs and the associated MSOs are provided by Algorithm 4 [3].

The input structural model (including causal information) is first processed by function *build_cse* into a table in which every line corresponds to a possible causal interpretation of a structural equation of the model, i.e. a *causal structural equation* noted cse_i. A support field, noted supp(cse_i), is added to the structure field, struct(cse_i) given by the biadjacency matrix:

- the entry (cse_i, e_j) of the support field is 1 if $e_j \in$ supp(cse_i), and 0 otherwise.
- the entry (cse_i, v_j) of the structure field, noted $s(i, j)$, is non zero if variable v_j is involved in cse_i or one of its ascendants:
  - · it is marked # if it has been eliminated;
  - · it is marked $\otimes$ if v_j is the computed variable;
  - · it is marked $\times$ otherwise.

The elimination of one variable v_j between cse_p and cse_q leads to computing a new *cse* (function *build_new_cse* in Algorithm 4). This elimination is allowed when the *cse*s have consistent causalities, support and structure.

Consistent causality is captured by the condition $Cond_{causality}$, which equals 1 when $s(p, j) = \otimes$ and $s(q, j) = \times$ or vice versa.

Combining an already combined equation with one of its own ascendant relations is not allowed. The combination of cse_p and cse_q is allowed if $Cond_{support} = 1$, i.e. |supp(cse_p) $\cup$ supp(cse_q)| = $V_{el} + 1$, where V_{el} is equal to the number of eliminated variables in cse_p and cse_q, i.e. number of $\#$.

[3] The original algorithm Travé-Massuyès et al. [2006] accounts for validity conditions associated to structural equations but the present paper does not include this option, which does not exist in the other three algorithms.

Reintroducing a variable that has already been eliminated or eliminating the same variable through different paths is not allowed neither. This is captured by $Cond_{structure}$ which is 1 if $s(cse_p, v_\beta) \in \{\times, \otimes, \} \wedge s(cse_q, v_\beta) = \#$ or $s(cse_q, v_\beta) \in \{\times, \otimes, \} \wedge s(cse_p, v_\beta) = \#$.

Despite the above conditions, the algorithm cannot avoid to obtain semantically identical (noted $\equiv$) combined structural equations, i.e. differing only by their causal interpretation, from different elimination paths. Every potential new structural equation is hence checked against the existing ones. Final combined equations, i.e. with all unknown variables eliminated, correspond to SO sets, but they may not be MSO. Non minimal ones must hence be discarded.

Algorithm 4: Find SARRs and Associated MSOs through Variable Elimination

```
function $SARR$=find_SARR ($M(x, z), X, Z$)
$f = 0$; $cse_{i=1,\ldots,h}$ = build_cse($M, x, z$);
foreach $x_i \in X$ do
    Define the set $J = \{j / x_i \in struct(cse_j)\}$ ;
    if card $J > 1$ then
        foreach $(p, q) \in J^2, p \neq q$ do
            if $Cond_{support} = 1 \wedge Cond_{structure} = 1 \wedge$
            $Cond_{causality} = 1$ then
                $new\_cse$ := build_new_cse;
                if $new\_cse \not\equiv cse_i \forall i \leq h$ then
                    if $s(new\_cse, x_i) = \# \forall x_i \in X(new\_cse)$ then
                        $SARR_{f+1} = new\_cse$;
                        $MSO_{f+1} = supp(SARR_{f+1})$; $f = f + 1$
                    end
                    $cse_{h+1} = new\_cse$; $h = h + 1$
                end
            end
        end
    end
end
```

4.4 *Possible Conflict Computation algorithm (PCC-Algorithm)*

The following algorithms summarize the Possible Conflict Computation, PCC, approach from Pulido and Alonso-González [2004]. PCC-algorithm is based on DX concepts for dependency-precompilation, following GDE concepts. The process is carried out in two steps by means of a bottom up depth search.

First step, described in Algorithm 5, iterates on each equation e_i in M, searching recursively for every MSO set containing e_i throughout Algorithm 6. This one recursively removes in each step one remaining *unknown* variable in the possible MSO, adding one new equation sharing that variable.

Algorithm 5: Step 1.1: Find every strictly overdetermined subsystems in M for each equation e in M

```
function $SMSO$=find_every_possible_mso ($M$)
foreach equation $e$ in $M$ do
    find_possible_mso($M \setminus$ {e}, {e}, $e_{unknowns}$, $SMSO$);
end
```

The search for causally consistent interpretations in each MSO proceeds in a similar way. Algorithm 7 iterates on each MSO set and searches for every causally consistent interpretation in theMSO through algorithm 8.

Algorithm 6: Step 1.2: Find possible MSOs

```
function find_possible_mso (M, $possible\_mso, variables, SMSO$)
if $variables$ == {} then
    if $possible\_mso$ is minimal w.r.t. $SMSO$ then
        remove every superset of $possible\_mso$ in $SMSO$;
        insert $possible\_mso$ in $SMSO$
    end
else
    foreach equation $e' \in M$ do
        foreach $y \in (e'_{unknowns} \cap variables)$ do
            find_possible_mso ($M \setminus \{e'\}$, $possible\_mso \cup \{e'\}$,
            $variables \cup \{e'_{unknowns}\} \setminus \{y\}$, $SMSO$);
        end
    end
end
```

Algorithm 7: Step 2.1: For each possible MSO, we test any possible causal matching, $possible\_cm$; SCM is the set of causal matchings

```
function SCM := find_every_possible_cm_foreach_mso (SMSO)
foreach mso in SMSO do
    find_possible_cm($mso$, {}, $SCM$)
end
```

Algorithm 8: Step 2.2. $possible\_cm$ is a valid causal matching for the MSO if it is consistent for every equation in the MSO

```
function find_possible_cm ($mso$, $possible\_cm$, $SCM$)
if $mso == \emptyset$ then
    insert $possible\_cm$ in $SMC$ ;
else
    foreach equation $e'$ in $mso$ do
        foreach causal interpretation $c'$ in $e'$ do
            if consistent($c'$,$possible\_cm$) then
                $mso = mso \setminus \{e'\}$;
                $possible\_cm = possible\_cm \cup \{c'\}$;
                find_possible_cm($mso$ , $possible\_cm$, $SCM$);
            end
        end
    end
end
```

Regarding the computation of the complete set of global causal matchings for each MSO, algorithm *find_possible_cm()* recursively explores every possible causal matching. Although these expressions are semantically identical, it might be useful to have different executable models when non-linearities and dynamics are involved.

5. COMPUTATIONAL COMPLEXITY

In general the number of MSO sets may grow exponentially in the number of equations. This gives an upper bound for the computational complexity in the general case.

MSO-Algorithm In many applications the order of structural redundancy, which depends on the number of available sensors, is low and in this case better computational complexity can be achieved. In the worst case, all unknown variables are included in each equation. The complete version of Algorithm 1 traverses so called PSO sets, sets where $M = M^+$, exactly once in the subset lattice. Given a model with n equations and with structural redundancy φ, there are at most

$$\sum_{k=n-\varphi+1}^{n} \binom{n}{k} \quad (2)$$

PSO subsets. For a fixed order of structural redundancy φ, the complete version of Algorithm 1 has order of $n^{\varphi+1.5}$ time complexity, where n is the number of equations. The proof can be found in Krysander et al. [2008].

CBMSOs-Algorithm The computational complexity can be studied in two parts:

(i) The ranking algorithm used to find the basic MSO sets has complexity $O(nm)$ where n is the number of constraints and m is the number of unknown variables Blanke and Lorentzen [2006].

(ii) Being φ the structural redundancy of a model, the φ basic MSO sets are combined, using Algorithm 2, in groups of 2 to φ (in the worst case). For each combination, there are at most r shared constraints to be removed. r is less than or equal to m. For a worst case and for a fixed number of unknowns, the computational complexity of the algorithm is exponential, and for a fixed order of structural redundancy, it has polynomial time complexity Gelso et al. [2008].

SARR-Algorithm The SARR generation algorithm formulates the problem of generating SARRs like a variable elimination problem. Variable elimination algorithms have been widely studied in the literature and it is well known that they are efficient if the problem is sparse, but otherwise require substantial time and memory. Their performance can be bounded using a graph parameter, called the induced width, which is a measure of the dependencies between variables. The complexity of variable elimination algorithms is time and space exponential in the induced width of the problem interaction graph Dechter [2003]. The induced width depends on the variable ordering chosen to perform the eliminations. The choice of the elimination ordering makes no difference to the final result but makes an immense difference to the efficiency of the variable elimination algorithm.

PCC-Algorithm The complete set of MSOs is found using Depth First Search, DFS, with backtracking on the whole hyper-graph representing the system model, M_{SD}. Algorithm 5 starts with each equation, e, in M_{SD}. Therefore, complexity is $O(n^d)$ for n equations, and an average of d unknowns in each equation. Each MSO_i has an average of n_e equations; in the general case $n_e \leq n$; in practice, $n_e < n$ or $n_e << n$. Each equation $e \in MSO_i$ has an average of k causal interpretations. Algorithm 7 also performs DFS with backtracking to find every possible consistent matching for every equation in MSO_i. Therefore, we have again exponential complexity $O({n_e}^k)$, but it is applied to a reduced problem. Moreover, there is a limited number of causal assignments for each equation, i.e., not every causal assignment is feasible.

6. EMPIRICAL COMPARISON

The benchmark problem concerns the two coupled tanks depicted in figure 1, that provide a continuous water flow Q_0 to a consumer.

The components of the benchmark are tanks T1 and T2, controllers PI and On/Off, pump P1, valves Vb and Vo, level transducers LT1 and LT2, flow transducer FT, mU_p, and U_p transducer S(U_p). The set of known variables Z are the two tanks levels my_1 and my_2, the pump flow mQ_p, the output signal of the PI controller mU_p, the PI set point h_{1c}, the positions mU_b and mU_0 of valves Vb and Vo, respectively. Then, $Z = [my_1, my_2, mQ_p, mU_p, h_{1c}, mU_b, mU_0]$. The subset X of unknown variables contains the remaining variables.

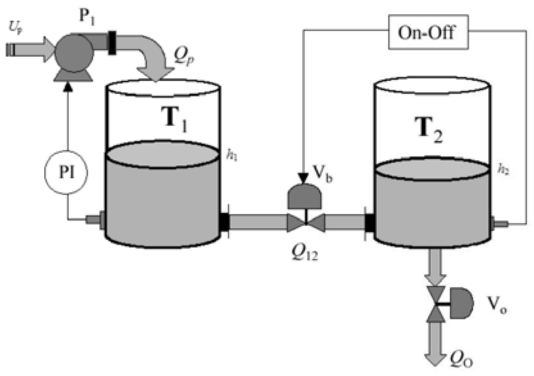

Fig. 1. The Chem benchmark.

The equational model for the benchmark is the following:

Equation	Support
$e_1 : A_1\dot{h}_1 = Q_p - Q_{12}$	OK(T1)
$e_2 : A_2\dot{h}_2 = Q_{12} - Q_0$	OK(T2)
$e_3 : Q_p = \begin{cases} U_p \text{ if } 0 < U_p < Q_p\text{max} \\ 0 \text{ if } U_p \leq 0 \\ U_p\text{max if } U_p \geq Q_p\text{max} \end{cases}$	OK(P1)
$e_4 : Q_{12} = C_{vb} \cdot sgn(h_1 - h_2)\sqrt{\lvert h_1 - h_2 \rvert}$	OK(Vb)
$e_5 : Q_0 = C_{vo} \cdot \sqrt{h_2} \cdot mU_0$	OK(Vo)
$e_6 : U_p = K_p(h_{1c} - h_1(t)) + K_i \int (h_{1c} - h_1(t))dt$	OK(PI)
$e_7 : mU_b = \begin{cases} 0 \text{ if } 0.09m \leq h_2 \geq 0.11m \\ 1 \text{ if } 0 \leq h_2 < 0.09m \end{cases}$	OK(On/Off)
$e_8 : my_1 = h_1$	OK(LT1)
$e_9 : my_2 = h_2$	OK(LT2)
$e_{10} : mQ_p = Q_p$	OK(FT)
$e_{11} : mU_p = U_p$	OK(S(U_p))

A_1 and A_2 are the cross section areas of the cylindric tanks; K_p and K_i are the gains and the integral terms of the PI controller, respectively; C_{vb} and C_{vo} are the global hydraulic flow coefficients of the valves Vb and Vo; and mU_0 is the Vo valve position $\in \{0, 1\}$ provided by the user.

Results For the benchmark model, the MSO-Algorithm and the CBMSOs-Algorithm both consistently provided the 46 MSO sets listed in table 1. The numbers between brackets refer to model equations by their index. 24 out of 46 MSOs are also minimal SARRs (those marked with a star), as found by the SARR-Algorithm and the PCC-Algorithm, that also found the 46 MSO sets.

7. RELATED WORK

This paper gathers four recent algorithms using structural analysis to determine the MSOs, all their causal interpretations or associated SARR. Other algorithms have been proposed that closely relate to the topic of this paper.

In Ploix et al. [2005], an algorithm based on elimination rules is presented. It relies on a structural analysis of the constraints, and information about causality can be taken into account to discard unachievable overdetermined sets. This algorithm can

Table 1. The 46 MSO sets and 24 SARRs in the CHEM-benchmark model.

n^o	MSO sets	n^o	MSO sets
1	$\{7,9\}^*$	24	$\{1,2,5,7,8,10\}$
2	$\{6,8,11\}^*$	25	$\{1,2,5,6,9,10,11\}$
3	$\{3,10,11\}^*$	26	$\{1,2,5,6,7,10,11\}$
4	$\{3,6,8,10\}^*$	27	$\{1,2,4,5,9,10\}^*$
5	$\{2,4,5,8,9\}^*$	28	$\{1,2,4,5,8,10\}^*$
6	$\{2,4,5,7,8\}^*$	29	$\{1,2,4,5,7,10\}^*$
7	$\{2,4,5,6,9,11\}$	30	$\{1,2,4,5,6,10,11\}^*$
8	$\{2,4,5,6,7,11\}$	31	$\{1,2,3,5,8,9,11\}$
9	$\{2,3,4,5,6,9,10\}$	32	$\{1,2,3,5,7,8,11\}$
10	$\{2,3,4,5,6,7,10\}$	33	$\{1,2,3,5,6,9,11\}$
11	$\{1,4,8,9,10\}^*$	34	$\{1,2,3,5,6,9,10\}$
12	$\{1,4,7,8,10\}$	35	$\{1,2,3,5,6,8,9\}$
13	$\{1,4,6,9,10,11\}^*$	36	$\{1,2,3,5,6,7,11\}$
14	$\{1,4,6,7,10,11\}$	37	$\{1,2,3,5,6,7,10\}$
15	$\{1,3,4,8,9,11\}^*$	38	$\{1,2,3,5,6,7,8\}$
16	$\{1,3,4,7,8,11\}$	39	$\{1,2,3,4,5,9,11\}^*$
17	$\{1,3,4,6,9,11\}^*$	40	$\{1,2,3,4,5,8,11\}^*$
18	$\{1,3,4,6,9,10\}^*$	41	$\{1,2,3,4,5,7,11\}^*$
19	$\{1,3,4,6,8,9\}^*$	42	$\{1,2,3,4,5,6,11\}^*$
20	$\{1,3,4,6,7,11\}$	43	$\{1,2,3,4,5,6,10\}^*$
21	$\{1,3,4,6,7,10\}$	44	$\{1,2,3,4,5,6,9\}^*$
22	$\{1,3,4,6,7,8\}$	45	$\{1,2,3,4,5,6,8\}^*$
23	$\{1,2,5,8,9,10\}$	46	$\{1,2,3,4,5,6,7\}^*$

be put in the variable elimination class and is therefore to be related to Algorithm 3.

Blanke and Lorentzen [2006] presents a Matlab® toolbox called SaTool. It is an implementation of the structural analysis theory of Staroswiecki and co-workers Staroswiecki [2002]. A ranking algorithm Blanke and Lorentzen [2006] is used to find a complete matching in the structure graph. The overdetermined sets are obtained by means of the constraints that are not involved in the complete matching. This algorithm, which does not deliver all the MSOs, is the starting point of Algorithm 2.

In Izadi-Zamanabadi [2002], as contrasted with the algorithm cited above, an algorithm based on determining several matchings is presented. Different matchings result in different overdetermined subsystems. This approach is developed to deal with causality by decomposing the system into different parts, which is also a mean to circumvent the complexity problem.

Dustegor et al. [2004] can also deal with causality. To find a matching, a method based on a class of algorithms that solve the Stable Marriage Problem is presented and adapted for the fault detection purpose. In mathematics, the Stable Marriage Problem is a well-known combinatorics problem dealing with finding a stable matching, i.e. a matching in which no element of the first matched set prefers an element of the second matched set that has not the inverse preference.

In Ligeza and Gorny [2000], an algorithm for potential conflict generation in CA-EN-like causal structures is presented. This algorithm identifies potential conflicts by backward propagation over a causal structure (the Potential Conflict Structure, PCS). It would be equivalent to directly find the whole set of SARRs given a causal model (every causal assignment is available for each constraint).

For non-linear polynomial models, algorithms based on elimination techniques, e.g. Gröbner Basis, can be used to obtain analytical redundancy relations of the system Frisk [2000], Staroswiecki and Comtet-Varga [2001], Ceballos et al. [2004]. As basis of this method, model equations are manipulated to eliminate unknown variables such as disturbances, faults and internal states.

8. CONCLUSION

This work has presented and compared four recently proposed algorithms for computing the set of constraints involved in residual generation, together with the computational sequence required to do it. This work clarifies the underlying concepts of the algorithms and their relationships and provides an analysis of their theoretical complexity. The algorithms have been tested on a benchmark and have provided consistent results. Future work should extend this survey to the alternative algorithms mentioned in the related work section.

REFERENCES

M. Blanke and T. Lorentzen. Satool - a software tool for structural analysis of complex automation systems. In *Proceedings of IFAC Safeprocess'06*, Beijing, China, 2006.

R. Ceballos, M.T. Gómez, R. M. Gasca, and S. Pozo. Determination of possible minimal conflict sets using components clusters and Gröbner bases. In *Proceedings of DX'04*, pages 21–26, 2004.

M.-O. Cordier, P. Dague, F. Lévy, J. Montmain, M. Staroswiecki, and L. Travé-Massuyès. Conflicts versus analytical redundancy relations : A comparative analysis of the model-based diagnostic approach from the artificial intelligence and automatic control perspectives. *IEEE Trans. Syst. Man Cy. B.*, 34 (5):2163–2177, 2004.

R. Dechter. *Constraint Processing*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

A. L. Dulmage and N. S. Mendelsohn. Coverings of bipartite graphs. *Canadian Journal of Mathematics*, 10:517–534, 1958.

D. Dustegor, V. Cocquempot, and M. Staroswiecki. Structural analysis for residual generation: towards implementation. *Proceedings of the IEEE International Conference on Control Applications*, 2:1217–1222, 2004.

E. Frisk. Residual generator design for non-linear, polynomial systems - A gröbner basis approach. In *Proceedings of IFAC Safeprocess'00*, Budapest, Hungary, 2000.

E.R. Gelso, S.M. Castillo, and J. Armengol. An algorithm based on structural analysis for model-based fault diagnosis. In *Artificial Intelligence Research and Development. Frontiers in Artificial Intelligence and Applications*, volume 184, pages 138–147. IOS Press, 2008.

R. Izadi-Zamanabadi. Structural analysis approach to fault diagnosis with application to fixed-wing aircraft motion. *Proceedings of the American Control Conference, 2002.*, 5:3949–3954, 2002.

M. Krysander, J. Åslund, and M. Nyberg. An efficient algorithm for finding minimal over-constrained sub-systems for model-based diagnosis. *IEEE Trans. Syst. Man Cy. A.*, 38(1), 2008.

A. Ligeza and B. Gorny. Systematic conflict generation in model-based diagnosis. In *Proceedings of IFAC Safeprocess'00*, pages 1103–1108, Budapest, Hungary, 2000.

M. Nyberg and M. Krysander. Combining AI, FDI, and statistical hypothesis-testing in a framework for diagnosis. In *Proceedings of IFAC Safeprocess'03*, Washington, USA, 2003.

S. Ploix, M. Désinde, and S. Touaf. Automatic design of detection tests in complex dynamic systems. In *16th IFAC World Congress*, Prague, 2005.

B. Pulido and C. Alonso-González. Possible conflicts: a compilation technique for consistency-based diagnosis. *IEEE Trans. Syst. Man Cy. B., Special Issue on Diagnosis of Complex Systems*, 34(5):2192–2206, 2004.

R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

M. Staroswiecki. *Structural analysis for fault detection and isolation and for fault tolerant control*, chapter Encyclopedia of Life Support Systems. Fault Diagnosis and Fault Tolerant Control. Oxford, UK, 2002.

M. Staroswiecki and G. Comtet-Varga. Analytical redundancy relations for fault detection and isolation in algebraic dynamic systems. *Automatica*, 37 (5):687–699, 2001.

L. Travé-Massuyès, T. Escobet, and X. Olive. Diagnosability analysis based on component supported analytical redundancy relations. *IEEE Trans. Syst. Man Cy. A.*, 36(6), 2006.