# A Filtering Process for General Constraint-Satisfaction Problems : Achieving Pairwise-Consistency Using an Associated Binary Representation

**P. Janssen P. Jégou B. Nouguier M.C. Vilarem**

**Centre de Recherche en Informatique de Montpellier**
**860, Rue de Saint Priest, 34100 Montpellier FRANCE**
**Tel : (33) 67 63 04 60 Fax : (33) 67 54 30 79 Email : CRIM@FRMOP11.BITNET**

## ABSTRACT

In this paper, we are interesting in using a partial consistency, issued from relational databases theory, within the Constraint-Satisfaction Problems (CSPs) framework : pairwise-consistency. This partial consistency concerns general CSPs (i.e. CSPs the constraints of which may involve more than two variables). We provide a polynomial algorithm for achieving it; then we can extend the class of polynomially solvable CSPs. This algorithm is based on a minimal binary representation of a general CSP, of which presents some properties we give.

**Key words :** Constraint-Satisfaction Problems (CSP), partial consistency, arc-consistency, pairwise-consistency, qual graph.

* this work was supported by GS DIALR

## 1 INTRODUCTION

Constraint-satisfaction problems (CSPs) involve the assignment of values to variables which are subject to a set of constraints. Examples of CSPs are map coloring, conjunctive queries in a database [2], line drawings understanding [17] [19] [22], pattern matching in production rules systems [20], combinatorial puzzles, peptidic synthesis [13]... In the general case, finding a solution or testing if a CSP admits a solution is a NP-complete problem. Therefore, different levels of consistency have been introduced. The methods to achieve these local consistencies are considered as filtering algorithms : they may lead to problem simplifications, without changing the solution set. They have been used as well to improve the representation prior the search, as to improve backtrack during the search [10]. They have also been used in the framework of dynamic CSPs [11].

Most of these consistencies deal only with binary CSPs (CSP with only binary constraints). We propose here another partial consistency, pairwise-consistency, for general CSPs, and a polynomial algorithm to process it; this allows us to extend the class of polynomially solvable CSPs. The basic idea is to represent a general CSP by a binary CSP, while preserving structural properties.

In this paper, we first recall CSP's definition and main results. Then we define pairwise-consistency and its properties; we provide also a polynomial algorithm to achieve pairwise-consistency. This algorithm is based on an equivalent binary representation for the general CSP. There can be several similar representations; therefore, in the last part, we give an algorithm to find a minimal one, the properties of which allow the extension of Freuder's theorem [9] to $\alpha$-acyclic CSPs [2].

## 2 CONSTRAINT-SATISFACTION PROBLEMS

### 2.1 Definition

A CSP involves a set X of n variables $X_1,X_2,...X_n$ having domains $D_1,D_2,...D_n$ and a set C of m constraints $C_1,...C_m$. Each $D_i$ defines the set of values that variable $X_i$ may assume. D is the set of all domains. A constraint $C_i$ is defined on a set of variables $(X_{i_1},...X_{i_{j_i}})$ by a subset of the cartesian product $D_{i_1}x... D_{i_{j_i}}$; we note this subset $P_i$ ($P_i$ is the set of value arrangements satisfying the constraint $C_i$). P is the set of all $P_i$, for i =1..m.

In this paper, we will write a CSP $\mathcal{P}$ as $\mathcal{P}$=(X,D,C,P).

A CSP solution is a value assignment for all variables, such that all the constraints are satisfied. For a CSP $\mathcal{P}$, the hypergraph (X,C) is called the constraint hypergraph. If all constraints involve at most two variables, the CSP is a binary CSP, and (X,C) is a graph (generally called constraint graph). For a given CSP. the problem is either to find all solutions or a solution, or to know if there exists any solution. All these problems are



1984/89/0000/0420$01.00 © 1989 IEEE

known to be NP-complete.

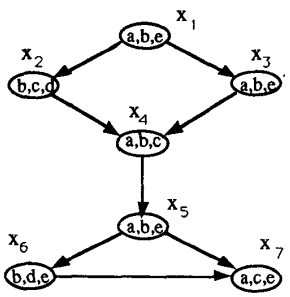


**Figure 1** : this is a CSP with variables X = {$X_1$,$X_2$,$X_3$,$X_4$, $X_5$,$X_6$,$X_7$}; the constraints are the arcs of the graph; each variable domain is given by the set of associated letters. The constraints are defined by non strict lexicographic ordering : for example, the constraint ($X_2$,$X_4$) is defined by the set {(b,b),(b,c),(c,c)}. This problem has several solutions; the assignment (a,b,a,b,b,d,e) to X is one of them.

## 2.2 Main results

CSPs are normally solved by different versions of backtrack search. Consequently, many works try to improve the search efficiency. They mainly deal with binary CSPs. Freuder, considering the problem of finding one solution, gives a preprocessing procedure for selecting a good variable ordering prior to running the search. One of his main results is a sufficient condition for backtrack-free search [9]. This condition concerns on one hand a structural property of the constraint graph, and on the other hand more or less local consistencies. Roughly speaking, we could express the idea as : "the more the constraint graph is connected, the higher the level of the consistency must be". Nevertheless, one can use this theorem only for simple cases : consistency verification is easy only for low level consistencies. Moreover, in the other cases, achieving consistency leads to a structural modification of the constraint graph. As an outcome, Freuder [9], Pearl and Dechter [6] give two classes of polynomially solvable CSPs. See for example :

**Corollary 2.1** [9] : if the constraint graph is acyclic, then there is a polynomial algorithm to find a solution.

To solve the general problem, various efforts have been made to improve backtrack's performance; they can be classified along the two following dimensions :
- improving backtrack during search (see [4] [6] [10] [18])
- improving the representation prior to search : these techniques are mainly based either on some decomposition of the problem (cycle-cutset method [5] or tree-clustering method [7]), or on the obtention of some partial consistencies [8] [14] [19] [22].

Here, we develop here the last approach, more precisely arc-consistency [22], also known as domain-filtering. To verify the arc-consistency, any value $d_i$ of a variable $X_i$ needs to be compatible with any constraint involving $X_i$.

**Definition** : a domain $D_i$ is **arc-consistent** iff $D_i \neq \emptyset$ and $\forall d_i \in D_i$, $\forall C_j \ni X_i$, $d_i \in P_j[X_i]$, where $P_j[X_i]$ is the restriction of $P_j$ to $X_i$. A CSP is **arc-consistent** if each one of the variables domains is arc-consistent.

**Property 2.2** [9] : if a binary CSP is arc-consistent, and if its constraint graph is acyclic, then the CSP admits a solution and there is a backtrack-free search order.

Nevertheless, in the general case, an arc-consistent CSP does not necessarily admit a solution. If the given CSP is not arc-consistent, we can still transform it in a CSP $\mathcal{P}'$ : the domains $D'_i$ of $\mathcal{P}'$ are obtained by deleting the values of $D_i$ when the arc-consistency condition is not true. If one of the $D'_i$ is empty, $\mathcal{P}$ does not admit a solution, else $\mathcal{P}'$ is arc-consistent. $\mathcal{P}'$ is called the arc-consistent-closure of $\mathcal{P}$. It presents the following properties :
- $\mathcal{P}$ and $\mathcal{P}'$ have the same solution set.
- $\mathcal{P}'$ and $\mathcal{P}$ have the same constraint hypergraph.
- $\mathcal{P}'$ is somehow simpler than $\mathcal{P}$ : $\forall$ i, $D_i' \subseteq D_i$.

So, processing the arc-consistent-closure is a filtering process deleting values which cannot belong to any solution. In [16], it is shown that arc-consistency can be achieved by a linear algorithm.

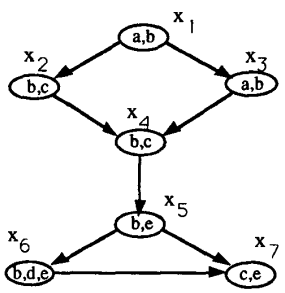


**Figure 2** : figure 2.1 CSP arc-consistent closure. Note for example that value e has been deleted in $D_1$ because it does not exist any value in $D_2$ compatible with it for the constraint C=($X_1$,$X_2$).

## 3 PAIRWISE-CONSISTENCY

Here, we present a new form of partial consistency within the CSP framework : the pairwise-consistency.

### 3.1 Partial consistencies for general CSP

Only one partial consistency has been studied within the framework of general CSPs : arc-consistency. The results are quite similar to those obtained in the case of binary CSPs : generalization of the property 2.2 [13] and linear algorithms for the processing of the arc-consistent closure [13] [15]. These results led to the following corollary.

**Corollary 3.1.** [13] : if the constraint hypergraph is Berge-acyclic, there exists a polynomial algorithm to find a solution.

The cycle definition used in this corollary is given in [1]. In the following chapters we use another definition, proposed in [2] and called $\alpha$-cycle. We do not report it here, knowing the following properties is sufficient : (1) any $\alpha$-cycle is a Berge cycle; the converse is false [2], and (2) there is a linear algorithm to test if a hypergraph is $\alpha$-acyclic [21].

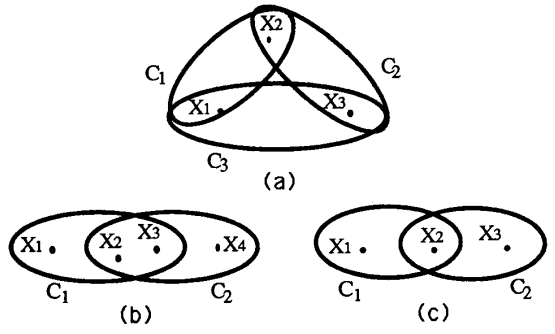


**Figure 3 :** cyclicity and hypergraphs : the hypergraph (a) is Berge-acyclic and $\alpha$-acyclic, (b) is Berge-cyclic but $\alpha$-acyclic, and (c) is both Berge-cyclic et $\alpha$-cyclic.

We define the pairwise-consistency, basing on works concerning relational databases [2]. It allows to extend the corollary 3.1 to $\alpha$-acyclic hypergraphs. Whereas arc-consistency is a local consistency between variables domains and constraints, pairwise-consistency defines a consistency between constraints.

**Definition :** a CSP $\mathcal{P} = (X,D,C,P)$ is **pairwise-consistent** iff

- $\forall C_i, \forall C_j / C_i \cap C_j \neq \emptyset$ , $P_i[C_i \cap C_j] = P_j[C_i \cap C_j]$ and $P_i[C_i \cap C_j] \neq \emptyset$
- $\forall P_i, P_i \neq \emptyset$ .

(the expression $P_i[C_i \cap C_j]$ represents the restriction of $P_i$ to the variables of $C_i \cap C_j$).

There is no direct link between arc and pairwise-consistency : a pairwise-consistent CSP may be not arc-consistent, and reciprocally (example 1).

**Example 1 :** arc-consistent but not pairwise-consistent CSP: Consider the CSP $\mathcal{P}_1 = (X,D,C,P)$

$X = \{X_1,X_2,X_3,X_4\}$,

$D = \{D_1,D_2,D_3,D_4\} / D_i = \{a,b\}$,

$C = \{C_1,C_2,C_3\} / C_1=(X_1,X_2,X_3)$, $C_2=(X_2,X_3,X_4)$, $C_3=(X_1,X_4)$,

$P = \{P_1,P_2,P_3\} / P_1=\{(b,a,a),(a,a,b),(a,b,a)\}$, $P_2=\{(a,b,a), (b,a,a), (b,a,b)\}$, $P_3=\{ (a,a), (b,a), (b,b)\}$.

This problem is arc-consistent but not pairwise-consistent; since : $P_1[C_1 \cap C_2] = \{(a,a), (a,b), (b,a)\} \neq P_2[C_1 \cap C_2] = \{(a,b), (b,a)\}$

A database result [2] has pointed out the link between pairwise-consistency and a constraint hypergraph property. In terms of CSPs, this result is the following :

**Property 3.1 :** If a CSP is pairwise-consistent and if its constraint hypergraph is $\alpha$-acyclic, then it admits a solution which can be obtained in a polynomial time.

### 3.2 The pairwise-consistent closure

As for arc-consistency, it is possible to obtain from pairwise-consistency a filtering operation on general CSPs. To this aim, we associate to each CSP $\mathcal{P}$ a CSP, noted $\mathcal{P}^P$, called pairwise-consistent closure of $\mathcal{P}$ defined below :

**Definition :** let $\mathcal{P} = (X,D,C,P)$ a CSP. The **pairwise-consistent closure** of $\mathcal{P}$ is the CSP $\mathcal{P}^P = (X,D,C,Q)$ defined by :

- $\forall i \; Q_i \subseteq P_i$.
- $\mathcal{P}^P$ is pairwise-consistent
- $\mathcal{P}^P$ is maximal in the following sense : there is no CSP $\mathcal{P}' = (X,D,C,R)$ such as $\forall i \; Q_i \subseteq R_i \subseteq P_i$ , $\mathcal{P}'$ pairwise-consistent and $Q \neq R$.

It can be underlined that the pairwise-consistent closure of a CSP $\mathcal{P}$ does not necessarily exist. Thus, the single CSP $\mathcal{P}'$ which verifies the property ($\forall C_i, \forall C_j / C_i \cap C_j \neq \emptyset$, $Q_i[C_i \cap C_j] = Q_j[C_i \cap C_j]$ ) may be such that, for some i, $Q_i[C_i \cap C_j] = \emptyset$, i. e. $Q_i = \emptyset$. In this case $\mathcal{P}$ has no solution.

**Property 3.2 :** When the pairwise-consistent closure of a CSP exists, it is unique.

**Proof :** suppose that $\mathcal{P}_1 = (X,D,C,Q)$ and $\mathcal{P}_2 = (X,D,C,R)$ are two distinct pairwise consistent closures of a CSP $\mathcal{P} = (X,D,C,P)$. Let us build the CSP $\mathcal{P}' = (X,D,C,T)$ the following way : $\forall i \quad T_i = R_i \cup Q_i$ ; one can easily see that $\mathcal{P}'$ is pairwise-consistent and therefore $\mathcal{P}_1$ and $\mathcal{P}_2$ are not maximal. ■

**Property 3.3 :** If $\mathcal{P}^p$ exists, then $\mathcal{P}$ and $\mathcal{P}^p$ have the same set of solutions.

**Proof :** let S be a solution of $\mathcal{P} = (X,D,C,P)$, S is pairwise-consistent; if S is not a solution of $\mathcal{P}^p = (X,D,C,Q)$ then $\exists i$ : $S[C_i] \notin Q_i$; let $\mathcal{P}' = (X,D,C,R)$ the CSP defined by $\forall i$, $R_i = Q_i \cup S[C_i]$. S and $\mathcal{P}^p$ being pairwise-consistent, $\mathcal{P}'$ is pairwise-consistent and therefore $\mathcal{P}^p$ is not maximal. ■

**Example 2 :** pairwise-consistent closure of the CSP $\mathcal{P}_1$ shown by example 1 : Let $\mathcal{P}_2 = (X,D,C,Q)$ be the pairwise-consistent closure of $\mathcal{P}_1$, we have : $Q_1 = \{ (a,a,b), (a,b,a) \}$, $Q_2 = \{ (a,b,a), (b,a,a) \}$ and $Q_3 = \{(a,a)\}$. It may be seen that $\mathcal{P}_2$ is not arc-consistent since $b \notin D_1$ does not belong to any tuple in $Q_1$.

Another interesting property concerns the links between arc and pairwise-consistency. Although pairwise-consistency does not act on the variables domains, it is easy from a pairwise-consistent CSP to get an arc-consistent CSP.

**Property 3.4. :** let $\mathcal{P} = (X,D,C,P)$ be a CSP; let a CSP $\mathcal{P}^{pac} = (X,D',C,P)$ defined by : $\forall X_i \; D'_i = P_k[X_i]$ where $C_k$ is any constraint involving $X_i$. If $\mathcal{P}$ is pairwise-consistent then $\mathcal{P}^{pac}$ is arc and pairwise-consistent.

**Proof** : showing that $\mathcal{P}^{pac}$ is arc-consistent is sufficient : let $X_i$ be a variable, and $d_i \in D'_i$; from $\mathcal{P}^{pac}$ definition, we have : $\exists C_k \ni X_i$ with $d_i \in P_k[X_i]$. Since $\mathcal{P}$ is pairwise-consistent, $\forall C_h \ni X_i$ : $P_k[C_h \cap C_k] = P_h[C_h \cap C_k]$ and therefore, as $X_i \in C_h \cap C_k$, $\forall C_h \ni X_i$, $P_k[X_i] = P_h[X_i]$. So $\forall X_i \in X$, $\forall d_i \in D'_i$, $\forall C_h \ni X_i$, $d_i \in P_h[X_i]$.
Consequently, $\mathcal{P}^{pac}$ is arc-consistent. ■

Using this property, it is easy to define a closure based on these two partial consistencies. This way, we obtain another filtering process by achieving first pairwise-consistency, and then arc-consistency.

**Example 3 :** Let $\mathcal{P}_3 = (X,D',C,Q)$ be the arc-consistent closure of $\mathcal{P}_2$ (example 2); it is the arc-pairwise-consistent closure of $\mathcal{P}_1$. We have $D'_1 = \{a\}$, $D'_2 = D_2$, $D'_3 = D_3$ and $D'_4 = \{ a \}$.

Pairwise-consistency is thus a new partial consistency suitable for general CSPs, possibly associated with arc-consistency. We give below an efficient algorithm to achieve it.

## 4 ACHIEVING PAIRWISE-CONSISTENCY

To this aim, we associate to every general CSP $\mathcal{P}_H$ a binary CSP $\mathcal{P}_{G(H)}$ so that achieving pairwise-consistency on $\mathcal{P}_H$ consists in achieving arc-consistency on $\mathcal{P}_{G(H)}$.

### 4.1 Qual graphs

**Definition [3] :** let H = (X,C) a hypergraph. A **qual graph of H** is a graph G(H) = (C,E,v), where v is an edge labelling such that :

- $E \subseteq F = \{ \{C_i,C_j\} \subset C / i \neq j$ and $C_i \cap C_j \neq \emptyset \}$,
- $v(\{C_i,C_j\}) = C_i \cap C_j$
- $\forall C_i, C_j \in C$ if $C_i \cap C_j \neq \emptyset$, then there is in G(H) a chain $(C_i=C_0,C_1,...,C_q=C_j)$ such that $\forall k \in [0,q-1]$, $C_i \cap C_j \subseteq v(\{C_k,C_{k+1}\})$

There may be several qual graphs for the same hypergraph (figure 4). We can notice that the representing graph [1] of H (C,F,v), sometimes called intersection graph, is the greatest element of the qual graphs set.

Given a general CSP $\mathcal{P}_H$, and one of its qual graphs G(H), let us define the associated binary CSP $\mathcal{P}_{G(H)}$ :

**Definition :** Let $\mathcal{P}_H = (X,D,C,P)$ a CSP and H = (X,C) its constraint hypergraph. $\mathcal{P}_{G(H)} = (C,P,E,R)$ with :

- (C,E,v) = G(H),
- $C = \{C_1,... C_m\}$ a variables set with a domains set $P = \{P_1,... P_m\}$,

• if $E_k \in E$ / $E_k = \{C_i, C_j\}$, then $R_k = \{ (p_i,p_j) \in P_i x P_j$ / $p_i[v(E_k)] = p_j[v(E_k)] \}$ (e.g. value $p_i$ for $C_i$ and value $p_j$ for $C_j$ are compatible if the variables of $C_i \cap C_j$ have the same values).

A $\mathfrak{P}_{G(H)}$ solution is an assignment to the variables of C satisfying all the constraints defined by E and R.

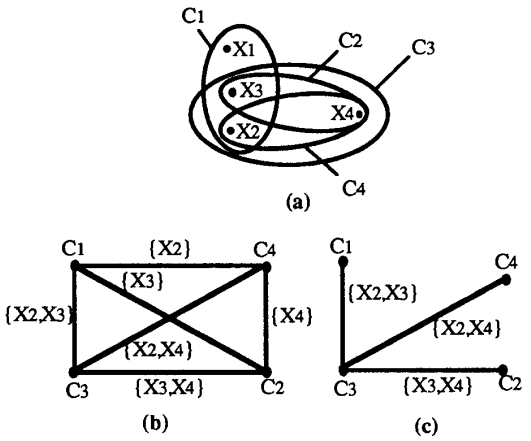


**Figure 4** : a hypergraph and two of its qual graphs

**Property 4.1.** : Let $\mathfrak{P}_H = (X,D,C,P)$ a general CSP, and $\mathfrak{P}_{G(H)} = (C,P,E,R)$ an associated binary CSP, then : $\mathfrak{P}_H$ pairwise-consistent $\Leftrightarrow$ $\mathfrak{P}_{G(H)}$ arc-consistent.

**Proof :**

• $\Rightarrow$

$\mathfrak{P}_H$ pairwise-consistent

$\Rightarrow \forall C_i \in C, \forall E_k=\{C_i, C_j\} \in E, P_i[v(E_k)] = P_j[v(E_k)] \neq \emptyset$ and $\forall i, P_i \neq \emptyset$

$\Rightarrow \forall C_i \in C, \forall E_k=\{C_i, C_j\} \in E, \forall p_i \in P_i, \exists p_j \in P_j, p_i[v(E_k)] = p_j[v(E_k)]$ and $\forall i, P_i \neq \emptyset$

$\Rightarrow \forall C_i \in C, \forall p_i \in P_i, \forall E_k \ni C_i, p_i \in R_k[C_i]$ and $\forall i, P_i \neq \emptyset$

$\Rightarrow \mathfrak{P}_{G(H)}$ arc-consistent.

• $\Leftarrow$

$\mathfrak{P}_{G(H)}$ arc-consistent. Let $C_i, C_j \in C$ / $C_i \cap C_j \neq \emptyset$; there is a chain in G(H), ($C_0=C_i, C_1, C_2, \ldots C_q=C_j$), such that $\forall k \in [0,q-1]$, $C_i \cap C_j \subseteq v(\{C_k,C_{k+1}\})$, let $k \in [0, q-1]$ and $E_h = \{C_k,C_{k+1}\}$. As $\mathfrak{P}_{G(H)}$ is arc-consistent : $\forall p_k \in P_k$, $p_k \in R_h[C_k]$ and $\forall p_{k+1} \in P_{k+1}$, $p_{k+1} \in R_h[C_{k+1}]$ and $P_k \neq \emptyset$ consequently $P_k[v(E_h)] = P_{k+1}[v(E_h)] \neq \emptyset$ and since $C_i \cap C_j \subseteq v(E_h)$, we have $P_k[C_i \cap C_j] = P_{k+1}[C_i \cap C_j] \neq \emptyset$, therefore, $\forall k, f \in [0, q]$, $P_k[C_i \cap C_j] = P_f[C_i \cap C_j] \neq \emptyset$, particularly : $P_i[C_i \cap C_j] = P_j[C_i \cap C_j] \neq \emptyset$

$\mathfrak{P}_H$ is pairwise-consistent. ■

**Corollary 4.1.** : if $(C,P^{ac},E,R)$ is the arc-consistent closure of $\mathfrak{P}_{G(H)}$, then the pairwise-consistent closure of $\mathfrak{P}_H$ is $\mathfrak{P}^p_H=(X,D,C,P^{ac})$.

**Proof :** obvious.

## 4.2 Achieving pairwise-consistency of a CSP

**Method :** To achieve pairwise-consistency of a CSP $\mathfrak{P}_H$ we may consider an associated qual graph G(H) = (C,E,v) and use the corollary 4.1 : achieving arc-consistency of $\mathfrak{P}_{G(H)}$ i.e. $\mathfrak{P}^{ac}_{G(H)}=(C,P^{ac},E,R)$ allows to get $\mathfrak{P}^p_H = (X,D,C,P^{ac})$.

This can be made using the linear algorithm AC4 [16] on $\mathfrak{P}_{G(H)}$. AC4 does not require the explicit building of $\mathfrak{P}_{G(H)}$ : it only requires the ability of testing, for any constraint, two values compatibility, and therefore the explicit construction of R is needless.

**Algorithm evaluation :**

the problem parameters are the following ones :

- n = |X| counts the variables of $\mathfrak{P}_H$,
- m = |C| is the number of constraints in $\mathfrak{P}_H$,
- e = |E| the number of G(H) edges,
- k = max { $|P_i|$ / $P_i \in P$ }, the size of the domains in $\mathfrak{P}_{G(H)}$ (or the size of the constraints in $\mathfrak{P}_H$),
- a = max { $|C_i|$ / $C_i \in C$ }, maximal arity of the $\mathfrak{P}_H$ constraints.

**Property 4.2.** : Achieving pairwise-consistency can be done in polynomial time.

**Proof** : The complexity of AC4 on a given G(H) is $O(eak^2)$ (complexity of AC4 is $O(ek^2)$ and factor a is issued from an elementary test cost). Besides, the intersection graph can be obtained in $O(am^2)$ . The whole complexity is thus $O(eak^2+am^2)$. ■

For a given constraint hypergraph, the intersection graph is the qual graph having the maximal number of edges. Therefore, we might take advantage of using another qual graph with less edges, so that we could reduce the factor e in $O(eak^2 + am^2)$.

## 5 ABOUT MINIMALITY OF QUAL GRAPHS

There can be several qual graphs of very different sizes (see figure 4) associated to a hypergraph. So, in this part, we propose an algorithm to find minimal qual graphs.

### 5.1 Minimal qual graphs

**Definition :** Let H = (X,C) be a hypergraph; $G_m(H)$ is **a minimal qual graph for H** if $G_m(H) = (C,E_m,v_m)$ is a qual graph, and $E_m$ is minimal (i.e. no one of the partial graphs of $G_m(H)$ is a qual graph for H).

**Example 4 :** qual graph (c) of figure 4 is a minimal qual graph for hypergraph (a).

There can be several minimal qual graphs associated to a hypergraph. We show that they have the same number of edges.

**Notations :** given H = (X,C) and G(H) = (C,E,v), let us define the following notations:

$\mathcal{I} = \{ C_i \cap C_j \ / \ C_i \cap C_j \neq \varnothing \}$.

For $A \in \mathcal{I}$, $\mathbb{S}_A = \{ C_i \in C \ / \ A \subseteq C_i \}$

Let $A_1, A_2, \ldots A_h$ be a numbering defining a total order on $\mathcal{I}$ compatible with $\supset$ (i.e. $\forall A_k, A_i \in \mathcal{I} \ / \ A_i \supset A_k \Rightarrow i < k$)

$E^k = \{ e \in E \ / \ v(e) = A_i , i \leq k \}$ and $E_{A_j} = \{ e \in E \ / \ v(e) = A_i \}$.

$G^i(\mathbb{S}_{A_j})$ is the subgraph of $(C,E^i)$ induced by $\mathbb{S}_{A_j}$.

NbCC(G) denotes the number of connected components of G.

We consider below a hypergraph (X,C) and a total order on $\mathcal{I}$ compatible with $\supset$.

**Lemma 5.1. :** Let G = (C,E,v) be a qual graph; if $C_{i_1}, C_{i_2} \in C$ are such that $C_{i_1} \cap C_{i_2} = A_i$, and there is a chain $(C_0=C_{i_1}, \ldots, C_k, \ldots C_q= C_{i_2})$ in G s.t. $\forall k \in [0, q-1], A_i \subseteq v(\{C_k,C_{k+1}\})$, then : $\forall k \in [0, q-1], \{C_k,C_{k+1}\} \in E^i$.

**Proof :** $A_i \subseteq v(\{C_k,C_{k+1}\})=A_r$ and by the total order definition $r \leq i$; so, $\{C_k,C_{k+1}\} \in E^i$. ■

**Lemme 5.2. :** G = (C,E,v) is a qual graph $\Leftrightarrow \forall i \in [1, h]$, $G^i(\mathbb{S}_{A_i})$ is connected.

**Proof :**

• $\Rightarrow$ $\forall C_{i_1}, C_{i_2} \in \mathbb{S}_{A_i}$, $C_{i_1} \cap C_{i_2} = A_j \supseteq A_i$, and there is a chain in G, $(C_0=C_{i_1}, \ldots, C_k, \ldots C_q= C_{i_2})$ such that :

- $\forall k \in [0, q]$, $A_j \subseteq C_k$, then $C_k \in \mathbb{S}_{A_j} \subseteq \mathbb{S}_{A_i}$
- $\forall k \in [0, q-1]$, $\{C_k,C_{k+1}\} \in E^j$ (lemma 5.1.) and $E^j \subseteq E^i$; consequently, the considered chain appears in $G^i(\mathbb{S}_{A_i})$

• $\Leftarrow$ $\forall C_{i_1}, C_{i_2} / C_{i_1} \cap C_{i_2} = A_i$, there is a chain in $G^i(\mathbb{S}_{A_i})$, $(C_0=C_{i_1}, \ldots, C_k, \ldots C_q= C_{i_2})$ such that : $\forall k \in [0,q-1]$, $A_i \subseteq v(\{C_k,C_{k+1}\})$; since $G^i(\mathbb{S}_{A_i})$ is a partial subgraph of G, this chain also appears in G. ■

**Lemma 5.3. :** if G and G' are two qual graphs, then $NbCC(G^i(\mathbb{S}_{A_{i+1}})) = NbCC(G'^i(\mathbb{S}_{A_{i+1}}))$

**Proof :** it is sufficient to prove : $C_{i_1}, C_{i_2}$ connected in $G^i(\mathbb{S}_{A_{i+1}}) \Rightarrow C_{i_1}, C_{i_2}$ connected in $G'^i(\mathbb{S}_{A_{i+1}})$. If $C_{i_1}$ and $C_{i_2}$ are connected in $G^i(\mathbb{S}_{A_{i+1}})$, there is a chain $(C_0=C_{i_1}, \ldots, C_k, \ldots C_q= C_{i_2})$ such that : $\forall k \in [1, q-1], \exists r \leq i \ / \ v(\{C_k,C_{k+1}\}) = A_r$. Then $C_k, C_{k+1} \in \mathbb{S}_{A_r}$, and by lemma 5.2., $C_k$ and $C_{k+1}$ are connected in $G'^r(\mathbb{S}_{A_r})$, and furthermore in $G'^i(\mathbb{S}_{A_{i+1}})$ since $E'^r \subseteq E'^i$ and $\mathbb{S}_{A_r} \subset \mathbb{S}_{A_{i+1}}$. ■

**Lemma 5.4. :** let G = (C,E,v) be a minimal qual graph, then $\forall i \in [0, h-1]$, $|E_{A_{i+1}}| = NbCC(G^i(\mathbb{S}_{A_{i+1}}))-1$

**Proof :** by lemma 5.2., $G^{i+1}(\mathbb{S}_{A_{i+1}})$ is connected. If $\mathbb{S}_{A_{i+1}}$ is not connected in $G^i$, the edges connecting it in $G^{i+1}$ belong to $E_{A_{i+1}}$. The minimum number of these edges is $(NbCC(G^i(\mathbb{S}_{A_{i+1}})) -1)$ and any additional edge would be redundant (its deletion would not disconnect $G^{i+1}(\mathbb{S}_{A_{i+1}})$). Since G is minimal: $|E_{A_{i+1}}| = (NbCC(G^i(\mathbb{S}_{A_{i+1}})) -1)$. ■

**Property 5.1. :** let H = (X,C) be a hypergraph, let G=(C,E,v) and G' = (C,E',v') be two minimal qual graphs for H, then |E| = |E'|.

**Proof :** by lemmas 5.3. and 5.4., $\forall i \in [1, h]$, $|E_{A_i}| = |E'_{A_i}|$, so |E| = |E'|. ■

### 5.2 Computation of a minimal qual graph

The proof of property 5.1 gives the idea of the algorithm : starting with the graph $G^0=(C,E^0)$ where $E^0 = \varnothing$, each element of $\mathcal{I}$ is processed in order $A_1, A_2, \ldots A_h$. At step k, we add edges such that the resulting graph $G^k = (C,E^k)$ has the following properties :

(i) $\forall e \in E^k$, $v(e) \in \{A_1, \ldots A_k\}$.

(ii) $\forall j, 1 \leq j \leq k$, $G^j(\mathbb{S}_{A_j})$ is connected

(iii) let G'=(C,E',v') be a minimal qual graph, then $|E^k| = |E'^k|$.

Property (iii) ensures the final graph minimality, while property (ii) is equivalent to the qual graph condition (lemma 5.2.).

**Algorithm :**

$E^0 \leftarrow \emptyset$;

for k <- 1 to h do

*{$G^{k-1}(\mathbb{S}_{A_k})$ has $n_k$ connected components; for each one of them, we select one vertex; let $C_{n_1}$, ..., $C_{n_k}$ be the resulting set}*

$E^k \leftarrow E^{k-1} \cup \{ n_k - 1$ edges which connect $C_{n_1},..., C_{n_k} \}$

*{if $G^{k-1}(\mathbb{S}_{A_k})$ is already connected, $E^k = E^{k-1}$}*

rof;

$G_m(H) \leftarrow G^h$;

**Proof** : by induction on k :

- for k = 1, properties (i), (ii), (iii) hold
- suppose (i), (ii), (iii) hold for j, j = 1, ... k-1.

(i) is true for $G^k$ : actually, every added edge e connects two vertices of $\mathbb{S}_{A_k}$, then $A_k \subseteq v(e)$; so, from definition of the total order on $\mathfrak{J}$, $v(e) \in \{A_1,..A_k\}$

(ii). $\forall$ j ≤ k-1, $G^j(\mathbb{S}_{A_j})$ is connected. At step k, there is no edge deletion; consequently, $G^j(\mathbb{S}_{A_j})$ remains connected. The purpose of step k is to connect $G^{k-1}(\mathbb{S}_{A_k})$, then $G^k(\mathbb{S}_{A_k})$ is actually connected.

(iii) the number of added edges is $|E_{A_k}| = (NbCC(G^{k-1}(\mathbb{S}_{A_k}))-1)$ (lemma 5.4); this number is equal to $|E'_{A_k}|$ (lemma 5.3 and 5.4).

Property (ii) holds in $G_m(H)$, which consequently is a qual graph; moreover, by property (iii), $G_m(H)$ has the same number of edges as a minimal qual graph (property 5.1), therefore $G_m(H)$ is minimal. ■

**Complexity :**

With a, m and n as in 4.2. and h = |$\mathfrak{J}$|, f = |F| the number of edges in the representing graph of H, and $e_m = |E_m|$ the number of edges in a minimal qual graph for H.

- $\mathfrak{J}$ ordering computation : O(n.f + f.a.log(h)).

Comparing two members of $\mathfrak{J}$ requires O(a) time. We can represent $\mathfrak{J}$ by an array indexed by cardinality of $\mathfrak{J}$ members; the $i^{th}$ member of this array is a tree encoding $A_j$'s of cardinality i. Building this structure requires O(n.f+f.a.log(h)) time.
Finding a total ordering on $\mathfrak{J}$ is achieved in O(h).

- $G_m$ construction : O( h(m + $e_m$) )

Step k : O( $|\mathbb{S}_{A_k}| + |E^k|$ )
searching connected components of $G^{k-1}(\mathbb{S}_{A_k})$ runs in $O(|\mathbb{S}_{A_k}| + |E^{k-1}|)$ time $E_{A_k}$ building is in $O(|E_{A_k}|)$
The overall complexity is O( h(m + $e_m$) ) time because $|\mathbb{S}_{A_k}| \le m$ and $|E^k| \le e_m$.

The total cost of this algorithm is O( n.f + f.a.log(h) + h(m+$e_m$)) time ; this expression can be overestimated by $O(n.m^2.\log(m^2) + m^4)$.

When implemented on a minimal qual graph, the process of pairwise-consistent-closure requires $O(nm^2\log(m^2)+m^4+e_m ak^2)$ time. This complexity seems greater with respect to the previous one : $O(am^2+eak^2)$ (see 4.2.). Nevertheless, we have to consider all the factors : let d be max $|D_i|$, then $k \le d^a$; thus, finding $G_m$ can be advantageous when $e_m << e$. This is often the case in practice, especially when the size of the hypergraph (factors m and n) is small with regard to the constraints size (factors d and k). Furthermore, in the framework of dynamic CSPs we have to solve a sequence of CSPs with only local changes [11]. In many cases, we may very quickly obtain the minimal graph of a new CSP from the previous one. So, the total cost may be less than expected.

## 5.3 Minimal qual graphs properties

We can link a constraint hypergraph and its minimal qual graphs by the following property.

**Property 5.2 :** H α-acyclic (connected) ⇔ $G_m(H)$ is a tree

**Proof :** It has been shown in [2] that H is α-acyclic iff H admits a join-tree G (acyclic qual graph). It is enough to notice that G is minimal since no partial graph of G is a qual graph. ■

Besides, we have shown $\mathfrak{P}_H$ and $\mathfrak{P}_{G(H)}$ are equivalent, in the sense that there is a bijection between their solution sets [12]. This result leads to a generalization of Freuder's corollary 2.1 to α-acyclic CSPs.

**Property 5.3. :** let $\mathfrak{P}_H$ be a general CSP. If H is α-acyclic then there exists a polynomial algorithm to find a solution.

**Proof :** from property 5.2, we can associate to $\mathfrak{P}_H$ an equivalent binary CSP $\mathfrak{P}_{G_m(H)}$, the graph of which is acyclic. This can be polynomially done. In addition, we can polynomially find a solution to $\mathfrak{P}_{G_m(H)}$ (Corollary 2.1). ■

From the previous property, we can notice two points:
- the class of the polynomially solvable CSPs is extended;
- even if H is not α-acyclic, $\mathcal{P}_{Gm(H)}$ may be considered as an equivalent binary representation of $\mathcal{P}_H$. The main advantage of such a representation is the ability of using binary CSPs results. Nevertheless this requires that the transformation maintains polynomially solvability and that the corresponding cost is not too high : acyclicity conservation is a partial answer to the first point.

## 6 CONCLUSION

Pairwise-consistency is a partial consistency on general CSPs; in the framework of relational databases, Beeri & all. have shown it is a necessary condition for a solution to exist. We showed that achieving it consists in a new filtering process; its complexity is polynomial and we gave a bound for it. In addition, for an already pairwise-consistent CSP, it is easy to process arc-consistency. The algorithm presented here is based on a binary representation of general CSPs : we associate to any constraint hypergraph a set of graphs : the qual graphs set, any element of which is a binary representation of the given CSP.

We studied the minimal elements in the qual graphs set, i.e. graphs with a minimum number of edges. We proposed a polynomial algorithm to find one of them. One can use these results to improve the efficiency of achieving pairwise-consistency. Moreover, we showed that any minimal qual graph of an acyclic hypergraph is also acyclic. Consequently, we can extend Freuder's theorem to general CSPs. This representation might be studied in further works, in order to take advantage of the numerous studies concernig binary CSPs .

## REFERENCES


[1] C. Berge, "Graphes et hypergraphes," Dunod (France), 1970.

[2] C.Beeri, R. Fagin, D. Maier & M.Yannakakis, "On the desirability of acyclic database schemes," J. Assoc. Comput., vol. 30, p. 479,1983.

[3] P.A. Bernstein, N.Goodman, "The power of natural semijoins" SIAM J. of computing 10,4,1981.

[4] R. Dechter, "Constraints processing incorporating, backjumping, learning and cutset-decomposition," Proceedings Forth IEEE Conference on AI Applications, San Diego, CA, p. 312, 1988.

[5] R. Dechter & J. Pearl, "The cycle-cutset method for improving search performance in AI applications," Proceedings Third IEEE Conference on AI applications, Orlando, p. 224, 1987.

[6] R. Dechter & J. Pearl, "Network-based heuristics for constraint-satisfaction problems," Artificial Intelligence, vol. 34 p. 1, 1988.

[7] R. Dechter & J. Pearl, "Tree-clustering schemes for constraints-processing," Proceedings AAAI 88, St Paul, p. 150, 1988.

[8] E.C. Freuder, "Synthesizing constraint expressions," Comm. ACM, vol. 21, p. 958, 1978.

[9] E.C. Freuder, "A sufficient condition for backtrack-free search," ACM, vol. 29 (1), p. 24, 1982.

[10] R.M. Haralick & G.L. Elliot, "Increasing tree search efficiency for constraint-satisfaction problems," Artificial Intelligence vol. 14 p. 263, 1980.

[11] P. Janssen, P. Jégou, B. Nouguier, M.C. Vilarem, "Problèmes de conception : une approche basée sur la satisfaction de contraintes," Les systèmes experts et leurs applications, gèmes Journées Internationales, Avignon, p. 71, 1989.

[12] P. Janssen, P. Jégou, B. Nouguier, M.C. Vilarem, "Une opération de filtrage pour les CSP généraux : un calcul de la fermeture paire-consistante basé sur une représentation binaire des CSP n-aires," R.R. C.R.I.M. n° 69, 1989.

[13] P. Janssen & M.C. Vilarem, "Problèmes de satisfaction de contraintes : techniques de résolution et application à la synthèse de peptides," R.R. C.R.I.M. n° 54, 1988.

[14] A.K. Macworth & E.C. Freuder, "The complexity of some polynomial network consistency algorithms for constraint satisfaction problems," Artificial Intelligence 25, p. 65, 1985.

[15] R. Mohr & G. Masini, "Good old discrete relaxation," ECAI 89, Munich, p. 651, 1989.

[16] R. Mohr & T.C. Henderson, "Arc and path consistency revisited," Artificial Intelligence, vol. 28 (2), 1986.

[17] U. Montanari, "Networks of constraints : fundamental properties and applications to picture processing," Information Sciences, vol. 7, 1974.

[18] B.A. Nadel, "Three consistent labelling algorithms and their complexities : search-order dependent and effectively instance-specific results," Report DCS-TR-171, Computer Sc. Dept. Rutgers University New Brunswick, NJ, 1986.

[19] A. Rosenfeld, R. Hummel & S. Zucker, "Scene labeling by relaxation operations," IEEE Trans. Systems Man Cybernet. vol. 6, p. 420, 1976.

[20] G. Sabatier, "Optimisation du pattern matching pour un système du type SNARK," 6° Congrès AFCET INRIA Reconnaissance des formes et intelligence artificielle, Antibes, Fr., p. 799, 1987.

[21] R.E. Tarjan & M. Yannakakis, "Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs," SIAM J Comput., vol. 13,3, p. 566, 1984.

[22] D.L. Waltz, "Understanding line drawings of scenes with shadows," P.H.Winston (Editor), The Psychology of Computer Vision, Mc Graw-Hill , New-York, p. 19, 1975.