

# Mining historical data to build constraint viewpoints

Christian Bessiere, Joël Quinqueton, Gilles Raymond

Université Montpellier 2, LIRMM

## Abstract

Constraint programming has shown convincing success in solving real world problems from the industry and is now widely used on industrial tasks. However, designing Constraint Networks is limited to experts in this domain, which constitutes a bottleneck for spreading this approach.

Automated or partially automated constraint network modelling appears a good solution to overcome this weakness. Promising initial results have been reached in constraint network learning; however, all of these results take for granted the variables and the domains of the constraint network to build.

Therefore, we propose here an automated method to generate several different viewpoints for the problem we seek to model. To do so, we process existing solutions to problems close to the target problem. Our main idea is that a viewpoint general enough to describe different solutions of close problems will also be able to describe any solution of the target problem.

As an experimental demonstration we provide viewpoints built with our method for three different kinds of problems.

## 1 Introduction

The CSP approach to problem resolution has proved its effectiveness and is now widely employed in industry on problems with very large numbers of parameters. Unfortunately, even if the basic notions of constraint networks can be reasonably easily explained to non-experts, these concepts remain difficult for them to put into practise on real problems. Modelling is far from being easy and is reserved to specialists in the domain. This bottleneck needs to quickly disappear for the CSP approach to keep its users and gain new ones.

In the field of CSP partially automatic modelling a very global approach is presented in [1]. On the other hand, [2] and [3] are interesting works aimed at constraint learning. However, these efforts start from known sets of variables and domains. Nonetheless, determining these sets is complex in practice, and is crucial in obtaining an efficient constraint network. This is why we propose here an automatic method to determine these sets, as the initial step of the modelling process.

We start with a set of historical data. The historical data that we use are solutions to problems close to the target problem, the one we seek to model. From

these data, we extract candidate variables and their domains with which we can construct viewpoints (constraint networks restricted to variable and domain sets): each of these viewpoints is able to represent all of these historical solutions. By hypothesis, the target problem has a lot in common with the close problems for which we have known solutions. So we propose that these viewpoints are not only capable of describing the historical solutions but also the solutions of our target problem.

Clearly, the initially supplied data must conform to certain conditions in order to justify this proposition: these conditions are explored in this paper. At the end of the process we obtain a set of potential viewpoints from which we will have to select the more relevant in order to build efficient constraint models. This following step in automatic constraint network modelling is not described in this paper.

Our method has been tested on three very different problems using our tool, *ViewPointDigger*.

This approach is fully compatible with the previously cited work in constraint learning. Combining these complementary techniques should lead to theoretical progress, which in turn will lead to medium-term progress for practical applications of CSP in the widest world.

## 2 Background

In this section, we present a basic revision of the constraint programming formalism and we define the vocabulary associated to the historical data that we will work on.

### 2.1 CSP formalism

**Definition 1 (Constraint network)** *A constraint network is defined as a triplet  $(\mathcal{X}, \mathcal{D}, \mathcal{C})$  where:*

$\mathcal{X} = \{X_1, X_2, \dots, X_n\}$  *is a set of variables.*

$\mathcal{D} = \{D_{X_1}, D_{X_2}, \dots, D_{X_n}\}$  *is the set of their domains: each variable  $X_i$  takes its values from the domain  $D_{X_i}$ .*

$\mathcal{C} = \{C_1, C_2, \dots, C_m\}$  *is a sequence of constraints on  $\mathcal{X}$  and  $\mathcal{D}$ , where a constraint  $C_i$  is defined by the sequence  $var(C_i)$  of variables it involves, and the relation  $rel(C_i)$  specifying the allowed tuples on  $var(C_i)$ .*

**Definition 2 (Instance)** *Let  $Y = \{Y_1, Y_2, \dots, Y_k\}$  be a subset of  $\mathcal{X}$ . An instance  $e_y$  on  $Y$  is a tuple  $(v_1, v_2, \dots, v_k) \in D_{Y_1} \times D_{Y_2} \times \dots \times D_{Y_k}$ . This instance is partial if  $Y \neq \mathcal{X}$ , complete otherwise (noted  $e$ ). An instance  $e_Y$  on  $Y$  violates the constraint  $C_i$  iff  $var(C_i) \subseteq Y$  and  $e_Y[var(C_i)] \notin rel(C_i)$ .*

**Definition 3 (Solution of a constraint network)** *A complete instance over the set  $\mathcal{X}$  of variables is a solution of the constraint network  $N = (\mathcal{X}, \mathcal{D}, \mathcal{C})$  iff it does not violate any constraint. Otherwise it is a non-solution.  $Sol(N)$  denotes the solutions over  $N$ .*

**Definition 4 (Constraint viewpoint)** A *constraint viewpoint* is defined as a tuple  $(\mathcal{X}, \mathcal{D})$ . It is the part of a constraint network without the constraint sequence.

The notion of constraint viewpoint was informally introduced by Geelan92 [4] and formalized by Law and Lee [5]. The importance of this concept in the constraint network design process has been established in several other works ([6], [7] and [8]).

## 2.2 Historical data

The data which we will process is a set of tables each of which is a description of a solution to a problem  $\mathcal{P}'$  close to the target problem  $\mathcal{P}$ , the one we seek to model.

**Definition 5 (Table)** A *table*  $T$  is a relation  $(A_1, \dots, A_k)$  where the  $A_i$  are the attributes of  $T$ .

The set of attributes of  $T$  is denoted its *scheme*, a subset of it is denoted a *partial scheme*.

**Definition 6 (History)** A *History* is a set of tables sharing the same scheme. We denote *h-solutions* (historical solutions) the tables that constitute the history.

**Definition 7 (Domain of the attributes)** The *domain* of an attribute is the set of values occurring in a *h-solution*, or in a set of *h-solutions*, for this attribute.

We denote  $D_h(A_i)$  the domain of the attribute  $A_i$  in the *h-solution*  $h$  and  $D_H(A_i)$  the domain of  $A_i$  on the history  $H$ .

We extend the notion of domain to a partial scheme as follows:

Let  $S = \{A_1, \dots, A_n\}$  be a partial scheme and  $h$  a *h-solution*,  $D_h(S) = D_h(A_1) \times \dots \times D_h(A_n)$

**Definition 8 (Term and partial terms)** We denote *terms* the entities of a relation and *partial terms* any subset of values included in a term.

**Definition 9 (h-solution)** A *h-solution*  $h$ , to a problem  $\mathcal{P}$  with a scheme  $s$  is a table with a scheme  $s$ .  $h\text{-Sol}_s(\mathcal{P})$  is a set a *h-solution* to the problem  $\mathcal{P}$  through the scheme  $s$  for which exists a bijection from  $\text{Sol}(\mathcal{P})$  in  $h\text{-Sol}_s(\mathcal{P})$ .

**Definition 10 (close problems)** Two Problems,  $\mathcal{P}_1$  and  $\mathcal{P}_2$ , are said close to each other iff exists a scheme of attributes  $s$  for which exists  $h\text{-Sol}_s(\mathcal{P}_1)$  and  $h\text{-Sol}_s(\mathcal{P}_2)$ .

Here, we have clearly defined the constraint programming elements we use and the vocabulary on the data we have to process.

### 2.3 The school time table an academic case

For a better understanding let's now see in detail a practical example of history. We use a simplified weekly school timetable for the years 2000 to 2003. The target problem here is to compute a valid timetable for the year 2004. There is one table (h-solution) for each year and each line (term) of a table describes one lesson through 4 attributes (Time, Subject, Class and Room).

**Time** describes the day and the hour when the lesson takes place. There are 5 possible days and 4 possible hours.

**Subject** indicates the kind of lesson. There are 7 subjects: Math, Science, English, French, Sport, Music and Art.

**Class** indicates the student group who follows the lesson. There are 4 classes which are, in growing age order: Class6, Class5, Class4 and Class3.

**Room** is the place where the lesson is given. There are 4 different places: Room1, Room 2, RoomTP (for science lessons) and Outside (for sport lessons).

Let,s see on this example some of the elements we have defined in the previous

<b>Sample history: A school timetable</b>							
h-solution 1: year 2000				h-solution 2: Year 2001			
Time	Subject	Class	Room	Time	Subject	Class	Room
Monday 8h	English	Class5	Room1	Monday 8h	Math	Class3	Room2
Monday 8h	Math	Class3	Room3	Monday 10h	Math	Class4	Room1
Monday 8h	Science	Class6	RoomTP	Monday 10h	English	Class3	Room2
Monday 10h	Math	Class4	Room1	Monday 14h	French	Class5	Room1
Monday 10h	English	Class6	Room2	Monday 14h	English	Class4	Room2
Monday 14h	Math	Class5	Room1	Monday 16h	English	Class5	Room2
Monday 14h	Science	Class3	Room2	Monday 16h	Sport	Class3	Outside
Monday 14h	English	Class4	Room3	Tuesday 8h	English	Class5	Room1
Monday 14h	Sport	Class6	Outside	Tuesday 8h	Math	Class3	Room2
Monday 16h	French	Class3	Room1	Tuesday 8h	Science	Class6	RoomTP
Tuesday 8h	English	Class4	Room1	Tuesday 10h	Math	Class4	Room1
Tuesday 8h	Math	Class6	Room3	Tuesday 10h	English	Class6	Room2
...	...	...	...	...	...	...	...
h-solution 3: Year2002				h-solution 4: Year 2003			
Time	Subject	Class	Room	Time	Subject	Class	Room
...	...	...	...	...	...	...	...

Fig. 1. An example of history

section.

*Sample target problem P:* "To design a valid timetable for the year 2004"

*Sample history H:* The school timetables for the year 2000 to 2003

*Sample h-solution h:* The timetable for the year 2001

*Sample attribute of h:* 'Subject'

*Sample attribute domain:*  $D_{2001}('Subject') = \{'Maths', 'Science', 'English', 'French', 'Music', 'Sport'\}$

*Sample partial scheme:*  $\{'Time', 'Room'\}$

*Sample domain of a partial scheme:*  $D_{2001}(\{'Time', 'Room'\}) = D_{2001}('Time') \times D_{2001}('Room')$

*Sample term:*  $\{'Monday\_8h', 'Science', 'Class6', 'Room\_TP'\}$

*Sample partial term:*  $\{'Science', 'Room\_TP'\}$

And we can assure that the timetable design problems for the years 2000 to 2003 ( $\mathcal{P}_{2000}$ ,  $\mathcal{P}_{2001}$ ,  $\mathcal{P}_{2002}$ ,  $\mathcal{P}_{2003}$  and  $\mathcal{P}_{2004}$ ) are close to each other because their solutions can all be expressed through tables with the scheme  $\{'Time', 'Subject', 'Class', 'Room'\}$ .

### 3 The approach

#### 3.1 Main principle and limitations

In this section we describe the formal base we rely on to build viewpoints for our target problem according to the history.

The history is a set of h-solutions to problems which are *close* to our target problem. We can build a viewpoint that can describe every h-solution of the history: such a viewpoint is said to *match* the history. It is important to note that we determine the domains of the viewpoints from the values occurring in the history: in order for the viewpoints generated to represent every solution of the target problem, we need the following hypothesis to be true.

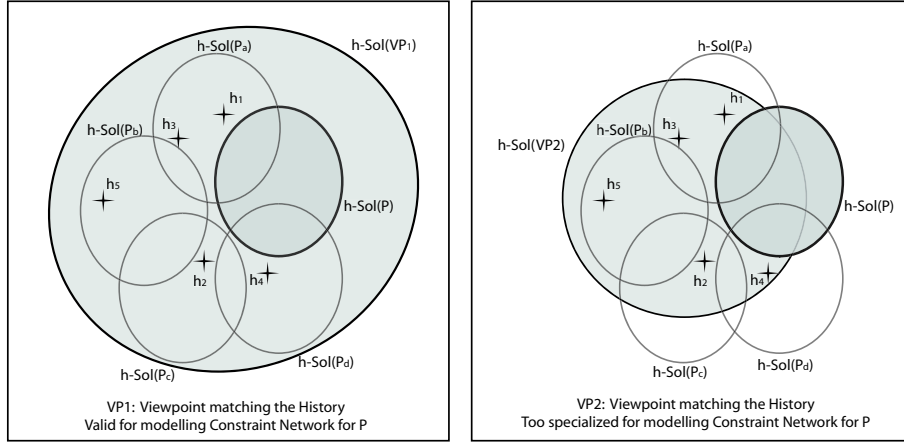
**Hypothesis 1 (Domain constancy)** *For each attribute of the h-solutions, any values present in any h-solution of the target problem must occur at least once in the history.*

*We can express this, for any h-solution  $h_{\mathcal{P}}$  to the target problem, as follows:*

$\forall h_{\mathcal{P}} \in h\text{-Sol}_s(\mathcal{P}),$  a h-solution of the the target problem,  $\forall A_i,$  an attribute of  $h_{\mathcal{P}}, D_{h_{\mathcal{P}}}(A_i) \subseteq D_H(A_i)$

In Fig.2,  $P_a, P_b, P_c$  and  $P_d$  are problems close to the target problem  $P$ .  $h_1$  to  $h_5$  are the h-solutions of the history.  $VP_1$  and  $VP_2$  are two different viewpoints matching the history. The left part of Fig.2 shows how a viewpoint matching the history can cover every solution of the target problem.

We cannot be sure that the viewpoint matching the history will be able to describe any solution of the target problem. This is illustrated in the right part of Fig.2, which highlights the fact that we need viewpoints general enough to include the solutions of the target problem. We show in section 3.3 how we select such general viewpoints. As any other learning technique, our method may fail for some viewpoints, but since we generate a lot of candidate viewpoints, a simple validation or refutation process with an expert of the target problem



**Fig. 2.** Viewpoints matching an history (general case)

would eliminate the too specialized viewpoints.

In this section, we have presented our approach and its limitations.

### 3.2 From partial terms to viewpoints

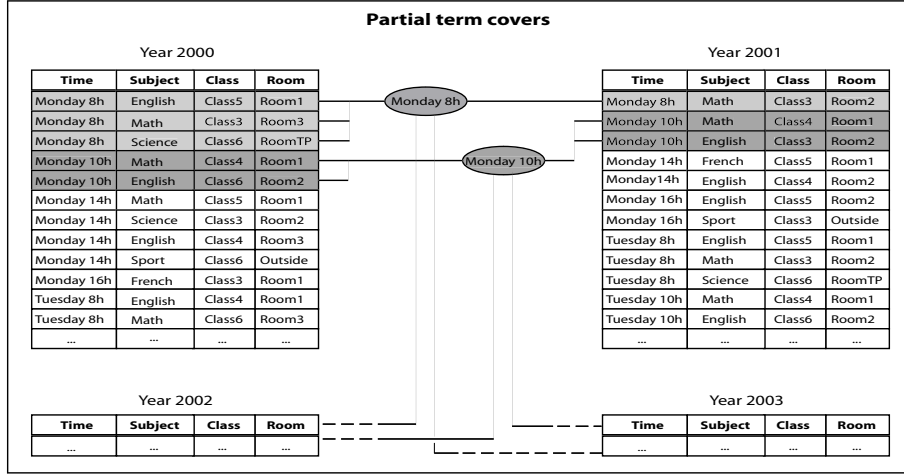
Our goal now is to build viewpoints matching a given history. To do so we determine *candidate variables* according to the history and then select a subset of them that constitute a viewpoint.

We use partial terms to generate *candidate variables* for the viewpoints we intend to build. To present this step we need to introduce the concept of cover of the partial terms.

**Definition 11 (Cover of a partial term)** *The cover  $C_h(p)$  of the partial term  $p$  in the  $h$ -solution  $h$  is the set of terms of  $h$  that contain the partial term  $p$ .*

**Definition 12 (Candidate variables associated to a partial term)** *The candidate variables associated to a partial term  $p$  occurring in  $h$  is a set of variables  $\mathcal{X}_{h,p} = \{X_{p,1}, \dots, X_{p,k}\}$  of variables where:  $|\mathcal{X}_{h,p}| = |C_h(p)|$ . The domain of these variables is the cartesian product of the domains of the attributes that are not valuated in  $p$ , plus the special value  $\varepsilon$ . The  $\varepsilon$  value used in an solution indicates that this variable should not to be taken into account in the reading of this solution.*

We also need to define a set of variables that covers a whole  $h$ -solution.



**Fig. 3.** Partial term cover on a history

**Definition 13 (Covering group on a h-solution)** Let  $P_h$  be a set of partial terms occurring in  $h$ ,  $P_h$  is a covering group on  $h$  iff:

$$\bigcup_{p \in P_H} C_h(p) = h$$

Now, with these elements we are able to describe any h-solution from a covering group, as presented in the following theorems.

**Theorem 1 (Building a viewpoint matching a h-solution)** The set of candidate variables associated to the partial terms of any covering group on a h-history  $h$  can be instantiated to represent  $h$ .

**Proof 1** Let  $P_h$  be a covering group on a h-solution  $h$ .

Let  $\mathcal{X}_{P_h}$  and  $\mathcal{D}_{P_h}$  be the sets of candidate variables and their domains built from the partial terms of  $P_h$ .

It comes from the definition of a covering group that there exists an application  $f$  such as:

$$\begin{aligned} h &\xrightarrow{f} \mathcal{X}_h, \text{ where } \mathcal{X}_h \subseteq \mathcal{X}_{P_h} \\ t &\xrightarrow{f} X_{p,i}, \text{ where } p \subset t \end{aligned}$$

According to the definition of the candidate variables:

$$t \xrightarrow{f} X_{p,i} \implies \exists v \in D(X_{p,i}), t = p \cup v.$$

So there exists an instance of  $\mathcal{X}_h$  that exactly maps  $h$ . And the remaining variables of  $\mathcal{X}_{P_h}$  can be set to the  $\varepsilon$  value so they will not be taken into account in the interpretation of the instance. This is why we can affirm that a viewpoint constituted from a covering group on  $h$  does match  $h$ .

**Theorem 2 (Building a viewpoint matching a history)** *The set of candidate variables associated to the partial terms of a set of covering groups, one per h-solution  $h$  of a history  $H$ , can be instantiated to represent any  $h$  of  $H$ .*

**Proof 2** *Let  $P_H = \{P_{h_1}, \dots, P_{h_k}\}$  be a set of covering groups, one on each h-solution  $h$ , in a history  $H$ .*

*Let  $\mathcal{X}_{P_H}$  and  $\mathcal{D}_{P_H}$  be the sets of candidate variables and their domains built from the partial terms of  $P_H$ .*

*$\forall h \in H, \exists P_h \in P_H$ , where  $P_h$  is a covering group on  $h$*

*It comes from Theorem 1 that  $\mathcal{X}_{P_H}$  can be instantiated to represent any  $h$  in  $H$ .*

*So if the viewpoint  $(\mathcal{X}_{P_H}, \mathcal{D}_{P_H})$  matches any  $h$  in  $H$ , it matches  $H$ .*

We have demonstrated here that finding a covering group on each  $h$  in  $H$  leads to a viewpoint that matches the history. Let's now see how to select general enough covering groups, whose viewpoints will represent the solutions of the target problem.

### 3.3 From constant partial terms to constant viewpoints

The number of covering groups for each h-solution is very large and the number of unions of these covering groups is far larger. Among the viewpoints derived from these unions, we focus on those which assure a minimum generalization level.

*Remark: For the simplicity of this article's remaining explanations all the h-solutions of our history have the same number of terms. This is not always true but this does not invalidate our approach and we will describe further on how to handle such cases.*

We focus on viewpoints that match the history and have the same covering group on each h-solution. Such viewpoints describe all the h-solutions with the same variables: we can say they are *general* to the whole history and we can expect that they are sufficiently general to allow the description of our target problem solutions as well.

To build such viewpoints, we need to find partial terms that occur a constant number of times in every h-solution of the history. If we can build *constant groups* with such terms, they will cover each h-solutions in the history. Let's now see in detail how to find these *constant groups*.

**Definition 14 (Constant partial term)** *A partial term  $p$  is said constant on a history  $H$  iff:  $\forall h_i, h_j \in H, C_{h_i}(p) \neq \emptyset$  and  $|C_{h_i}(p)| = |C_{h_j}(p)|$*

**Definition 15 (Constant group)** *A constant group is a set of partial terms covering every h-solution of the history and whose every element is a constant partial term.*

A constant group may still contain a lot of variables. In order to provide a usable viewpoint, we need to reduce the number of variables. Because each term of a h-solution is semantically atomic, the minimum number of variables we can reach is the number of terms in one h-solution. Note that the covers of the minimal constant group constitute, for each h-solution, a partition of it.

**Definition 16 (Minimal constant group)** *A minimal constant group  $P_H$  on a history  $H$  is minimal iff:*

$$\forall h \in H, \sum_{p \in P_H} |C_h(p)| = |h|$$

Note that for any history where all the h-solutions have the same number of terms, there exists one minimal constant group. This group,  $P_{H,\emptyset}$ , is built from the partial term  $\{\emptyset\}$ . Because  $\{\emptyset\}$  is included in all terms, we have  $|\mathcal{X}| = |h|$  for every  $h$  of  $H$ . So, the minimal constant group  $P_{H,\emptyset}$  allows to build a viewpoint that we call *the root viewpoint*.

**The root viewpoint:**

$\mathcal{X}_\emptyset = X_{\emptyset,1}, \dots, X_{\emptyset,|h|}$   
 $\forall X_{\emptyset,i} \in \mathcal{X}_\emptyset, D(X_{\emptyset,i}) = D(A_0) \times \dots \times D(A_k)$ , where  $\{A_0, \dots, A_k\}$  is the scheme of  $H$ .

As we can see, the domain of the variables of the root viewpoint is far too large to be efficiently solved with constraint programming. Moreover, it is really hard to put constraints on such a viewpoint in order to construct a constraint network that models the target problem.

Therefore, we must seek constant minimal viewpoints with more specialized variables and domains of smaller size. The bigger a partial term is, the more specialized the candidate variables built from it will be, and the smaller the size of their domains will be. This specialization is possible because the viewpoints integrate the cardinality constraints that are expressed in the history through the constancy of certain partial terms.

For the rest of the explanations, let's focus on a particular subset of the minimal constant groups: the *homogeneous constant groups*.

**Definition 17 (Homogeneous constant group)** *A constant group  $P_H$  on a history  $H$  in which every partial term is associated to the same partial scheme  $s$  is said homogeneous.*

Note that the covers of a homogeneous constant group constitute a partition of  $h$  because two partial terms of such a group cannot belong to the same term. A homogeneous constant group is therefore always minimal.

*Remark: h-solutions with different sizes*  
*In order to simplify the explanation we have chosen to ignore the case of histories where two distinct h-solutions don't have the same number of terms. To manage*

these histories, we add special terms to the smaller  $h$ -solutions until all the  $h$ -solutions have the same size. These special terms contain only  $\varepsilon$  values, which can be interpreted as any value of the corresponding domain. With this modification, our viewpoint modeling approach leads to satisfying results. At the instantiation, we only have to set some of our viewpoint's variables to the  $\varepsilon$  value. Therefore we can ensure that the generated viewpoints match the history.

## 4 Algorithm

### 4.1 Pruning techniques

As we have seen, the first step of the viewpoint building is to find constant partial terms. In order to do so we study the partial terms occurring in our history in growing size order. The lattice of inclusion between the schemes (Fig.4), which contains the partial term inclusion lattices, gives us a way to avoid useless searches by exploiting the following partial term inclusion property.

**Property 1** According to the definition of the cover of a partial term:

$$\forall h \in H, p' \subset p \implies C_h(p) \subset C_h(p')$$

So if  $\exists h \in H, |C_h(p')| = 0$

$|C_h(p)| = 0$  (i.e  $p$  cannot be constant).

Therefore we check partial terms in growing size order, eliminating the ones that cannot be constant according to Property 1.

The algorithm 1, *Extraction of constant partial terms*, presents this method.

### 4.2 Algorithmic complexity

Let's now determine the worst-case complexity of this algorithm.

At each level of the scheme inclusion lattice, we merge partial terms associated to different schemes. We have  $a$  attributes in  $H$  so there are  $a$  levels in the lattice. There are at most  $|h|$  different partial terms associated to each pattern and at most  $\binom{a}{\frac{a}{2}}$  schemes in a given level of the lattice.

So the worst case complexity of our algorithm is:

$$O(a \cdot \binom{a}{\frac{a}{2}} \cdot h^2) = O\left(\binom{a}{\frac{a}{2}}\right) = O(a!)$$

The worst-case complexity is far from the average complexity because it does not take the pruning into account. Moreover, in the most extreme case, all the  $h$ -solutions contain exactly the same terms, all different from each other. This is obviously not a realistic case.

Because we maintain information on previous partial terms during the algorithm execution, the space complexity is of the same order as the time complexity. And the space complexity is reduced by the pruning in the same extent as the time complexity.

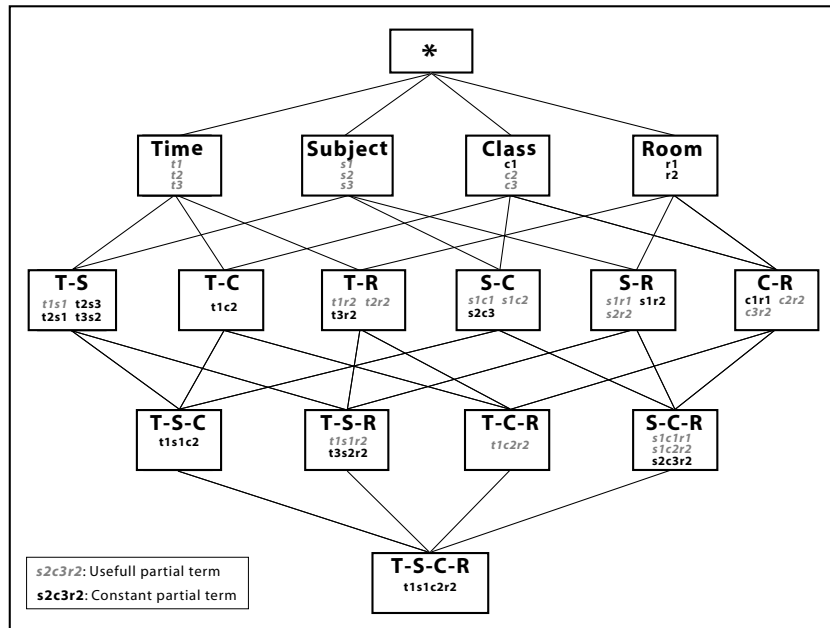


Fig. 4. Scheme inclusion lattice

## 5 Experimentations

We applied our technique for extracting constraint homogeneous viewpoints on three typical problems: the 5-queen problem, the sudoku and a simplified school timetable design. We found several constant homogeneous viewpoints for each problem. In the following result presentations, we ignore the root viewpoint which is always present and of very little interest for us.

### 5.1 5-queen problem

Here, the history is constituted of 8 valid solutions to the 5-queen problem. The h-solutions are 5-row tables, each of which indicates the column and row indexes of one of the queens.

#### Results

Our tool produces two viewpoints.

```
ViewPoint 1 : Partial scheme = Line; |X| = 5, |D| = 5
--- X_Line1 --- D:Column --- |D|=5
--- X_Line2 --- D:Column --- |D|=5
--- X_Line3 --- D:Column --- |D|=5
--- X_Line4 --- D:Column --- |D|=5
```

**Données:**  $H$  an history containing  $k$  h-solution  $h_i$ ,  $T_1$  the set of partial terms of size 1 taken from the history

**Résultat:** Every constant partial term on  $H$

```

begin
   $P \leftarrow \phi$  /* Usefull partial terms */
   $P_{cst} \leftarrow \phi$  /* Constant partial terms */
  for each  $p$  in  $T_1$  do
     $minCov \leftarrow \text{Min}(|C_h(p)|, h \in H)$ 
     $maxCov \leftarrow \text{Max}(|C_h(p)|, h \in H)$ 
     $C_H(p) \leftarrow C_{h_1}(p) \cup \dots \cup C_{h_k}(p)$ 
    if  $minCov > 0$  then
       $P \leftarrow P \cup p$ 
      if  $maxCov = minCov$  then
         $P_{cst} \leftarrow P_{cst} \cup p$ 
  for  $i$  from 1 to  $|A|-1$  do
    for each  $(p_1, p_2)$  from  $P$  where  $|p_1| = i$  and  $|p_2| = i$  do
      if  $(|p_1 \cup p_2| = i + 1)$  and  $(p_1 \cup p_2 \notin P)$  then
         $p \leftarrow p_1 \cup p_2$ 
         $minCov \leftarrow \text{Min}_{h \in H} |C_h(p_1) \cap C_h(p_2)|$ 
         $maxCov \leftarrow \text{Max}_{h \in H} |C_h(p_1) \cap C_h(p_2)|$ 
         $C_H(p) \leftarrow (C_{h_1}(p_1) \cap C_{h_1}(p_2)) \cup \dots \cup (C_{h_k}(p_1) \cap C_{h_k}(p_2))$ 
        if  $minCov > 0$  then
           $P \leftarrow P \cup p$ 
          if  $maxCov = minCov$  then
             $P_{cst} \leftarrow P_{cst} \cup p$ 
  return  $P_{cst}$ 
end

```

**Algorithm 1:** Extraction of constant partial terms

--- X\_Line5 --- D:Column --- |D|=5

ViewPoint 2 : Partial scheme = Column; |X| = 5, |D| = 5

--- X\_Column1 --- D:Line --- |D|=5

--- .....

--- X\_Column5 --- D:Line --- |D|=5

The first viewpoint is constituted of 5 variables: one for the queen in the first line, one for the queen in the second line, and so on. The domain of every variable is the set of integers from 1 to 5, this number indicates the column where the queen is.

The second viewpoint is the exact symmetric of the first, with one variable for each column's queen. The values indicate the the line where the queen is.

These two viewpoints integrate respectively the facts that there is always exactly one queen on each line and one queen on each column.

## 5.2 Sudoku

We start here from ten valid, already filled sudoku grids. To each cell of a grid is associated a term that gives the line index, the column index and the value of the cell.

### Results

```
ViewPoint 1 : Partial pattern = Col; |X| = 81, |D| = 81
--- X_Col1_1 --- D:Line/Val --- |D|=81
--- X_Col1_2 --- D:Line/Val --- |D|=81
--- .....
--- X_Col9_9 --- D:Line/Val --- |D|=81
```

```
ViewPoint 2 : Partial pattern = Line; |X| = 81, |D| = 811
--- X_Line1_1 --- D:Col/Val --- |D|=81
--- X_Line1_2 --- D:Col/Val --- |D|=81
--- .....
--- X_Line9_9 --- D:Col/Val --- |D|=81
```

```
ViewPoint 3 : Partial pattern = Val; |X| = 81, |D| = 81
--- X_Col1_1 --- D:Line/Col --- |D|=81
--- X_Val1_2 --- D:Line/Col --- |D|=81
--- .....
--- X_Val9_9 --- D:Line/Col --- |D|=81
```

These first 3 viewpoints are respectively based on the fact that there are exactly 9 values occurring exactly 9 times in each h-solution, 9 column indexes present 9 times and 9 line indexes present 9 times.

```
ViewPoint 4 : Partial pattern = Line-Col; |X| = 81, |D| = 9
--- X_Line1-Col1 --- D:Val --- |D|=9
--- X_Line1-Col2 --- D:Val --- |D|=9
--- .....
--- X_Line9-Col9 --- D:Val --- |D|=9
```

```
ViewPoint 5 : Partial pattern = Line-Val; |X| = 81, |D| = 9
--- X_Line1-Val1 --- D:Col --- |D|=9
--- X_Line1-Val2 --- D:Col --- |D|=9
--- .....
--- X_Line9-Val9 --- D:Col --- |D|=9
```

```
ViewPoint 6 : Partial pattern = Col-Val; |X| = 81, |D| = 9
--- X_Col1-Val1 --- D:Line --- |D|=9
--- X_Col1-Val2 --- D:Line --- |D|=9
--- .....
--- X_Col9-Val9 --- D:Line --- |D|=9
```

The viewpoint 4 is based on the fact that there is only one value from 1 to 9 for one cell of the grid.

The viewpoint 5 is based on the fact that every line must be filled with, for each cell, one different value from 1 to 9.

The viewpoint 6 is based on the fact that every column must be filled with, for each cell, one different value from 1 to 9.

These three viewpoints are interesting because they express some aspects of the sudoku problem's structure. The viewpoint 4 gives us fundamental information on how to fill the grid. The viewpoints 5 and 6 integrate the 'all diff' constraint existing respectively on column values and line values.

### 5.3 School timetable

From our four-year history of school timetable, presented in section 2.3, we automatically discovered 3 viewpoints.

#### Results

```
ViewPoint 1 : Partial pattern = Class; |X| = 46, |D| = 882
--- X_Class3_1 --- D:Time/Room/Subject --- |D|=882
--- X_Class3_2 --- D:Time/Room/Subject --- |D|=882
--- .....
--- X_Class6_9 --- D:Time/Room/Subject --- |D|=882
```

This viewpoint is based on the fact that each class had the same number of courses during the past four years. So we have 13 lessons for Class 3, 12 for Class 4, 12 for Class 5 and 9 for Class 6. This viewpoint is valid but not usable because the domain of each variable is too large.

```
ViewPoint 2 : Partial pattern = Subject; |X| = 46, |D| = 504
--- X_Math_1 --- D:Time/Room/Class --- |D|=504
--- X_Math_2 --- D:Time/Room/Class --- |D|=504
--- .....
--- X_Math_12 --- D:Time/Room/Class --- |D|=504
--- .....
--- X_Musique_1 --- D:Time/Room/Class --- |D|=504
--- X_Musique_2 --- D:Time/Room/Class --- |D|=504
```

This viewpoint takes advantage of the constant number of lessons on a specific subject to associate a variable to each different lesson on a subject.

We have here an interesting viewpoint but the domain size is still too big to build an efficient constraint network with it.

```
ViewPoint 3 : Partial pattern = Class-Subject; |X| = 46, |D| = 126
--- X_Class3-Science_1 --- D:Time/Room --- |D|=126
--- X_Class3-Science_2 --- D:Time/Room --- |D|=126
--- X_Class3-Science_3 --- D:Time/Room --- |D|=126
```

```
--- X_Class3-Sport --- D:Time/Room --- |D|=126
--- X_Class3-English_1 --- D:Time/Room --- |D|=126
--- .....
--- X_Class6-Art --- D:Time/Room --- |D|=126
```

The viewpoint building process has finally discovered that each class had the same number of lessons in a specific subject in the past four years. This last viewpoint associates a variable to each lesson of a specific class on a specific subject. This model looks quite interesting for building a usable constraint network.

These experimental results validate our method by providing several usable viewpoints for each problem. A relevant selection among these automatically generated viewpoints is the next step of our work.

## 6 Conclusions

In this paper, we have proposed the very first step of a method to get rid of the bottleneck of modelling in the CSP approach to Problem Solving.

This automatic modelling is based on a set of historical data which describe solutions to problems close to our target problem. We have focused the first step of this modelling on the determination of the variables and their domains, producing constraint viewpoints which are able to describe all of the target problem solutions.

To do so, we have looked for viewpoints that can represent any of the solutions of the history. We have then selected among them the viewpoints general enough to represent any solution to problems of the same kind (such as the target problem). Using our tool, which can easily build a particular subset of these viewpoints, we have produced viewpoints for three different simple problems.

Our method can be combined with Machine Learning approaches which learn constraints from examples, once the variables and the domains are given. We are now investigating in this direction.

## References

1. Alan M. Frisch, Christopher Jefferson, Bernadette Martínez Hernández, and Ian Miguel. The rules of constraint modelling. In *IJCAI*, pages 109–116, 2005.
2. Remi Coletta, Christian Bessière, Barry O’Sullivan, Eugene C. Freuder, Sarah O’Connell, and Joël Quinqueton. Semi-automatic modeling by constraint acquisition. In *CP*, pages 812–816, 2003.
3. Christian Bessière, Remi Coletta, Frédéric Koriche, and Barry O’Sullivan. A sat-based version space algorithm for acquiring constraint satisfaction problems. In *ECML*, pages 23–34, 2005.
4. P. A. Geelen. Dual viewpoint heuristics for binary constraint satisfaction problems. In *ECAI*, pages 31–35, 1992.
5. Yat Chiu Law and Jimmy Ho-Man Lee. Model induction: A new source of csp model redundancy. In *AAAI/IAAI*, pages 54–, 2002.

6. B. M. W. Cheng, Kenneth M. F. Choi, Jimmy Ho-Man Lee, and J. C. K. Wu. Increasing constraint propagation by redundant modeling: an experience report. *Constraints*, 4(2):167–192, 1999.
7. Barbara M. Smith. Dual models of permutation problems. In *CP*, pages 615–619, 2001.
8. Toby Walsh. Permutation problems and channelling constraints. In *LPAR*, pages 377–391, 2001.