A New Incomplete Method for CSP Inconsistency Checking

Belaïd Benhamou and **Mohamed Réda Saïdi**
Laboratoire des Sciences de l'Information et des Systèmes (LSIS)
Centre de Mathématiques et d'Informatique
39, rue Joliot Curie - 13453 Marseille cedex 13, France
email:{Belaid.Benhamou;saidi}@cmi.univ-mrs.fr

Abstract

Checking CSP consistency is shown, in theory, to be an NP-complete problem. There is two families of methods for CSP consistency checking. The first family holds the complete methods which make an exhaustive search on the solution space. These methods have the advantage to prove CSP inconsistency, but their complexity grows exponentially when the problem size increases. The second family includes the incomplete methods that make a local search on the solution space. These methods have been efficiently used to find solutions for large size consistent CSPs that complete methods can not solve. One major drawback of the incomplete methods, is their inability to prove CSP inconsistency. One of the challenges that have been put forward by the CP community (Selman et al. 1997) is to provide incomplete methods that can deal with CSP inconsistency efficiently. The work that we present here, is a contribution towards an answer to this hard challenge. We introduce a new incomplete method for CSP inconsistency checking that is based on both a new notion of dominance between CSPs and a coloration of the CSP micro-structure. We experimented the method on randomly generated CSP instances and the results obtained are very promising.

Introduction

The work that we present here is a contribution to solve the challenge 5 given in (Selman, Kautz, & McAllester 1997). Challenge 5 consists in exploiting incomplete methods to prove efficiently CSP inconsistency. The motivation in solving this challenge, comes from the advances made on CSP and SAT solvers. Indeed, several large size problems that are known to be hard for complete methods like Davis-Putnam-Logemann-Loveland (Davis, Logemann, & Loveland 1962) or its improvements, had been solved efficiently by using incomplete methods (Selman, Levesque, & Mitchell 1992; Hoos & Stützle 2004) that are based on a local search in the solution space. However, local search methods can not be used to prove CSP inconsistency and lose a great part of their usefulness in CSP Solving.

Although Challenge 5 is known since 1997, the are only few works that had been done on this subject: we can find some works that use incomplete approaches to prove unsatisfiability of propositional formulas in (Klerk, Maaren, & Warners 2000; Goldberg 2002; Prestwich & Lynce 2006; Aloul, Lynce, & Prestwich 2007; Audemard & Simon 2007) and mainly two other works in CSPs (Gaur, Jackson, & Havens 1997; Bès & Jégou 2005) that attempt to prove CSP inconsistency with incomplete approaches.

We propose in this work a new incomplete method for binary CSP inconsistency checking. This method is based on a new notion that we call *CSP dominance* and on a coloration of the CSP micro-structure. We will prove that the known coloration approach used in (Gaur, Jackson, & Havens 1997) is a particular case of our method. We implemented the new method and experimented it on several random inconsistent CSP instances, and the results obtained are very promising and show that our method outperforms all the other tested methods.

This paper is organized as follows: CSP background is given in the second section. We introduce in the third section the basis of the new method that we propose in this work and show that the approach used in (Gaur, Jackson, & Havens 1997) is a particular case of our approach. We evaluate the proposed method in the fourth section where we can see the experiment results obtained on random CSP instances and a comparison of our method with some other methods. We conclude the work in last section.

Background

A CSP is a structure $P(n) = (V, D, C)$ where: $V = \{v_1, ..., v_n\}$ is a set of n variables; $D = \{D_1, ..., D_n\}$ is the set of finite discrete domains associated to the CSP variables, D_i includes the set of possible values of the CSP variable v_i, $C = \{C_1, ..., C_m\}$ is a set of m constraints each involving some subsets of the CSP variables. The constraints are given in their extension form, each constraint $C_i \in C$ is represented by the list of its permitted value tuples. A binary constraint is a constraint which involves at most two variables. A binary CSP $P(n)$ (a CSP involving only binary constraints) can be represented by a constraint graph $G(V, E)$ where the set of vertices V is the set of the CSP variables and each edge $(v_i, v_j) \in E$ connects the variables v_i and v_j involved in the constraint $C_{ij} \in C$. The micro-structure (Jégou 1993) of a binary CSP $P(n)$ is a graph $M_{P(n)}(V \times D, \acute{E})$, where each edge of $\acute{E}$ corresponds either to a tuple allowed by a specific constraint or to an allowed tuple because there is no constraint between the associated variables.

An *Alldifferent* constraint is a global constraint that forces all the variables in its scope to take different values. $Alldifferent$ is a well studied global constraint, several efficient dedicated algorithm are known in the literature.

An instantiation $I = (\langle v_1, d_1\rangle, \langle v_2, d_2\rangle, \ldots, \langle v_n, d_n\rangle)$ is the variable assignment $\{v_1 = d_1, v_2 = d_2, \ldots, v_n = d_n\}$ where a value d_i of the domain D_i is assigned to the variable v_i. The instantiation I is consistent if it satisfies all the constraints of C, thus I is a solution of the CSP.

The Dominance-Coloration method

Before describing the theoretical basis of our method, we recall the principle of the method introduced in (Gaur, Jackson, & Havens 1997).

The main theoretical result on which the method is based is given in the following corollary.

Corollary 1 *(Gaur, Jackson, & Havens 1997) If the micro-structure $\mathcal{M}_{P(n)}$ of a CSP $P(n)$ having n variables, can be colored with $n-1$ colors, then the CSP is inconsistent.*

To illustrate how this corollary can be used in a CSP inconsistency proof, we take an instance of the pigeon-holes problem with three pigeons and two holes that we denote by $Pigeons(3)$. The problem consists in putting the pigeons in the holes, in such way that each hole holds at most one pigeon. Figure 1 gives the constraint graph of the CSP representing $Pigeons(3)$ in the left part and its micro-structure in the right part. The CSP variables v_i, $i \in \{1,2,3\}$ represent the pigeons and the values a and b express the holes.

The micro-structure of $Pigeons(3)$ admits a 2-coloration. Indeed, it is trivial to see that the vertices $\langle v_1, a\rangle$, $\langle v_2, a\rangle$ and $\langle v_3, a\rangle$ can be colored with one color, say color 1 for instance, and the other vertices $\langle v_1, b\rangle$, $\langle v_2, b\rangle$ and $\langle v_3, b\rangle$ are then colored with the color 2. Now, by application of the result of Corollary 1, we can deduce that the problem is inconsistent, since there is no clique of size 3 (a 3-clique).

For a CSP $P(n)$ having n variables, the authors of (Gaur, Jackson, & Havens 1997) try to find a $(n-1)$-coloration of its micro-structure to show its inconsistency, we call this method the coloration method.

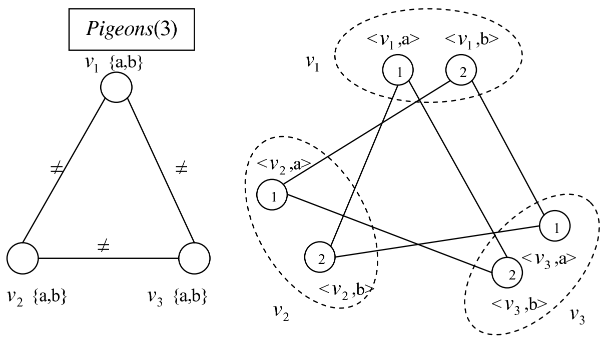


Figure 1: $Pigeons(3)$: pigeon-holes instance

The basis of the Dominance-Coloration method

Now, we define the notion of dominance between CSPs which is one of the two fundamental notions on which the method is based.

Definition 1 *If $P(n)$ and $P'(n)$ are two CSPs having n variables, then $P'(n)$ dominates $P(n)$, if only if, there exists a mapping that matches each solution of $P(n)$ with a solution of $P'(n)$.*

From the previous definition we understand that a CSP $P'(n)$ dominates another CSP $P(n)$ iff each solution of $P(n)$ can be mapped to a solution of $P'(n)$. The interaction between dominance and CSP consistency is given in the following proposition.

Proposition 1 *If an inconsistent CSP $P'(n)$ dominates another CSP $P(n)$, then $P(n)$ is inconsistent.*

Proof **1** *The proof can be easily derived from Definition 1.*

We define now the k-coloration notion which is the second basic element of our method.

Definition 2 *Let us given a CSP $P(n)$ and its micro-structure $\mathcal{M}_{P(n)}$. A k-coloration of $\mathcal{M}_{P(n)}$, is a mapping f_k that associates to each vertex of $\mathcal{M}_{P(n)}$, a color represented by an integer j such that $1 \le j \le k$ and the colors of adjacent vertices must be different.*

Now, we show how to associate to a given CSP $P(n)$ and a given k-coloration f_k of its micro-structure $\mathcal{M}_{P(n)}$, a CSP Alldiff($P(n)$,f_k) expressing a global *Alldifferent* constraint between its variables and we will prove that the Alldifferent CSP dominates $P(n)$.

Definition 3 *Let $P(n)=(V, D, C)$ be a CSP having n variables, $\mathcal{M}_{P(n)}$ its micro-structure and a k-coloration f_k of $\mathcal{M}_{P(n)}$. We associate to $P(n)$ and f_k, the CSP Alldiff($P(n), f_k$)=(V', D', C') where:*

- *$V' = \{v'_1, v'_2, ..., v'_n\}$, we associate a variable v'_i to each variable v_i of $P(n)$;*
- *$D' = \{D'_1, D'_2, ..., D'_n\}$, where each domain D'_i is defined as follows: D'_i=$\{f_k(\langle v_i, d_i\rangle)/d_i \in D_i\}$; in other words D'_i includes all the colors involved in f_k when coloring the vertices $\langle v_i, d_i\rangle$, $\forall d_i \in D_i$;*
- *C'=$\{Alldifferent(v'_1, v'_2, ..., v'_n)\}$, that is, C' is formed by the single global Alldifferent constraint $Alldifferent(v'_1, v'_2, ..., v'_n)$ defined on the variables of V'.*

Example 1 *Take the pigeon-hole CSP of Figure 1 (Left part) and the 2-coloration f_2 applied to color its micro-structure (the right part). We associate to the pigeon-hole CSP $Pigeons(3)$, the CSP Alldiff($Pigeons(3), f_2$)=(V', D', C') where:*

- *V'=$\{v'_1, v'_2, v'_3\}$, represents the variables associated with the variables $\{v_1, v_2, v_3\}$ of the the initial CSP $Pigeons(3)$;*
- *D'=$\{D'_1, D'_2, D'_3\}$, where D'_i=$\{f_2(\langle v_i, a\rangle), f_2(\langle v_i, b\rangle)\}$= $\{1, 2\}$, $\forall i \in \{1, 2, 3\}$;*
- *C'=$\{Alldifferent(v'_1, v'_2, v'_3)\}$.*

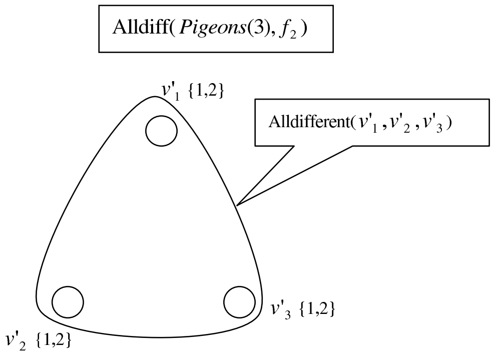

Alldiff(Pigeons(3), f_2)
v'_1 {1,2}
Alldifferent(v'_1 , v'_2 , v'_3)
v'_3 {1,2}
v'_2 {1,2}

Figure 2: The Alldiff($Pigeons(3), f_2$) associated with the pigeon-holes CSP $Pigeons(3)$ and the 2-coloration f_2.

The constraint graph of the CSP Alldiff($Pigeons(3), f_2$), is given in Figure 2.

Now, we introduce the key property of this work. The main results is derived from the dominance relation existing between $P(n)$ and the CSP Alldiff($P(n), f_k$) associated with the CSP $P(n)$ and the k-coloration f_k. We have the following theorem.

Theorem 1 *If $P(n)$ is a CSP having n variables, and f_k a k-coloration of its micro-structure $\mathcal{M}_{P(n)}$, then the associated CSP Alldiff($P(n), f_k$) dominates the CSP $P(n)$.*

***Proof* 2** *We shall prove that, each solution I of the CSP $P(n)$ can be mapped to a solution I' of the CSP Alldiff($P(n), f_k$). Let I=($\langle v_1, d_1\rangle, \langle v_2, d_2\rangle, ..., \langle v_n, d_n\rangle$) be an instantiation of $P(n)$, and consider the mapping $\Gamma : \prod_{i=1}^{n}(\{v_i\} \times D_i) \longrightarrow \prod_{i=1}^{n}(\{v'_i\} \times D'_i)$, such that $\Gamma(I)$=($\langle v'_1, f_k(\langle v_1, d_1\rangle)\rangle, \langle v'_2, f_k(\langle v_2, d_2\rangle)\rangle, ...,$ $\langle v'_n, f_k(\langle v_n, d_n\rangle)\rangle$). Now, suppose that I is a solution of $P(n)$, then we deduce that the vertices in the set $S = \{\langle v_1, d_1\rangle, \langle v_2, d_2\rangle, ..., \langle v_n, d_n\rangle\}$ form a clique of size n of the micro-structure $\mathcal{M}_{P(n)}$. Because, f_k is a k-coloration of $\mathcal{M}_{P(n)}$, then the vertices of the n-clique have two by two different colors. That is, $\forall$ $\langle v_i, d_i\rangle$, $\langle v_j, d_j\rangle$ $\in S$ such that $i \neq j$, we have $f_k(\langle v_i, d_i\rangle) \neq f_k(\langle v_j, d_j\rangle)$. Finally, $\Gamma(I)$=($\langle v'_1, f_k(\langle v_1, d_1\rangle)\rangle, \langle v'_2, f_k(\langle v_2, d_2\rangle)\rangle, ...,$ $\langle v'_n, f_k(\langle v_n, d_n\rangle)\rangle$)=$I'$ is a solution of the CSP Alldiff($P(n), f_k$).*

Theorem 1 holds an important result. We showed that the CSP Alldiff($P(n), f_k$) associated with $P(n)$ and f_k dominates the CSP $P(n)$. On the other hand, the CSP Alldiff($P(n), f_k$) has a single global *Alldifferent* constraint that is well studied in CP community and for which several good algorithms exist. One of the most known algorithms for *Alldifferent* constraint, is the one of Régin (Régin 1994) which is based on maximum matching search in bipartite graphs. This algorithm performs a generalized arc consistency (GAC) on global Alldifferent constraints in polynomial time complexity. The GAC algorithm is complete for inconsistency checking of *Alldifferent* CSPs and is efficient in practice. We use it to check the consistency of the CSP Alldiff($P(n), f_k$) derived from $P(n)$ and f_k. If the CSP Alldiff($P(n), f_k$) is shown to be inconsistent, then we deduce from Proposition 1 that the CSP $P(n)$ is inconsistent.

A necessary condition for the CSP Alldiff($P(n), f_k$) to be consistent is that, the number of different values in $\cup_{i=1,n} D'_i$ must at least equal n (i.e. $\mid \cup_{i=1,n} D'_i \mid \geq n$). Otherwise, there will be fewer values than variables, then the global *Alldifferent* becomes inconsistent.

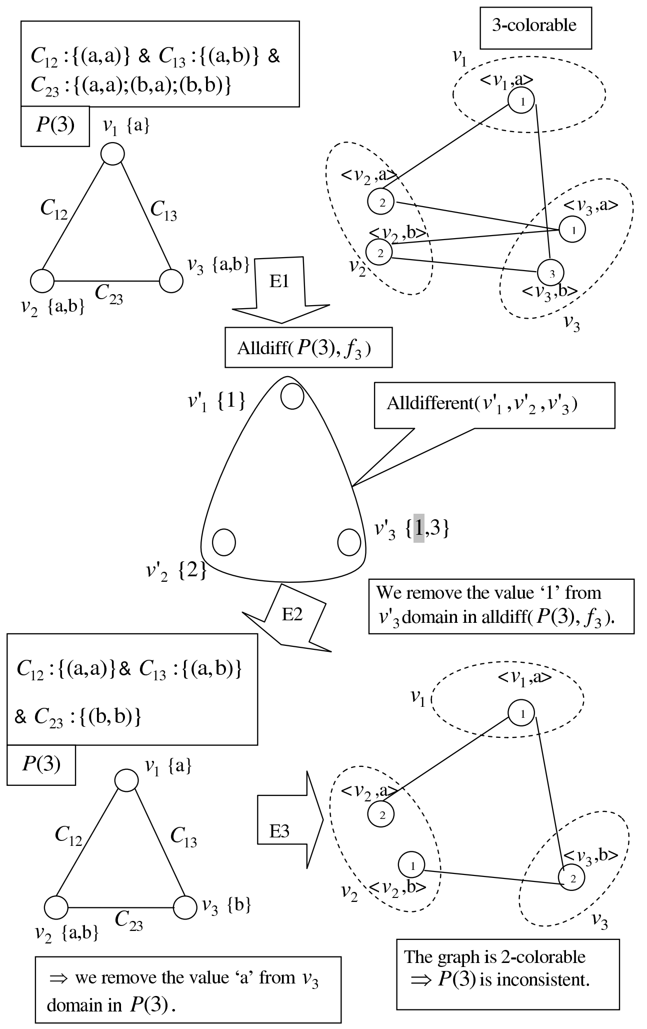
Remark 1 *Since the domain values of Alldiff($P(n), f_k$) are the colors used by f_k to color $\mathcal{M}_{P(n)}$, then the CSP inconsistency detected by the color method given in (Gaur, Jackson, & Havens 1997) is a particular case of inconsistency of the CSP Alldiff($P(n), f_k$) corresponding to the trivial case where Alldiff($P(n), f_k$) contains less values than variables which is included in our method. Therefore our method includes the color method (Gaur, Jackson, & Havens 1997) and will detect more inconsistencies.*

Theorem 1 states that when the CSP Alldiff($P(n), f_k$) is inconsistent, $P(n)$ is inconsistent too. However if we run GAC on Alldiff($P(n), f_k$) and the CSP remains consistent, then we can not deduce any information on the consistency of $P(n)$, but, some values that do not participate in any solution in Alldiff($P(n), f_k$) could be filtered by GAC during the consistency checking. We show in the following that we can use such values to simplify the CSP $P(n)$ by removing some corresponding values that do not participate in any solution of $P(n)$. We have the following proposition.

Proposition 2 *Let $P(n)$=(V, D, C) be a CSP, f_k a k-coloration of its micro-structure, Alldiff($P(n)$,f_k)=(V', D', C') the Alldiff CSP associated to $P(n)$ and f_k, and $v'_i \in V'$ the variable associated with $v_i \in V$. If there exists a value $d'_i \in D'_i$ of the variable v'_i, such that d'_i does not participate in any solution of Alldiff($P(n), f_k$), then each value $d_i \in D_i$ of the variable v_i verifying $f_k(\langle v_i, d_i\rangle) = d'_i$ does not participate in any solution of $P(n)$.*

***Proof* 3** *By hypothesis, the CSP Alldiff($P(n), f_k$) dominates the CSP $P(n)$. Thus, each solution I=($\langle v_1, d_1\rangle, \langle v_2, d_2\rangle, ..., \langle v_n, d_n\rangle$) can be transformed by the mapping $\Gamma : \prod_{i=1}^{n}(\{v_i\} \times D_i) \longrightarrow \prod_{i=1}^{n}(\{v'_i\} \times D'_i)$, to a solution $\Gamma(I)$=($\langle v'_1, f_k(\langle v_1, d_1\rangle)\rangle, \langle v'_2, f_k(\langle v_2, d_2\rangle)\rangle, ...,$ $\langle v'_n, f_k(\langle v_n, d_n\rangle)\rangle$) of the CSP Alldiff($P(n), f_k$). Therefore, if d_i participates in a solution of $P(n)$, then $f_k(\langle v_i, d_i\rangle) = d'_i$ will participate in a solution of Alldiff($P(n), f_k$). Thus, conversely if the value $f_k(\langle v_i, d_i\rangle) = d'_i$ does not participate in any solution of Alldiff($P(n), f_k$), then the value d_i of v_i does not participate in any solution of $P(n)$.*

Example 2 *Consider the CSP $P(3)$ of Figure 3. Its micro-structure admits a 3-coloration f_3. The coloration method (Gaur, Jackson, & Havens 1997) is not pertinent here, and can not prove the inconsistency of $P(3)$. On other hand, the CSP Alldiff($P(3), f_3$) associated to $P(3)$ and the 3-coloration, obtained at the step (E1) is consistent. We can deduce nothing on the consistency of $P(3)$. However the value 1 of the domain of the variable v'_3 participate in no solution of the CSP Alldiff($P(3), f_3$), thus it is suppressed by the GAC filtering. Now by applying Proposition 2, we* *deduce that the value a of the variable v_3 does not participate in any solution of $P(3)$, thus it is removed. This leads to a simplification of the CSP $P(3)$ (step E2). Now take the simplified CSP and try to find a new coloration for its micro-structure (step E3). We obtain a 2-coloration which means that the CSP is inconsistent.*



C_{12} : {(a,a)} & C_{13} : {(a,b)} & C_{23} : {(a,a);(b,a);(b,b)}
$P(3)$
v_1 {a}
C_{12}
C_{13}
C_{23}
v_2 {a,b}
v_3 {a,b}
3-colorable
v_1
<v_1,a>
<v_2,a>
<v_2,b>
v_2
<v_3,a>
<v_3,b>
v_3
1
2
3
E1
Alldiff($P(3), f_3$)
v'_1 {1}
Alldifferent(v'_1, v'_2, v'_3)
v'_3 {1,3}
v'_2 {2}
E2
We remove the value '1' from v'_3 domain in alldiff($P(3), f_3$).
C_{12} : {(a,a)} & C_{13} : {(a,b)} & C_{23} : {(b,b)}
$P(3)$
v_1 {a}
v_2 {a,b}
v_3 {b}
E3
<v_3,b>
The graph is 2-colorable ⇒ $P(3)$ is inconsistent.
⇒ we remove the value 'a' from v_3 domain in $P(3)$.

Figure 3: An example of inconsistency detection after value filtering

The Dominance-Coloration algorithm

The previous example gives an intuition of an incremental method to check CSP inconsistency by using the result of Proposition 2 and Theorem 1. Indeed, by iterating several times CSP simplification and coloration we can show the inconsistency of a CSP. The baseline method is sketched in Algorithm 1.

The principle of this algorithm consists in repeating the following main operations until the inconsistency is shown or the number of fixed steps is reached.

1. Use some incomplete graph coloring algorithm (here we use DSATUR (Brélaz 1979)) to compute a k-coloration f_k of the micro-structure G=$\mathcal{M}_{P(n)}$ (line 5). Notice that the inconsistency check of (Gaur, Jackson, & Havens 1997) is made in lines 6 and 7. That is, if a k-coloration such that $k < n$ is found, then the CSP is inconsistent.
2. Otherwise, derive the CSP $P'(n)$=Alldiff($P(n), f_k$) from $P(n)$ and f_k (line 9).
3. Apply the general arc consistency method (GAC) on $P'(n)$=Alldiff($P(n), f_k$) (line 10), and then get in return, either a proof of the inconsistency of the CSP $P'(n)$=Alldiff($P(n), f_k$) (line 11), thus $P(n)$ is inconsistent, or a simplified form $P''(n)$ of the CSP $P'(n)$.
4. Simplify $P(n)$ with respect to $P''(n)$ and compute the micro-structure of the resulting CSP (line 15), then compute a k-coloration for the new micro-structure, and repeat the described operations on the simplified form of $P(n)$ (lines 5-19).

Algorithm 1 The dominance-coloration algorithm

```
Require: $P(n)$ {$n$ is the number of variables}
Ensure: {"$P(n)$ inconsistent";"$P(n)$ simplified";"nothing"}
 1: $test \leftarrow 1$
 2: $Old\_P(n) \leftarrow P(n)$
 3: $G \leftarrow$ micro-structure($P(n)$) {Generate the micro-structure}
 4: while ($test \le NBTESTS$) {NBTESTS is the maximal number of steps} do
 5:     $f_k \leftarrow color(G)$
 6:     if ($k < n$) then
 7:         return "$P(n)$ inconsistent"
 8:     else
 9:         $P'(n) \leftarrow build\_Alldiff(P(n), f_k)$
10:         $P''(n) \leftarrow GAC(P'(n))$
11:         if ($is\_inconsistent(P''(n))$) then
12:             return "$P(n)$ inconsistent"
13:         else if ($P''(n) \neq P'(n)$) then
14:             Update($P(n)$, $P''(n)$) {simplification of $P''(n) \Rightarrow$ simplification of $P(n)$}
15:             $G \leftarrow$ micro-structure($P(n)$) {Generation of a new micro-structure}
16:             $test \leftarrow 0$ {simplification of $P(n)$, make NBTESTS new tests}
17:         end if
18:     end if
19:     $test \leftarrow test + 1$
20: end while
21: if ($Old\_P(n) \neq P(n)$) then
22:     return "$P(n)$ simplified"
23: else
24:     return "nothing"
25: end if
```

To make the algorithm terminate, the algorithm repeat the previous steps NBTESTS times. After performing the NBTESTS steps, the algorithm returns one alternative among the following: either the inconsistency of $P(n)$ is shown, or, the inconsistency is not proved, but we get in return a simplified form of $P(n)$; or, neither the inconsistency is shown, nor the CSP $P(n)$ is simplified.

If the method fails to prove the CSP inconsistency, but returns a simplified form of $P(n)$, then one can think that a complete algorithm like Forward Checking or MAC that we expect to combine in future with our method could solve the simplified CSP more efficiently than the original one.

If the method fits in the third case, then usually the coloration used was not pertinent. It is then important to find good heuristics for k-coloration choice, in order to compute k-colorations that avoid this worst case.

Experiments

Now we evaluate our method by experiments. We propose to use our method to check inconsistency of random CSP instances. In this first implementation, we are interested to know the rate of success of this new method in proving CSP inconsistency and to compare it to other methods.

Problem generation

The random instances are generated with respect to the following parameters: n the number of variables, d the size of the domains, $density$ the constraint density , and $tightness$ the tightness. The code of the generator that we used can be found at (*http://www.lirmm.fr/∼bessiere/generator.html*). For each instance of the parameters n, d, $density$ and $tightness$, a sample of 100 problems are generated randomly and the measures are taken in average.

The implemented methods

We implemented and compared six methods: Arc consistency (AC), Path consistency ($Path$), the coloration method given in (Gaur, Jackson, & Havens 1997) ($Color$), a hybrid method that uses both AC and $Color$ where AC is used as a preprocessing for $Color$ (AC_Color), a hybrid method combining the methods $Path$ and $Color$ where $Path$ is used as preprocessing for $Color$ ($Path\_Color$) and the Dominance-Coloration method ($Dominance\_Color$).

In addition to these methods, we used a complete forward checking algorithm (called here FC) as a reference method to evaluate and compare the performances of the different methods. This algorithm detects all the instances that are inconsistent, it is used to compute the success rate of the other incomplete methods in proving inconsistency.

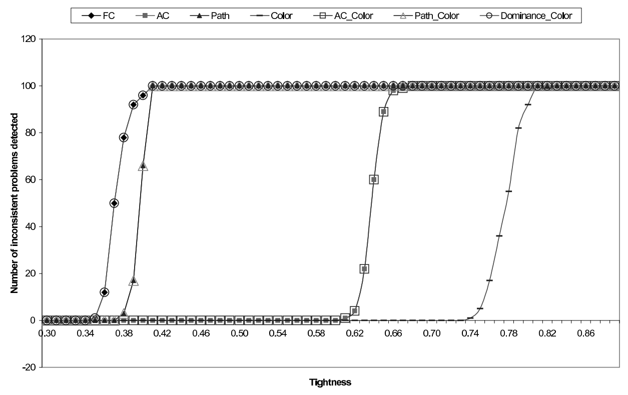


Figure 4: Inconsistency rate w.r.t $tightness$ for ($n = 20$, $d = 10$ et $density = 0.5$)

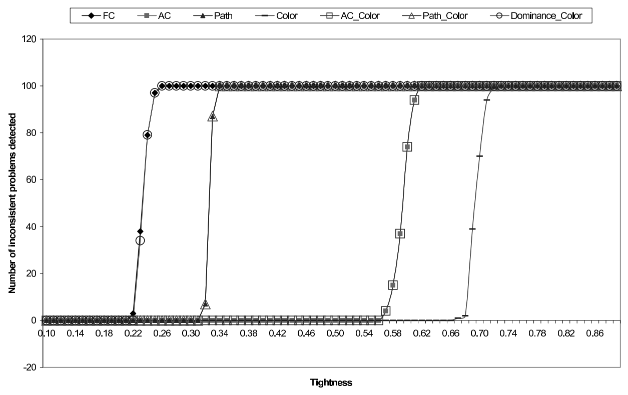


Figure 5: Inconsistency rate w.r.t $tightness$ for $n = 20$, $d = 10$ et $density = 0.9$

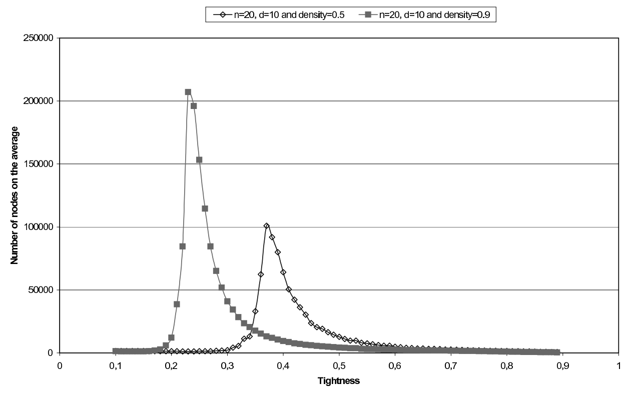


Figure 6: Number of nodes on the average w.r.t $tightness$ for $n = 20$, and $d = 10$

Results

We reported here the curves expressing the inconsistency detection rate with respect to a variation of the tightness for some fixed value of the other parameters n, d and $density$. The limit value of the constant NBTESTS of Algorithm 1 is fixed to 5. A run time limit for solving a sample of 100 instances is fixed to one hour for all the methods.

Figures 4 and 5 show the results of all the methods on the random CSPs where the number of variables is fixed to $n = 20$, the domain size to $d = 10$ and where we tested two densities $density = 0.5$ and $density = 0.9$. Here, we use an incomplete version of the graph coloring algorithm (Brélaz 1979) to compute a k-coloration of the microstructure.

We remark that the method $Dominance\_Color$ has the best inconsistency detection rate, it outperforms all the other methods. Its curve is superposed on the one of the complete method, meaning that it succeeds to solve almost all of the inconsistent problems.

Because path consistency filtering is more robust than arc consistency, the methods $Path$ and $Path\_Color$ have a better rate than both AC et AC_Color. The $Color$ method has the worst inconsistency detection rate, it solved only the over-constrained instances. Notice that the curves of AC

and AC_Color are confused, the same situation happened with the curves of $Path$ and $Path\_Color$. This explains the fact that the coloration operation does not improve AC nor $Path$.

We can see that our method outperforms both the $Path$ and $Path\_Color$ methods in the region where the instances generated are the hardest (near the pick of difficulty). We can see on the curves depicted in Figure 6 representing the average number of search nodes generated by the complete algorithm in function of the tightness that the pick of difficulty is in the region where the $Dominance\_Coloration$ algorithm outperforms the two previous algorithms. The pick is situated approximately in the neighborhood of $tightness = 0.37$ for the instances having a $density = 0.5$ and in the neighborhood of $tightness = 0.23$ for the instances having $density = 0.9$.

Our method looks to succeed where the other methods fail. It detected almost all of the inconsistent instances even in the hard region. It offers the best detection rate for the hard instances. We hope that we can use it in the future to show CSP inconsistency where the best complete algorithm fails.

Conclusion and perspectives

The main purpose behind this work is to offer a contribution to the CP community challenge which consists in finding efficient incomplete method that detect CSP inconsistency. We studied in this work the notion of dominance and showed how to combine it with graph coloring techniques to provide an incomplete method to prove CSP inconsistency. We introduced the method $Dominance\_Coloration$ that includes the $Color$ method given in (Gaur, Jackson, & Havens 1997). Our method is based on a nice property of dominance between the CSP that we are dealing with, and an Alldiff CSP that we derive from the original CSP and a k-coloration of its micro-structure. We showed that the derived Alldiff CSP dominates the original CSP, and some filtering of value in the Alldiff when using the known GAC algorithm induces value filtering in the original CSP without additional effort. Our method can then be seen as a new efficient CSP filtering technique. Its efficiency comes from the fact that GAC is polynomial and performs well on Alldiff CSPs.

The experiments made on random CSP instances, show that our method has a good rate of CSP inconsistency detection even when testing instances in the hard mushy region. The results obtained on the problems checked showed that our method outperforms (in inconsistency detection rate) all of the methods: Arc-Consistency (AC), path-consistency ($Path$), Coloration ($Color$), and the combined methods AC_Color, $Path\_Color$.

The work we give here is just a first contribution towards an answer to the challenge that consists in proving inconsistency by incomplete methods. There are many other directions that we are exploring:

- First, we are looking after coloration heuristics that will produce colorations which are more profitable for the dominance filtering. This will improve dramatically the inconsistency detection rate of the method.
- Since the dominance filtering could remove some extra inconsistent values from the variable domains, one can use it as a filtering procedure that can be applied prior to or during search. This is a particular interesting point that we are investigating theoretically and practically.
- Finally, it will be important to find other pertinent CSP derivations from the original CSP $P(n)$ that conserve the dominance property.

References

Aloul, F.; Lynce, I.; and Prestwich, S. 2007. Symmetry breaking in local search for unsatisfiability. In *SymCon'07*, 9–13.

Audemard, G., and Simon, L. 2007. Gunsat: A greedy local search algorithm for unsatisfiability. In *IJCAI'07*, 2256–2261.

Bès, J., and Jégou, P. 2005. Proving graph un-colorability with a consistency check of csp. In *ICTAI'05*, 693–694. Hong Kong, China: IEEE. POSTER.

Brélaz, D. 1979. New methods to color vertices of a graph. *Commun. ACM* 22(4):251–256.

Davis, M.; Logemann, G.; and Loveland, D. 1962. A machine program for theorem-proving. *Commun. ACM* 5(7):394–397.

Gaur, D. R.; Jackson, W. K.; and Havens, W. S. 1997. Detecting unsatisfiable csps by coloring the micro-structure. In *AAAI/IAAI*, 215–220.

Goldberg, E. 2002. Proving unsatisfiability of cnfs locally. *J. Autom. Reasoning* 28(5):417–434.

Hoos, H. H., and Stützle, T. 2004. *Stochastic Local Search: Foundations & Applications*. Elsevier / Morgan Kaufmann.

Jégou, P. 1993. Decomposition of domains based on the micro-structure of finite constraint satisfaction problems. In *Proceedings AAAI'93*.

Klerk, E. D.; Maaren, H. V.; and Warners, J. P. 2000. Relaxations of the satisfiability problem using semidefinite programming. *J. Autom. Reason.* 24(1-2):37–65.

Prestwich, S., and Lynce, I. 2006. Local search for unsatisfiability. In *SAT'06*, LNCS, 283–296. Springer.

Régin, J.-C. 1994. A filtering algorithm for constraints of difference in csps. In *AAAI '94: (vol. 1)*, 362–367. Menlo Park, CA, USA.

Selman, B.; Kautz, H. A.; and McAllester, D. A. 1997. Ten challenges in propositional reasoning and search. In *IJCAI'97*, 50–54.

Selman, B.; Levesque, H. J.; and Mitchell, D. G. 1992. A new method for solving hard satisfiability problems. In *AAAI'92*, 440–446.