

Pruning by the Use of Equally Constrained Variables

Igor Razgon ^{*} and Amnon Meisels

Department of Computer Science,
Ben-Gurion University of the Negev,
Beer-Sheva, 84-105, Israel
{irazgon,am}@cs.bgu.ac.il

Abstract. We introduce a notion of equally constrained variables of a constraint network. We propose a method of pruning that uses the notion. We combine the proposed method of pruning with FC-CBJ and call the obtained algorithm FC-CBJ-EQ. Our experimental results show that FC-CBJ-EQ outperforms FC-CBJ on constraint networks that encode randomly generated instances of the graph k -coloring and the subgraph isomorphism problems.

1 Introduction

When a problem is modeled as a CSP, the obtained model not always expresses specific features of the problem. Recognizing these "implicit" features could significantly reduce the search space. Examples to the fact that recognizing problem-specific features increases efficiency of constraint processing include: symmetry breaking methods (for example, [5, 6, 11]), recognizing tractable classes of CSPs (for example, [1, 3, 15]), methods that detect interchangeability (for example, [7, 16, 2]), etc.

In this paper we introduce a method of pruning for complete constraint solvers. The method is based on the discovery of *equally constrained variables*. Two variables v_1 and v_2 of a CSP are equally constrained with a variable v_3 of the CSP if for any value a that belongs to the domains of both v_1 and v_2 and for any value b that belongs to the domain of v_3 , the assignments $\langle v_1, a \rangle$ and $\langle v_3, b \rangle$ are compatible if and only if $\langle v_2, a \rangle$ and $\langle v_3, b \rangle$ are compatible.

Equally constrained variables occur in CSP-encodings of graph-theoretic problems. Consider, for example, the graph k -coloring problem. Let G be a graph which we would like to color in at most k colors. The problem can be modeled as a CSP in which variables correspond to the vertices of G , all domains are $\{1, \dots, k\}$, variables that correspond to adjacent vertices are "connected" by the inequality constraint. Variables v_1 and v_2 are equally constrained with a variable v_3 if the vertices that correspond to v_1 and v_2 are both adjacent to the vertex corresponding to v_3 or both non-adjacent to it.

^{*} The author would like to acknowledge the Lynn and William Frankel Center for Computer Sciences for financial support

We propose a modification of FC-CBJ [10] called FC-CBJ-EQ that utilizes information about equally constrained variables. It differs from FC-CBJ in the following two aspects.

1. At the preprocessing stage FC-CBJ-EQ finds all triples of variables v_1, v_2, v_3 of the processed constraint network such that v_1 and v_2 are equally constrained with v_3 .
2. When FC-CBJ-EQ deletes a value val from the current domain of a variable v , it associates with the value a so-called r -set, which is a set of unassigned variables responsible for discarding val . To compute r -sets, we adopt the technique described in [12, 13]. After computation of the r -set of val , FC-CBJ-EQ scans all unassigned variables. If it detects that some variable u and v are equally constrained on all variables of the r -set associated with val , the value val is removed from the current domain of u .

Thus the pruning effect of FC-CBJ-EQ is *filtering domains of unassigned variables during backtrack execution*. Note that such a pruning strategy is quite unusual for complete constraint solvers. Execution of constraint solvers can be represented as a sequence of lookahead and look-back phases [4]. During the lookahead phase a new assignment is performed and domains of unassigned variables are filtered. During the look-back phase, a solver discards a subset of assignments of the current partial solution. FC-CBJ-EQ filters domains of unassigned variables during the look-back phase. This is because the implicit constraints used for the filtering are discovered only after backtrack and thus cannot be used during the preceding lookahead phase. Such a method of pruning can be considered as a "lazy" or "postponed" lookahead.

We compare FC-CBJ-EQ and FC-CBJ on CSPs that encode randomly generated instances of the graph k -coloring and the subgraph isomorphism problems. We selected these problems for evaluation of FC-CBJ-EQ because they have potentially many occurrences of equally constrained variables. Our experiments demonstrate that FC-CBJ-EQ outperforms FC-CBJ on these problems.

The proposed method of pruning has a potential for further development, generalization and combination with other pruning methods.

The rest of the paper is organized as follows. Section 2 provides the necessary background. Section 3 introduces the notion of equally constrained variables and proves the properties that are used to develop the proposed pruning method. Section 4 describes the algorithm FC-CBJ-EQ. Section 5 provides experimental results. Section 6 discusses further development of the proposed approach.

2 Preliminaries

A binary *constraint network* (CN) $Z = \langle V, D, C \rangle$ is a triple consisting of a set of *variables* V , a set of *domains* D and a set of *constraints* C . Let $V = \{v_1, \dots, v_n\}$. Then $D = \{D_{v_1}, \dots, D_{v_n}\}$, where D_{v_i} is the domain of values of v_i , $C = \{C_{v_i, v_j} \mid i \neq j, 1 \leq i, j \leq n\}$, where $c_{v_i, v_j} \subseteq D_{v_i} \times D_{v_j}$ is the set of all compatible pairs of values of v_i and v_j . We assume that $C_{v_i, v_j} = C_{v_j, v_i}$. We

refer to the parts of Z as $V(Z)$, $D(Z)$, and $C(Z)$. To emphasize that a value val belongs to the domain of a variable v , we refer to this value as $val(v)$.

An *assignment* of a CN Z is a pair $\langle v_i, val \rangle$ such that $v_i \in V(Z)$, $val \in D_{v_i}$. A consistent set of assignments is a *partial solution* of Z . A partial solution that assigns all the variables of Z is a *solution* of Z . A CN that has no solution is *insoluble*. A *subnetwork* of a CN Z is a CN obtained by removing variables or values from Z . We define two types of subnetworks.

Definition 1. A *projection* of a CN Z to a set of variables $V' \subseteq V(Z)$ denoted by $Z(V')$ is a subnetwork of Z obtained by removing from Z all variables of $V(Z) \setminus V'$.

Definition 2. A *subnetwork of a CN Z induced by a partial solution P* denoted by $Z|_P$ is obtained by removing from Z all the variables assigned in P and removing from the domains of the remaining variables all values inconsistent with P .

The notions of projection and induced CSP are illustrated on Figure 1 where the ellipses represent the domains of values of variables and incompatible values of pairs of variables are connected by arcs.

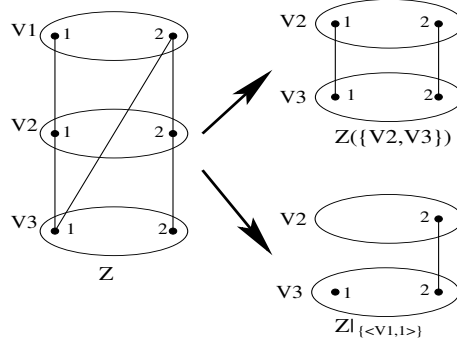


Fig. 1. Illustration of the projection and induced CSP notions.

Definition 3. A *nogood* of a CN Z is a partial solution of Z that cannot be extended to a full solution. In other words, P is a *nogood* of Z if and only if $Z|_P$ is *insoluble*.

3 Equally Constrained Variables and Responsibility Sets

Definition 4. Let Z be a CN and $v_1, v_2, v_3 \in V(Z)$. v_1 and v_2 are *equally constrained* with v_3 if for every $val' \in D_{v_1} \cap D_{v_2}$ and for every $val'' \in D_{v_3}$, $\langle v_1, val' \rangle$ and $\langle v_3, val'' \rangle$ are compatible if and only if $\langle v_2, val' \rangle$ and $\langle v_3, val'' \rangle$ are compatible.

Consider the CN on Figure 2. Variables V_1 and V_3 are equally constrained with V_2 . This is easy to verify by following the arcs from values 2, 3 of V_3 and V_1 to the values of V_2 .

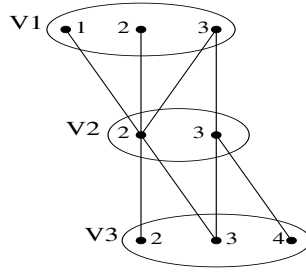


Fig. 2. Illustration of the notion of equally constrained variables.

Definition 5. Let Z be an insoluble CN. A set V' is called a responsibility set of Z if $Z(V' \cap V(Z))$ is insoluble.

Let Z be a CN and $v \in V(Z)$. Assume that $\{v, val\}$ is a nogood and let $V' \in V(Z)$ be a responsibility set of $Z|_{\{v, val\}}$. We formulate for this situation two properties that we use in the next section to construct our pruning procedure.

Proposition 1. Let $u \in V(Z) \setminus V'$ such that v and u are equally constrained with every variable of V' . If $val \in D_u$ then $\{u, val\}$ is a nogood and V' is a responsibility set of $Z|_{\{u, val\}}$.

To illustrate Proposition 1, consider the CN in Figure 3. We can see that $\{V_1, 1\}$ is a nogood and $\{V_2, V_3\}$ is a responsibility set of $Z|_{\{V_1, 1\}}$. Observe also that V_1 and V_4 are equally constrained with V_2 and V_3 . Therefore, $\{V_4, 1\}$ is also a nogood with a responsibility set $\{V_2, V_3\}$.

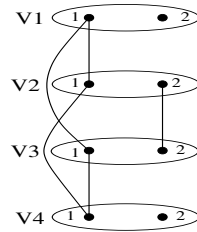


Fig. 3. Example of pruning according to Proposition 1.

Proof of Proposition 1. Consider two CNs: $Z(V' \cup \{v\})|_{\{v, val\}}$ and $Z(V' \cup \{u\})|_{\{u, val\}}$. Since u and v are equally constrained with all variables of V' , these

two CNs are equal. The former is insoluble, therefore the latter is also insoluble. The insolubility of $Z|_{\{\langle u, val \rangle\}}$ follows from the insolubility of $Z(V' \cup \{u\})|_{\{\langle u, val \rangle\}}$.

Note that V' is a responsibility set of $Z(V' \cup \{u\})|_{\{\langle u, val \rangle\}}$ because $V' = Z(V' \cup \{u\})|_{\{\langle u, val \rangle\}}$, therefore V' is a responsibility set of $Z|_{\{\langle u, val \rangle\}}$. \square

Proposition 2. *Let $u \in V'$ such that u and v are equally constrained with every variable of $V' \setminus \{u\}$. Assume that $val \in D_u$ and that for every value $val'(v)$, if $\langle u, val \rangle$ is compatible with $\langle v, val' \rangle$, then $val' \in D_u$ and $\langle v, val \rangle$ is compatible with $\langle u, val' \rangle$. Then $\{\langle u, val \rangle\}$ is a nogood and $V' \setminus \{u\} \cup \{v\}$ is a responsibility set of $Z|_{\{\langle u, val \rangle\}}$.*

Figure 4 illustrates Proposition 2. $\{\langle V_1, 1 \rangle\}$ is a nogood in the CN shown in the figure. The responsibility set of $Z|_{\{\langle V_1, 1 \rangle\}}$ is $\{V_2, V_3, V_4\}$. The variables V_1 and V_2 are equally constrained with V_3 and V_4 . Also $\langle V_2, 1 \rangle$ is compatible with $\langle V_1, 2 \rangle$ as well as $\langle V_1, 1 \rangle$ is compatible with $\langle V_2, 2 \rangle$. Therefore $\{\langle V_2, 1 \rangle\}$ is a nogood with the responsibility set $\{V_2, V_3, V_4\}$.

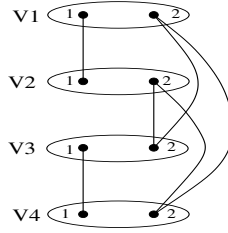


Fig. 4. Example of pruning according to Proposition 2.

Proof of Proposition 2. Assume that the statement of the lemma is not true. Then there is a partial solution P if Z assigning the variables of $V' \cup \{u, v\}$ and such that $\langle u, val \rangle \in P$. Let val' be the assignment of v in P . Let $P' = P \setminus \{\langle u, val \rangle \langle v, val' \rangle\}$.

According to the statement of the lemma, $val' \in D_u$, $\langle v, val \rangle$ is compatible with $\langle u, val' \rangle$ and $\{\langle v, val \rangle \langle u, val' \rangle\}$ is consistent with P' .

Therefore, $P' \cup \{\langle v, val \rangle \langle u, val' \rangle\}$ is a partial solution contradicting the statement that V' is a responsibility set of $Z|_{\{\langle v, val \rangle\}}$. \square

4 A Pruning Procedure for FC-CBJ

In this section we propose a pruning procedure based on the discovery of equally constrained variables. The procedure can be combined with a complete constraint solver. We show its combination with FC-CBJ [10]. We call the obtained solver FC-CBJ-EQ.

The further description of the FC-CBJ-EQ solver is divided into three parts. In the first part the maintenance of responsibility sets for insoluble CNs visited

by a constraint solver is described. In the second part the algorithm FC-CBJ-EQ is presented. In the last part the correctness of the backtrack procedure of FC-CBJ-EQ is proved.

4.1 Maintaining Responsibility Sets

In this section we describe a method that associates with every value removed from the current domain of a variable a set which we call the r -set of this value. The method is described for FC-CBJ.

FC-CBJ has two reasons to remove values. The first one is that a value of an unassigned variable is incompatible with the last assignment of the current partial solution. Such a value is eliminated by the lookahead procedure of FC-CBJ. Such a value is associated with the empty r -set.

The second reason for removing a value in FC-CBJ is that the domain of some unassigned variable becomes empty. In this case an assignment of the current partial solution is discarded and the value of the assignment is removed by the backtrack procedure from the current domain of its variable. The r -set associated with such a value is computed as follows. Let $\langle v, val \rangle$ be the assignment being discarded. Let w be the unassigned variable whose current domain becomes empty. We associate with val , which is removed from the current domain of v , the set $(S \cup \{w\}) \setminus \{v\}$, where S is the union of the r -sets of all the values of w .

To demonstrate the method consider the following example.

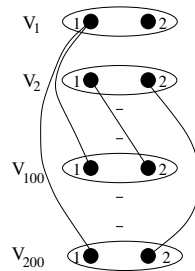


Fig. 5. Illustration of responsibility sets.

Example 1. Let Z be the CN illustrated on Figure 5. Assume that FC-CBJ is processing the CN. Let $\{\langle V_1, 1 \rangle, \langle V_2, 1 \rangle\}$ be the current partial solution. After application of the lookahead procedure, the domain of variable V_{100} becomes empty. All the removed values are associated with empty sets. Next, FC-CBJ backtracks. It removes value 1 from the current domain of V_2 and, according to the method described above, associates this eliminated value with the r -set $\{V_{100}\}$. The next partial solution tried by FC-CBJ is $\{\langle V_1, 1 \rangle, \langle V_2, 2 \rangle\}$. This partial solution empties the domain of variable V_{200} . Every value of V_{200} is associated with the empty r -set, therefore the r -set associated with $\langle V_2, 2 \rangle$ is

$\{V_{200}\}$. When FC-CBJ proceeds, it finds that the current domain of V_2 is empty. It discards the assignment $\langle V_1, 1 \rangle$, removes 1 from the current domain of V_1 , and associates with this value the set $\{V_{100}, V_{200}, V_2\}$.

It is possible to see from the example that the r -set of a value is a responsibility set of the CN induced by a nogood that was discarded by removing the value. For example, $\{V_{100}\}$ is a responsibility set of $Z|_{\{\langle V_1, 1 \rangle, \langle V_2, 1 \rangle\}}$, $\{V_2, V_{100}, V_{200}\}$ is a responsibility set of $Z|_{\{\langle V_1, 1 \rangle\}}$. We prove a more general claim in Section 4.3.

4.2 Pruning Procedure of Algorithm FC-CBJ-EQ

In this section we describe the algorithm FC-CBJ-EQ. It differs from FC-CBJ in the following three aspects.

1. FC-CBJ-EQ has a preprocessing procedure which computes whether v_1 and v_2 are equally constrained with v_3 for every triple of different variables (v_1, v_2, v_3) of the processed CN. The computation directly follows Definition 4. That is, for every value $a \in D_{v_1} \cap D_{v_2}$ and for every value $b \in D_{v_3}$, FC-CBJ-EQ checks whether $a(v_1)$ and $a(v_2)$ have the same compatibility with $a(v_3)$.
2. The lookahead procedure of FC-CBJ-EQ associates with the empty r -set every value incompatible with the current assignment.
3. FC-CBJ-EQ has a modified backtrack procedure.

The rest of the part describes the backtrack procedure of FC-CBJ-EQ. Similarly to FC-CBJ, FC-CBJ-EQ applies the backtrack procedure when the current domain of some unassigned variable becomes empty.

An important notion used in the algorithm is a *conflict set* [10]. The conflict set of a variable x denoted by $Conf(x)$ is a set of variables assigned by the current partial solution such that if P' is the subset of the current partial solution assigning these variables then for any $val'(x)$ removed from the current domain of x , either P' is incompatible with $\langle x, val' \rangle$ or $P' \cup \{\langle x, val' \rangle\}$ is a nogood.

Let w be the variable with empty current domain that caused FC-CBJ-EQ to apply the backtrack procedure. Let v be the last (chronologically) assigned variable of $Conf(w)$. Let val be the assignment of v in the current partial solution. Given the data, the backtrack procedure of FC-CBJ-EQ is described in Algorithm 1.

The pseudocode can be divided into three parts. In the first part (lines 1-7), the algorithm keeps the conflict set of w (line 1), deletes val from the current domain of v (line 2), and prepares for further pruning (lines 3-7). In the second part (lines 8-22), the algorithm checks every unassigned variable u and if the current domain of u contains val , the algorithm tries to remove val from the domain. The attempt of removing is described in lines 9-21.

The pseudocode in line 9-21 can again be divided into two parts (lines 9-13 and 15-19, respectively). The first part tries to prune $\langle u, val \rangle$ checking the condition stated in Proposition 1. The second part does the same regarding Proposition 2. If $\langle u, val \rangle$ is pruned in one of these parts, it is associated with

Algorithm 1 THE BACKTRACK PROCEDURE OF FC-CBJ-EQ

```
1:  $Conf \leftarrow Conf(w)$ 
2: Run the backtrack procedure of FC-CBJ (see [10], Section 3.9, the procedure fc-
   cbj-unlabel)
3: Let the  $r$ -set of  $val(v)$  be equal to  $(S \cup \{w\}) \setminus \{v\}$ , where  $S$  is the union of the
    $r$ -sets of all the values of  $w$ 
4: Discard the  $r$ -sets of all values that were restored in their current domain executing
   line 2
5: Let  $V^*$  be the  $r$ -set of  $val(v)$ 
6: Let  $V'$  be the set of all unassigned variables contained in  $V^*$ 
7: Let  $A$  be the subset of assigned variables of  $V^* \setminus V'$ 
8: for every unassigned variable  $u$  do
9:   if  $u$  and  $v$  are equally constrained with all variables of  $V'$  and  $val$  belongs to the
     current domain of  $u$  then
10:    if  $u \notin V'$  then
11:      Remove  $val$  from the current domain of  $u$ 
12:       $Conf(u) \leftarrow Conf(u) \cup Conf \cup A \setminus \{v\}$ 
13:      Let the  $r$ -set of  $val(u)$  be equal to  $V'$ 
14:    else
15:      if for every  $val'$  of the current domain of  $v$  compatibility of  $\langle u, val \rangle$  with
         $\langle v, val' \rangle$  implies that  $val'$  belongs to the current domain of  $u$  and  $\langle v, val \rangle$  is
        compatible with  $\langle u, val' \rangle$  then
16:        Remove  $val$  from the current domain of  $u$ 
17:         $Conf(u) \leftarrow Conf(u) \cup Conf(v) \cup A \setminus \{v\}$ 
18:        Set the  $r$ -set of  $u(val)$  equal  $V' \setminus \{u\} \cup \{v\}$ 
19:      end if
20:    end if
21:  end if
22: end for
```

a r -set and the conflict set of u is updated. Consistency of updating of these structures is proved in Section 4.3.

Let us see an example that illustrates a possible scenario of execution of FC-CBJ-EQ when it exhibits more powerful pruning ability than that of FC-CBJ.

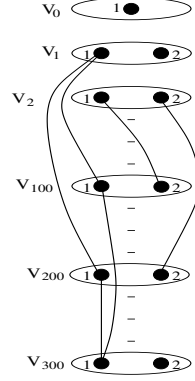


Fig. 6. Illustration of pruning that uses equally constrained variables.

Example 2. Let Z be the CN illustrated on Figure 6. Assume that FC-CBJ-EQ is processing Z , its current partial solution is $\{\langle V_0, 1 \rangle, \langle V_1, 1 \rangle, \langle V_2, 1 \rangle\}$, and the current domains of variables V_0, V_1, V_2 equal their initial domains. After a number of backtracks, FC-CBJ-EQ discards all values of V_2 and also $1(V_1)$. As in the previous example, the r -set associated with $1(V_1)$ is $\{V_{100}, V_{200}\}$. Then the backtrack procedure of FC-CBJ-EQ detects that V_1 and V_{300} are equally constrained on both V_{100} and V_{200} and that 1 belongs to the current domain of V_{300} . Therefore V_{300} satisfies the conditions stated in line 9 of Algorithm 1. Further, it satisfies the condition stated in line 10. Therefore, $1(V_{300})$ is deleted from the current domain of V_{300} and associated with the responsibility set $\{V_2, V_{100}, V_{200}\}$.

When FC-CBJ-EQ proceeds to execute it tries to assign V_1 with 2 . The lookahead procedure applied after the assignment removes 2 from the current domain of V_{300} . Thus, the current domain of V_{300} becomes empty and FC-CBJ-EQ backtracks. Note that FC-CBJ would not backtrack in this case, because it would not remove $1(V_{300})$ together with removing $1(V_1)$. Next, FC-CBJ tries to reassign V_0 , but 1 is the only value in the current domain of V_0 , therefore FC-CBJ-EQ reports that the CN is insoluble.

4.3 Correctness

To prove correctness of the backtrack procedure of FC-CBJ-EQ (Algorithm 1), we define a notion of a *consistent state* of FC-CBJ-EQ. Then we show that application of Algorithm 1 to a consistent state of FC-CBJ-EQ produces another consistent state.

Consider FC-CBJ-EQ that is processing a CN Z . A *state* of FC-CBJ-EQ contains the following data:

- the current partial solution;
- the current domains of variables;
- the conflict sets of variables;
- values removed from the current domains of variables and the r -sets associated with them.

Let $val(v)$ be a value removed from the current domain of a variable v . Let P' be a subset of the current partial solution assigning the variables of $Conf(v)$, the conflict set of v . We say that $Conf(v)$ is *consistent with* $v(val)$ if either $\langle v, val \rangle$ is incompatible with P' or $P' \cup \{\langle v, val \rangle\}$ is a nogood.

We say that $Conf(v)$ is *consistent* if it is consistent with every value removed from the current domain of v .

The r -set of $val(v)$ is consistent if either it is empty or it is a responsibility set of $Z|_{P' \cup \{\langle v, val \rangle\}}$.

A state of FC-CBJ-EQ is *consistent* if all conflict sets and all r -sets are consistent.

Now we are ready to formulate the theorem that claims the correctness of Algorithm 1.

Theorem 1. *Application of Algorithm 1 to a consistent state I of FC-CBJ-EQ produces another consistent state R of FC-CBJ-EQ.*

Proof. We assume that FC-CBJ is correct. Thus, we prove correctness only for those parts of the resulting state R that have been obtained by additional operations performed by FC-CBJ-EQ. In particular, we prove that:

1. the r -set associated with $val(v)$ (line 3 of Algorithm 1) is consistent;
2. the conflict sets of all variables whose domains have been pruned in lines 8-22 of Algorithm 1 are consistent;
3. the r -sets of all values removed in lines 8-22 of Algorithm 1 are consistent.

Assume by contradiction that the first claim does not hold. Let V^* be an r -set of $val(v)$ computed by Algorithm 1. Inconsistency of V^* means the following.

- V^* is not empty.
- Let P be the subset of the current partial solution that assigns $Conf(v)$ and let $P' = P \cup \{\langle v, val \rangle\}$. Then V^* is not a responsibility set of $Z|_{P'}$.

Note that P' is a subset of the current partial solution at the input state I . Furthermore, it assigns all the variables of $Conf$, the conflict set of the variable w in state I whose empty current domain causes FC-CBJ-EQ to backtrack. Let P'' be the subset of P' assigning the variables of $Conf(w)$ only.

Considering that the current domain of w is empty in state I , we conclude that P'' is a nogood in Z . Taking into account that $P'' \subseteq P'$ and that V^* is not a responsibility set of $Z|_{P'}$, we get that V^* is not a responsibility set of $Z|_{P''}$. Therefore $Z(Conf \cup V^*)$ has a solution P^* such that $P'' \subseteq P^*$.

Now, let us remember that V^* consists of w and the union of all r -sets associated with the values of w (Section 3.1) and excludes v . Let $\langle w, val'' \rangle \in P^*$. Let V'' be the r -set of $val''(w)$ in the state I . Let P^{restr} be the subset of P^* that assigns $Conf \cup \{w\} \cup V'' \setminus \{v\}$. Note that $P'' \subset P^{restr}$.

According to our assumption, the state I is consistent, therefore the r -set V'' of $val''(w)$ is consistent. This means that either it is empty or it is a responsibility set of $Z|_{P'' \cup \{\langle w, val'' \rangle\}}$. In the first case, $\langle w, val'' \rangle$ is incompatible with some assignment of P'' . In the second case, $P'' \cup \{\langle w, val'' \rangle\}$ is a nogood in $Z(Conf \cup \{w\} \cup V'')$. In both cases we get a contradiction with P^{restr} being a partial solution of $Z(Conf \cup \{w\} \cup V'')$.

Let us prove consistency of conflict and r -sets produced in lines 9-13. Proving the first part, we got that V^* is not only a responsibility set of $Z|_{P'}$ but also a responsibility set of $Z|_{P''}$. Let A be a subset of V^* that contains variables assigned in the current partial solution but unassigned in P'' (line 7 of Algorithm 1). Let P^A be the subset of the current partial solution of the state R that assigns the variables of A . The set V' obtained in line 6 of Algorithm 1 is a responsibility set of $Z|_{P'' \cup P^A}$.

Considering that $\langle v, val \rangle \in P'' \cup P^A$, we can rewrite $Z|_{P'' \cup P^A}$ as follows: $(Z|_{(P'' \cup P^A) \setminus \{\langle v, val \rangle\}})|_{\{\langle v, val \rangle\}}$. To make the notation shorter, denote the CN in the brackets by Z' .

Observe that $\{\langle v, val \rangle\}$ is a nogood in Z' and that V' is a responsibility set of $Z'|_{\{\langle v, val \rangle\}}$. Therefore if u and v are equally constrained on all variables of V' (line 9) and $u \notin V'$ (line 10) then $\{\langle v, val \rangle\}$ is a nogood in Z' and V' is a responsibility set of $Z'|_{\{\langle v, val \rangle\}}$.

Recall that Z' is a subnetwork of Z induced by $(P'' \cup P^A) \setminus \{\langle v, val \rangle\}$. Therefore $(P'' \cup P^A) \setminus \{\langle v, val \rangle\} \cup \{\langle u, val \rangle\}$ is a nogood in Z . Taking into account that $(P'' \cup P^A) \setminus \{\langle v, val \rangle\}$ is a subset of the current partial solution, we get that $\{\langle u, val \rangle\}$ is inconsistent with the current partial solution. Moreover, when we remove $val(u)$ from the current domain of u , we can preserve consistency of $Conf(u)$ by adding the set of variables assigned in $(P'' \cup P^A) \setminus \{\langle v, val \rangle\}$. But the set of variables equals $Conf \cup A \setminus \{v\}$, exactly what is added in line 12. Therefore, the conflict set of u obtained as result of pruning in lines 9-13 is consistent.

The consistency of conflict and r -sets created in lines 15-19 can be proved in a similar way. \square

5 Experiments

The goal of our experiments is to analyze behavior of FC-CBJ-EQ on the CNs with potentially large number of occurrences of equally constrained variables. We compare FC-CBJ-EQ with FC-CBJ on CNs that encode the graph k -coloring and subgraph isomorphism problems.

We measure the computational effort in the number of consistency checks, the number of nodes (partial solutions) visited, and the CPU time. We implemented both FC-CBJ and FC-CBJ-EQ in Microsoft Visual C++ 6.0. The experiments

were performed on a computer with Windows 98 Operating System, 300 MHz AMD processor, and 128MB RAM.

Every result shown in the tables below is obtained as average of 20 runs. To avoid exponential explosion of running time, the algorithms stop when the number of performed consistency checks is greater than $5 * 10^8$. Therefore, we use an additional computational effort measure: the percent of runs finished in less than $5 * 10^8$ consistency checks. Computing the average measures, we do not take into account the instances on which both the algorithms perform more than $5 * 10^8$ consistency checks.

In all the experiments both FC-CBJ and FC-CBJ-EQ use the fail first variable ordering heuristic [9] which takes a variable with the least domain size and the min-conflict value ordering heuristic [8].

5.1 Graph k -coloring problems

Given a graph G and a natural number k . The task is to color G in at most k colors. A CN Z suitable to this problem is constructed as follows:

- $V(Z)$ corresponds to $V(G)$, the set of vertices of G ;
- the domain of every variable is $\{1, \dots, k\}$;
- for any two nonadjacent vertices of G , the corresponding variables are not constrained (any two values are compatible);
- for any two adjacent vertices, the corresponding variables are "connected" by the inequality constraint.

We tested FC-CBJ and FC-CBJ-EQ on randomly generated instances of k -coloring for graphs with 30 vertices. The other parameters used for generation of graphs are the number of colors and the density of graphs. By the density of a graph we mean the probability that a pair of vertices of the graph are adjacent.

The results of experiments are given in Table 1. The first two columns determine parameters of the instances. The other columns are divided in two parts. The first part provides information about computational effort spent by FC-CBJ. The other gives the same information regarding FC-CBJ-EQ.

The columns "finished" contain the percent of runs in which the algorithm finished performing less than $5 * 10^8$ consistency checks. The columns "cons" present the number of consistency checks performed by the algorithms divided by 100000. The columns "nodes" present the number of nodes visited by algorithms divided by 1000. Finally, the columns "CPU" show the running time of the algorithms in seconds.

We can get the following information from Table 1.

- FC-CBJ-EQ outperforms FC-CBJ in the number of nodes visited.
- FC-CBJ-EQ outperforms FC-CBJ in the number of consistency checks and the running time if the effort spent to preprocessing is small relatively to the effort required to solve the problem. For easy problems, such as in the first line of the table, FC-CBJ-EQ performs worse than FC-CBJ because the preprocessing procedure of FC-CBJ-EQ takes the number of consistency

parameters		FC-CBJ				FC-CBJ-EQ			
colors	density	finished	cons	nodes	CPU	finished	cons	nodes	CPU
6	50	100	14	46	15	100	21	37	17
7	60	100	163	561	217	100	77	221	114
8	60	100	1602	6619	2834	100	458	1750	817
8	70	100	582	1985	885	100	252	443	410
8	80	100	80	219	124	100	23	1	1
9	70	80	1280	4434	2146	80	39	45	34
9	80	100	813	2191	1213	100	30	2	2
10	70	60	1250	4912	2330	80	737	2432	1541
10	80	0	5000	17674	9914	100	1065	4212	2925

Table 1. Experimental Results on the Graph k -Coloring Problem

checks comparable with the number of consistency checks required to solve the problem.

5.2 Subgraph Isomorphism Problem

Given two graphs G_1 and G_2 . The task is to determine whether G_2 is a subgraph of G_1 . To encode the problem as a CN, we assume that G_1 and G_2 have the same number of vertices denoted by n and that the vertices of the graphs are labeled by $\{1, \dots, n\}$.

We correspond to the problem a CN Z . The variables of Z are denoted by V_1, \dots, V_n . The values of every domain of Z are denoted by $\{1, \dots, n\}$. The constraints are defined by the following two rules:

- for every pair of nonadjacent vertices i and j of G_2 , the variables V_i and V_j are connected by the inequality constraint;
- for every pair i and j of adjacent vertices of G_2 , $\langle V_i, k \rangle$ and $\langle V_j, l \rangle$ are compatible if and only if the vertices k and l are adjacent in G_1 .

It is possible to verify that the CN Z has a solution if and only if G_2 is a subgraph of G_1 . Any solution of Z is a mapping from vertices of G_2 to the vertices of G_1 . A vertex i of G_2 is mapped to a vertex k of G_1 if V_i is assigned with k in the solution.

We tested FC-CBJ and FC-CBJ-EQ on the instances of the subgraph isomorphism problems where G_1 and G_2 have 18 vertices every one. We randomly generated instances of the problem using parameters d_1 and d_2 that are densities of G_1 and G_2 , respectively. The results are collected in Table 2. The meaning of columns is the same except the parameters of the problem.

As we can see, FC-CBJ-EQ exhibits the same behavior as for the instances of the graph k -coloring. FC-CBJ-EQ outperforms FC-CBJ except the instances where the time of solving is comparable with the time of preprocessing.

parameters		FC-CBJ				FC-CBJ-EQ			
d_1	d_2	finished	cons	nodes	CPU	finished	cons	nodes	CPU
30	20	100	5	12	8	100	29	11	9
30	30	100	1	2	1	100	22	1	2
40	30	100	5	10	7	100	25	9	7
40	40	100	3	5	4	100	21	4	5
50	30	100	25	86	48	100	45	83	53
50	40	100	32	10	60	100	48	9	60
50	50	100	10	27	17	100	27	23	17
60	40	100	336	1517	815	100	323	1352	860
60	50	100	63	219	126	100	74	195	131
60	60	100	29	96	60	100	40	66	52
70	50	100	152	619	360	100	154	549	341
70	60	100	247	991	582	100	210	750	521
70	70	100	120	483	287	100	101	299	221
80	60	100	1622	8353	4627	100	1211	5852	3457
80	70	100	2118	11119	6110	100	1336	6601	4202
80	80	100	1747	8806	4912	100	666	2971	2035
90	70	100	129	942	507	100	132	790	475
90	80	60	39	299	151	60	47	173	95

Table 2. Experimental Results on the Subgraph Isomorphism Problem

6 Further Development

We presented a modification of FC-CBJ called FC-CBJ-EQ that collects additional information during search and uses the information for pruning. Possible directions of further development include: improvement, generalization, and combination with other pruning methods. Let us discuss possibilities of improvement in a more detailed form.

We see two possibilities of improvement: better utilization of information collected by FC-CBJ and collecting more information that could be useful for pruning.

To understand a possible way of better utilization of the additional information maintained by FC-CBJ-EQ, recall, that FC-CBJ-EQ uses the following pruning principle. When the algorithm discovers a nogood with last assignment $\langle v, val \rangle$, it tries to remove val from the current domains of other unassigned variables. Therefore the pruning is "related" to the last assignment of the discovered nogood. However, a similar procedure could be applied to other assignments of the discovered nogood. Such a strategy could result in a better pruning behavior.

Collecting more information during search is another possibility to increase pruning effect. We believe that responsibility sets are the most crucial information for success of the proposed method of pruning. Therefore we will try to construct methods that associate with every removed value many r -sets, maintain them in a succinct form and quickly check possibility of pruning for every

one of the maintained sets. This could be done in a way similar to that described in [14].

References

1. A. Bulatov and P. Jeavons. An algebraic approach to multi-sorted constraints. In *Principles and Practice of Constraint Programming-CP2003*, pages 183–187, Kinsale,Ireland, oct 2003. Springer.
2. B. Choueiry and G. Noubir. On the computation of local interchangeability in discrete constraint satisfaction problems. In *AAAI/IAAI*, pages 326–333, 1998.
3. M. Cooper, D. Cohen, and P. Jeavons. Characterising tractable constraints. *Artificial Intelligence*, 65:347–361, 1994.
4. R. Dechter and D. Frost. Backjump-based backtracking for constraint satisfaction problems. *Artificial Intelligence*, 136:147–188, 2002.
5. T. Fahle, S. Schamberger, and M. Sellmann. Symmetry breaking. In *CP2001*, pages 93–108. Springer, November 2001.
6. F. Focacci and M. Milano. Global cut framework for removing symmetries. In *CP2001*, pages 93–108. Springer, November 2001.
7. E.C. Freuder. Eliminating interchangeable values in constraint satisfaction problems. In *AAAI 91*, pages 227–233, 1991.
8. D. Frost and R. Dechter. Look-ahead value ordering for constraint satisfaction problems. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI'95*, pages 572–578, Montreal, Canada, 1995.
9. R. M. Haralick and G.L. Elliott. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14:263–313, 1980.
10. P. Prosser. Hybrid algorithms for the constraint satisfaction problem. *Computational Intelligence*, 9:268–299, 1993.
11. J.-F. Puget. Symmetry breaking revisited. In *CP2002*, pages 446–462. Springer, September 2002.
12. T. Schiex and G. Verfaillie. Two approaches to the solution maintenance problem in dynamic constraint satisfaction problems. In *Proc. of the IJCAI-93/SIGMAN Workshop on Knowledge-based Production Planning, Scheduling and Control, Chambery, France, (August 1993)*., 1993.
13. T. Schiex and G. Verfaillie. Nogood Recording for Static and Dynamic Constraint Satisfaction Problem. *International Journal of Artificial Intelligence Tools*, 3(2):187–207, 1994.
14. T. Schiex and G. Verfaillie. Stubbornness: an enhancement scheme for backjumping and nogood recording. In *Proceedings of the 12-th ECAI*, pages 165–169, 1994.
15. P. van Beek. Constraint tightness and looseness versus local and global consistency. *Journal of the ACM*, 44(4):549–566, 1997.
16. R. Weigel and B. Faltings. Structuring techniques for constraint satisfaction problems. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, pages 418–423, Nagoya, Japan, aug 1997. Morgan-Kaufmann.