Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

J.W. Spee

Finding all minimal covers of a set using implicit enumeration

Computer Science / Department of Software Technology Report CS-R9007 February

---

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Research (N.W.O.).

Finding All Minimal Covers of a Set Using Implicit Enumeration

J.W. Spee*
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

February 22, 1990

Abstract

In diagnostic reasoning, an association exists between hypotheses and their observable consequences in a knowledge base. Now a diagnosis of a set of observations consists of a set of hypotheses, such that the union of their associated sets of consequences minimally covers this set of observations. We describe an algorithm that finds all such minimal covers.

An *implicit enumeration* algorithm is used, which is an adapted version of similar algorithms used for the Set Covering Problem presented in the literature. Of this algorithm a correctness proof is given.

We prove completeness properties for our minimal cover problem of two search space reductions that can be found in the literature. We also introduce another reduction, which is based on a partitioning of the search space.

Finally, we show two heuristics, one of which can reduce the computation time of the algorithm considerably, and we prove it to be as least as strong as one of the search space reductions.

1980 Mathematics Subject Classification (1985) : 68Q20, 68Q25, 68R05
1987 CR Categories : F.2.2, G.2.1
Key Words & Phrases : Diagnostic Inference, Set Covering, Implicit Enumeration, Combinatorial Search, Algorithm Analysis

*The work in this document was conducted as part of the PRISMA project, a joint effort with Philips Research Eindhoven, partially supported by the Dutch "Stimulerings-projectteam Informatica-onderzoek" (SPIN).

Report CS-R9007
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Contents

1 Introduction

This report concerns the description of an algorithm for computing all minimal covers of a set. Such an algorithm is of significant importance in abductive diagnostic reasoning (cf. [1,2,3,4]). In particular, this algorithm is currently used in the EQUIP expert system (cf. [4]), in which abductive reasoning is applied in the domain of silicon chip production control.

In an abductive diagnostic inference system a knowledge base can be modelled by a relation $K \subseteq H \times O$, where H is a set of hypotheses and O a set of observations, such that
1 - for all $h \in H$ there is at least one $o \in O$ such that $(h, o) \in K$,
2 - for all $o \in O$ there is at least one $h \in H$ such that $(h, o) \in K$, and
3 - $H \cap O = \emptyset$.

For each hypothesis $h \in H$ we call the set of related observations in the knowledge base $Cl(h) = \{o | (h, o) \in K\}$ the **cluster** *associated with* h. Given a set $W \subseteq O$ of real world observations, we restrict each cluster to its intersection with W. If there are two different hypotheses h_1 and h_2 in H such that $Cl(h_1) = Cl(h_2)$, we regard h_1 and h_2 as equivalent.

Now a *diagnosis* of W consists of a set of hypotheses M such that the union of their associated clusters minimally covers W, that is,
$\bigcup\{Cl(h)|h \in M\} = W$ and $\forall i \in M \bigcup\{Cl(h)|h \in (M - \{i\})\} \neq W$.
This is where our minimal cover algorithm comes in. If we know all minimal covers of W, then we immediately also know all diagnoses of W.

Now that we have shown the setting of our algorithm we will restrain ourselves to the problem of finding all minimal covers.
More about abductive diagnostic inference can be found in [1,2], in which a slightly different approach is followed: all *minimum covers* are searched for, which are covers with a minimal cardinality. These form a subset of all minimal covers.

First we will give some important definitions. Then we will show a basic algorithm, using *implicit enumeration*. This algorithm is proven correct. Hereafter we demonstrate four search space reductions, two of which have been presented in literature about the Set Covering Problem (cf. for example [5,6]), in which some minimum cover is searched for. We will prove completeness properties of two of these reductions for our minimal cover problem. Finally we will present two heuristics and a formal property of one of them.

2 Preliminaries

In short, the set covering problem is this:
given a set W, and a set of sets S, such that each element of S has a non-empty intersection with W, and their union is a superset of W. Now all minimal subsets D of S have to be found, such that the union of the elements in D is still a superset of W. D is called a *minimal cover* of W.

Now we give some basic definitions and properties of the set covering problem :

Definition 2.1 Let $W = \{w_1, \ldots, w_n\}$ be a set of observations, and
$S = \{X_1, \ldots, X_k\}$ a set of sets, W and S finite, such that $X_i \cap W \neq \emptyset$, $i = 1, \ldots, k$.
S **covers** W if $\bigcup S \supseteq W$, that is, $X_1 \cup \ldots \cup X_k \supseteq W$, or, in other words, S is a **cover** of W. If also $\bigcup S \subseteq W$, then S **exactly covers** W and S is an **exact cover** of W.

Definition 2.2 Let W and S be defined as above.

S **minimally covers** W if S covers W and $\forall X \in S \ \bigcup(S - \{X\}) \not\supseteq W$. Then S is called a **minimal cover** of W.

Theorem 2.1 Let S be a set of sets that covers a set W. Then there exists a minimal cover of W, which is a subset of S.
Proof. Given: $\bigcup S = W$.
step 1: If $\forall X, X \in S$, $\bigcup(S - \{X\}) \neq \bigcup S$, then S is a minimal cover of W by definition.
step 2: Otherwise, $\exists X \in S$ such that $\bigcup(S - \{X\}) = \bigcup S$. Now omit this set X from S, or to put it differently, $S := S - \{X\}$, and goto step 1. □

Theorem 2.2 If S covers W, then restricting each element of S to its intersection with W gives an exact cover of W.
Proof. $S = \{X_1, \ldots, X_k\}$, $\bigcup S \supseteq W$.
$S' = \{X'_1, \ldots, X'_k\}$, where $X'_i = X_i \cap W$, $i = 1, \ldots, k$.
To prove that $W \subseteq \bigcup S'$, take an arbitrary $w \in W$.
Now $w \in \bigcup S$, so $w \in X_j$, for some X_j. Therefore also $w \in X'_j$ and $w \in \bigcup S'$. So $W \subseteq \bigcup S'$.
To prove that $\bigcup S' \subseteq W$, take an arbitrary $w \in \bigcup S'$.
Now $\exists X'_l$ with $w \in X'_l$. $X'_l = X_l \cap W$, so $w \in W$. Therefore $\bigcup S' \subseteq W$.
It follows that S' exactly covers W. □

From now on we assume that the sets that can be used to cover a set W are subsets of W, which automatically results in exact covers. So where we use '(minimal) cover', it could be replaced by '*exact* (minimal) cover'.

Finding a minimal cover can also be formulated in an integer linear programming style:

- $W = \{w_1, \ldots, w_n\}$ is the set to be covered, $S = \{X_1, \ldots, X_k\}$ is the set of sets used to cover W.
- $x_i = 1$ iff X_i is in the cover, $i = 1, \ldots k$.
- $A = (a_{ij})$ is an n by k matrix with $a_{ij} = 1$ if $w_i \in X_j$, otherwise $a_{ij} = 0$.
- find all $(x_1, \ldots, x_k)$, such that:
- $A\,(x_1, \ldots, x_k) \geq (1, \ldots, 1)$, where the inequality is taken pointwise.

- $\forall j,\ j = 1, \ldots, k$, if $x_j = 1$ then
  $A(x_1, \ldots, x_{j-1}, 0, x_{j+1}, \ldots, x_k) \not\geq (1, \ldots, 1)$.

In section 4 we shall use this representation quite often.

Definition 2.3 Let S be a set of sets.
S is **independent** if $\forall X \in S \ \bigcup(S - \{X\}) \neq \bigcup S$ (see [4]).

The following lemma follows immediately from the above:

Lemma 2.1
- A minimal cover of a set is independent.
- A independent cover of a set is a minimal cover of that set.

The cardinality of a set X will be denoted as $|X|$.

Definition 2.4 Let S be a set of sets. The **coverrate of an element w in S**, denoted as $\mathbf{cov}(w, S)$, is the number of sets in S of which w is an element, that is, $\text{cov}(w, S) = |\{X \in S | w \in X\}|$.

Theorem 2.3 Let S be a set of sets.
S is independent iff $\forall X \in S\ \exists w \in X\ \text{cov}(w, S) = 1$.
Proof. S is independent
iff $\forall X \in S\ \bigcup(S - \{X\}) \neq \bigcup S$
iff $\forall X \in S\ \exists w \in X$ such that $w \notin \bigcup(S - \{X\})$
iff $\forall X \in S\ \exists w \in X\ \text{cov}(w, S - \{X\}) = 0$
iff $\forall X \in S\ \exists w \in X\ \text{cov}(w, S) = 1$. □

Lemma 2.2 A subset of an independent set is independent.
Proof. Let D be an independent set and D' a subset of D.
Then by theorem 2.3, $\forall X \in D'\ \exists w \in X\ \text{cov}(w, D) = 1$.
Since $\text{cov}(w, D') \leq \text{cov}(w, D)$, also $\text{cov}(w, D') = 1$, for $X \in D'$.
Therefore, again by theorem 2.3, D' is independent. □

Corollary 2.1 Every subset of a minimal cover is independent.

Definition 2.5 Let D be a independent set of sets, and S a set of sets.
Then $D' = D \cup S$ is an **extension** of D.
D' is an **independent extension** of D if D' also is independent.

Definition 2.6 A **partial cover** of a set W is an independent set, the union of which is a subset of W.

Definition 2.7 Let W be a set and S be a set of sets that covers W. Then $MC(D', S)$ is the set of all minimal covers D of W, such that:
- D' is a subset of D.
- the independent extension D of D' is formed using a subset of S.

So the set of all minimal covers of W, using a set of sets C that covers W, is $MC(\emptyset, C)$.

It turns out that we need an algorithm to compute $MC(\emptyset, C)$, that is, we have to find all independent extensions of the empty set that are covers of W. In the next section we give a basic algorithm to do this, and this algorithm will be improved later on.

From now on we shall use the following standard identifiers:
W is the set of elements to be covered, C is the given set of sets that can be used to cover W, D is a partial cover of W.
When no confusion arises, we shall sometimes say that an element of W is covered, meaning that the singleton set containing this element is covered.

3 Basic Enumeration Algorithm

The *search space* of the SCP consists of all subsets of C. Simply enumerating all subsets is not feasible, for $2^{|C|}$ sets have to be considered.
A more efficient enumeration method must be used, that still finds all minimal covers, but that needs to check a minimal number of subsets.
A well-known technique to achieve this is *implicit enumeration*, a strategy that leads, as the enumeration proceeds, to the exclusion of large parts of the search space, that do not need any further consideration.
With implicit enumeration partial solutions are generated and simultaneously all completions of each are considered, which explains the term *implicit*.

In the literature, with respect to implicit enumeration methods for set covering, two basic approaches are taken. The first is constructing all subsets of C, and prune whenever a full cover has been reached. We shall call this "subset enumeration search". The second approach is in each step trying to increase the number of covered elements of W, by augmenting all current partial solutions with sets that contain a not yet covered element. We shall call this "increasing cover search".

We shall investigate these two approaches more deeply and improve upon both of them. This results in two enumerative algorithms that look rather similar. Of the second type of algorithm a correctness proof will be given,

and this algorithm will be the basis of further improvements introduced later on.

Both algorithms are recursive, which means that in the algorithm a call to itself is made, but with changed parameters. Therefore, as the computation proceeds, more than one instance of the algorithm exists. We call such an instance a *recursive instance.*

Using subset enumeration search (see also [5,7,8]), a binary tree is constructed, where every node in the tree is a recursive instance of the search procedure with a partial cover as one of its parameters. In every node a set X in C, that has not been inspected yet, is used to extend such a partial cover. In the son node on the right the partial cover is extended with X and removed from C, in the son node on the left X is simply removed from C. The enumeration scheme looks as follows, where W' is the set to be covered, C' is the set of sets that covers W' and D' is a partial cover of W that has to be extended with a subset of C':

Subset_Enumeration_Search(W', C', D'):
begin
  if $C' \neq \emptyset$
  then
    select a set X in C';
    Subset_Enumeration_Search(W', $C' - \{X\}$, D');
    if $W' - X \neq \emptyset$
    then Subset_Enumeration_Search($W' - X$, $C' - \{X\}$, $D' \cup \{X\}$)
    else if $D' \cup \{X\}$ is independent **then** report $D' \cup \{X\}$ **fi**
    fi
  fi
end

The initial call to this algorithm is "Subset_Enumeration_Search(W, C, $\emptyset$)". We assume that W and C are initially not empty.

Three improvements can be made quite easily. First, in the left son node all that has changed is the exclusion of a set X from C'. This can easily be incorporated in the father node for all X in C. Secondly, if $D' \cup \{X\}$ is not an independent extension of D', then, by Corollary 2.1, it has no use to pursue the search any further in this direction. As third improvement, it can be seen that by excluding a set X from C', the possibility arises that the

remaining sets do not cover the set $W' - X$. When this happens, searching can also be pruned. This results in the following improved algorithm:

```
Subset_Enumeration_Search( W', C', D'):
begin
    C'' := C';
    for all X in C'
    do
        D'' := D' ∪ {X};
        W'' := W' - X;
        C'' := C'' - {X};
        if W'' ≠ ∅
        then
            C''' := {X ∈ C'' | D'' ∪ {X} is independent };
            if ⋃C''' ⊇ W''
            then Subset_Enumeration_Search(W'', C''', D'')
            fi;
        else report D''
        fi
    od
end
```

In the second approach, "increasing cover search" (see also [3,4,6,9]), an element of the set to be covered is taken, and each partial cover is extended with a set that covers this selected element. The enumeration scheme looks as follows:

```
Increasing_Cover_Search( W', C', D'):
begin
    select an element w ∈ W';
    for all X ∈ C' such that w ∈ X
    do
        if W' - X ≠ ∅
        then
            Increasing_Cover_Search( W' - X, C' - {X}, D' ∪ {X})
        else if D' ∪ {X} is independent then report D' ∪ {X} fi
        fi
    od
end
```

The initial call to this algorithm is "Increasing_Cover_Search(W, C, $\emptyset$)". We assume that W and C are initially not empty.

For this algorithm also three improvements can be made.
First, only sets that can be used for an independent extension need to be passed to a next recursive instance of the algorithm. As second improvement we can make the following observation: each time that a pass through the statements in the **for**-loop has been made, one set X has been tried to extend the current partial cover. Now we know that this set X, if the extension is independent, will appear in every minimal cover which is an extension of D', the partial cover we started with in this instance of the algorithm. So X is for further investigation no longer needed, and need not to be passed any more. It turns out that with each pass through the **for**-loop, we can omit the last investigated set from further consideration.
This optimization appears to be new. Now, as third optimization, it is possible that the union of $C' - \{X\}$ does not cover $W' - X$. In this case, no more minimal covers will be found, and here we can cancel searching any further. Now we will show the resulting improved algorithm:

```
Increasing_Cover_Search( W', C', D'):
begin
    C'' := C';
    select an element w ∈ W';
    for all X ∈ C' such that w ∈ X
    do
        D'' := D' ∪ {X};
        W'' := W' - X,
        C'' := C'' - {X};
        if W'' ≠ ∅
        then
            C''' := {X ∈ C'' | D'' ∪ {X} independent };
            if ⋃C''' ⊇ W''
            then Increasing_Cover_Search(W'', C''', D'')
            fi
        else report D''
        fi
    od
end
```

A nice property of both improved algorithms is that every minimal cover is found exactly once, unlike the original increasing cover algorithm, which can find the same minimal cover several times. Both improved algorithms look rather similar. With increasing cover search, first an element in W' is chosen, and then an iteration through the elements in C' is made. With subset enumeration search, it happens the other way around: an iteration through the elements in C' is made, and for each set in C' it is known that it contains an yet uncovered element (this has been checked implicitly in the independence check of the calling instance). Furthermore, with each iteration in subset generation search, a set X in C' is removed. With increasing cover search this only happens when the selected element w is contained in X.
Finally, with increasing cover search, the selected element w can be chosen in such a way that hopefully a minimal number of recursive instances have to be created, unlike subset generation search, where no such choice exists.

It can be concluded that both algorithms have some advantages and disadvantages.

From now on we will use the improved increasing cover algorithm as basic implicit enumeration method for the rest of the paper. This choice will become clear later on. Now we give a correctness proof of this algorithm by proving three lemma's, in which W is a non-empty set that is covered by a set C.

Lemma 3.1 In a computation of 'Increasing_Cover_Search(W, C, $\emptyset$)', for each set D, with $|D| = m$, for some m,
D appears as third parameter in a call 'increasing_cover_search(W_D, C_D, D)', or D is reported as a minimal cover of W
iff
1- D is a partial cover of W, and
2- there is a sequence of calls 'Increasing_Cover_Search(W_i, C_i, D_i)',
$0 \leq i \leq m-1$, with $|D_i| = i$, $D - D_i \subseteq C_i$, $D_i \cup \{X\}$ is independent for every set $X \in C_i$, $W_i = W - \bigcup D_i$, $C_0 = C$, $W_0 = W$, $D_0 = \emptyset$, such that D is chosen as extension of D_{m-1} in the recursive instance with D_{m-1} as third parameter.

Proof. Let W and C be as given. We will use induction over the cardinality of D.
Basis: $|D| = 0$, that is, $D = \emptyset$: straightforward, for there is a call 'Increasing_Cover_Search(W, C, $\emptyset$)', namely the initial call. The empty set is a

partial cover of W and the sequence consists of this call itself. Also $\emptyset \cup \{X\}$ is independent for every set $X \in C$.

Induction Step: Assume the lemma has been proven for $|D| = m$.
Now look at a set D with $|D| = m + 1$.
D appears in a call 'Increasing_Cover_Search(W_D, C_D, D)' or D is reported as a minimal cover of W
iff
there is a recursive instance with as third parameter D' that is extended to D, with $|D| = m$
iff (by the induction hypothesis)
D' is a partial cover of W
and there is a sequence of calls 'Increasing_Cover_Search(W_i, C_i, D_i)', $0 \leq i \leq m-1$, such that $|D_i| = i$, $D' - D_i \subseteq C_i$, $D_i \cup \{X\}$ is independent for every set $X \in C_i$, $W_i = W - \bigcup D_i$, $C_0 = C$, $W_0 = W$, $D_0 = \emptyset$, where D' is chosen as extension of D_{m-1} in the recursive instance with D_{m-1} as third parameter
and D' appears in a call 'Increasing_Cover_Search($W_{D'}$, $C_{D'}$, D')'
such that $|D'| = m$, $D - D' \subseteq C_{D'}$, $D' \cup \{X\}$ is independent for every set $X \in C_{D'}$, $W_{D'} = W - \bigcup D'$
iff
there is a sequence of calls 'Increasing_Cover_Search(W_i, C_i, D_i)', $0 \leq i \leq m$, such that $|D_i| = i$, $D - D_i \subseteq C_i$, $W_i = W - \bigcup D_i$, $C_0 = C$, $W_0 = W$, $D_0 = \emptyset$, where D is chosen as extension of $D_m = D'$ in the recursive instance with D_m as third parameter, and D is a partial cover of W, and every set in C_m can be used for an independent extension of D_m.

So the induction hypothesis holds for $|D| = m + 1$.
This completes the proof of the lemma. □

Definition 3.1 Let D be a partial cover that is constructed in a recursive instance with as third parameter D_m, then we call the sequence $D_0, \ldots, D_m$ the *history* of D, with D_i, $0 \leq i \leq m$, the third parameter in the call 'Increasing_Cover_Search(W_i, C_i, D_i)' in the sequence that led to the construction of D.

Lemma 3.2 In a computation 'Increasing_Cover_Search(W, C, $\emptyset$)' each partial cover $D \subseteq C$ of W with $|D| = m$ appears in a call 'increasing_cover_search(W_D, C_D, D)' once at most or is reported as a minimal cover of W once at most.

Proof.
We will use induction over the cardinality of D.
Basis: $|D| = 0$, that is, $D = \emptyset$: straightforward, for there is a call 'Increasing_Cover_Search(W, C, $\emptyset$)', and this is the only call with $\emptyset$ as third parameter, for in each recursive call of the algorithm the cardinality of the partial cover strictly increases.
Induction Step: Assume the lemma has been proven for $|D| = m$.
Now look at a partial cover D with $|D| = m + 1$.
Assume by contradiction that D is reported more than once or there exists more than one call with D as third parameter.
Now there are at least two recursive instances of the algorithm were this D has been constructed. Say these recursive instances did have parameters W'_1, C'_1, D'_1 and W'_2, C'_2 and D'_2. Now $|D'_1| = |D'_2| = m$, and by the induction hypothesis we have that they appear in a call exactly once. Using Lemma 3.1 and Definition 3.1, let the history of D'_1 be $D'_{1,1}, \ldots, D'_{1,m-1}$ and let the history of D'_2 be $D'_{2,1}, \ldots, D'_{2,m-1}$. Let $D'_{1,p}$ be the largest partial cover in the history of D'_1, such that $D'_{1,p} = D'_{2,p}$, $1 \leq p \leq m - 1$.
By the induction hypothesis and Lemma 3.1, we know that $D'_{1,p}$ appears in a call 'Increasing_Cover_Search($W'_{1,p}$, $C'_{1,p}$, $D'_{1,p}$)' exactly once. Now look at this recursive instance of the algorithm.
Assume that $X \in D'_{1,p+1} - D'_{1,p}$ is chosen first in the for-loop for extending $D'_{1,p}$. Then also $X \in D'_1$.
Now X is removed from C'', the remaining set of sets in the algorithm.
In some later iteration $X' \in D'_{2,p+1} - D'_{2,p}$ is chosen for extending $D'_{1,p}$. Now X cannot be passed as a member of C''', for it has already been removed from C''. But $X \in D'_{1,p+1} \subset D$. So $D'_{2,p+1}$ can never be extended to D. Contradiction.
The same argument holds if $X' \in D'_{2,p+1} - D'_{2,p}$ was chosen first.
So there exists at most one recursive instance with D'_1 as third parameter, in which D is constructed. Now this D is reported as a minimal cover or it appears in a call 'increasing_cover_search(W_D, C_D, D)'.
So the induction hypothesis has been proven for $|D| = m + 1$.
This completes the proof of the lemma. □

Lemma 3.3 In a computation 'Increasing_Cover_Search(W, C, $\emptyset$)' each set D is reported iff D is a minimal cover of W.
Proof.
"⇒": Assume D is reported. Then by Lemma 3.1, D is a partial cover of W. If we look at the algorithm and Lemma 3.1, then we see that D is only

reported if $\bigcup D = W$. Therefore by Lemma 2.1 D is a minimal cover of W.
"$\Leftarrow$": Let $D \subseteq C$ be a minimal cover of W. We will prove by induction on l that the following property holds: if $l < |D|$, then there exists a call 'Increasing_Cover_Search(W_l, C_l, D_l)' such that $|D_l| = l$, $D - D_l \subseteq C_l$, $W_i = W - \bigcup D_l$, D_l a partial cover of W, $D_l \cup \{X\}$ is independent for every set $X \in C_l$.
Basis: $l = 0$: This is the initial call. D is a subset of C, so D can be constructed out of the partial cover $\emptyset$ using C. Also $\emptyset \cup \{X\}$ is independent for every set $X \in C$.
If $|D| = 1$, then D consists of exactly one set X, such that $X = W$. Therefore no matter which element in W is selected, X is chosen to extend the partial cover $\emptyset$, which gives the minimal cover D.

Induction Step: Assume the property holds for some $l < |D|$.
Now, if $l < |D| - 1$, we prove there is such a call for $l + 1$.
Look at the recursive instance with W_l, C_l and D_l as parameters. Now $\bigcup(D - D_l) = W_l$, so whatever element $w \in W_l$ is selected, in this recursive instance an extension D_{l+1} of D_l is constructed which is a subset of D. D_{l+1} is independent, for every set in C_l can be used as an independent extension of D_l. Therefore D_{l+1} is a partial cover of W. $|D_{l+1}| < |D|$ and $D_{l+1} \subseteq D$, so D_{l+1} is not a cover of W, and every set in $D - D_{l+1}$ is passed in C'''.
Now the lemma has been proven for $|D| = l + 1$.

Now we have proven that there exists a sequence of calls, the last one of which has parameters $W_{|D|-1}$, $C_{|D|-1}$ and $D_{|D|-1}$, such that $|D_{|D|-1}| = |D| - 1$, $D - D_{|D|-1} \subseteq C_{|D|-1}$, $W_{|D|-1} = W - \bigcup D_{|D|-1}$, and $D_{|D|-1}$ is a partial cover of W. Now there is a set $X \in C_{|D|-1}$ such that $X = W_{|D|-1}$, so whatever element w is selected, $D_{|D|-1}$ is extended to D. $\bigcup D = W$, so $W_{|D|-1} - X = \emptyset$ and D is reported as a minimal cover. □

Corollary 3.1 The improved increasing cover algorithm finds every minimal cover exactly once.
Proof. This follows immediately from Lemma 3.2 and 3.3 □

It has been proven that the algorithm finds all minimal covers. However, it can still be improved in a number of ways. When C satisfies some special properties then some elements can be removed without losing any minimal cover. We will discuss these properties in the next section and proof their correctness.

Also we can see that in the current algorithms no particular element $w \in W'$ is taken as next element used for extending a partial cover. Moreover, the order in which sets are investigated in the **for**-loop has not been specified. For this purpose we shall introduce in Section 5 some heuristics that can make the algorithm considerably more efficient.

4 Search Space Reductions

4.1 Introduction

If it would be possible to reduce C, such that no minimal covers are lost, and only partial covers are generated that can be extended to a minimal cover, then we would have an implicit enumeration algorithm, which is in some sense optimal: it would simply construct all minimal covers, without doing any redundant processing.
This can be achieved if at each recursive instance of the algorithm with partial cover D' and remaining set of sets C', the two following sets can be identified:
the set $\bigcap(MC(D', C')) - D'$, which contains all sets that occur in every minimal cover containing D', and the set $C' - \bigcup(MC(D', C'))$, which contains all sets of C' that occur in no minimal cover containing D'.
The former set can simply be added to the partial cover D', because any of its members is part of every minimal cover containing D'. Every set contained in the latter set will not appear in any minimal cover containing D'. So these sets can be eliminated from C' without any further consideration.

At each recursive instance of the algorithm these two sets should be computed exactly. We exhibit three *reductions*, two of which can also be found in literature [6,7,8,9]. There they are mainly used for the *minimum* cover problem. We shall prove that these reductions identify $\bigcap(MC(D', C')) - D'$ and $C' - \bigcup(MC(D', C'))$ exactly if D' is empty and $C' = C$, and otherwise subsets of these, which is also useful.
Using our improved implicit enumeration algorithm, these reductions can be useful in every recursive instance of the algorithm, and even in every iteration of the **for**-loop.

We also introduce another reduction, that can also be of use in every recursive instance, namely *partitioning* the remaining set of sets:
if the set of sets C' can be partitioned into sets $C'_1, \ldots, C'_k$, such that $\bigcup C'_i \cap \bigcup C'_j = \emptyset$, for all pairs i, j, $1 \leq i, j \leq k$, $i \neq j$, then at most

$2^{|C'_1|} + \ldots + 2^{|C'_k|}$ subsets have to be generated, which is less than $2^{|C'|}$, for $k > 1$.

First we will introduce a technique to reduce the set that has to be covered.

Definition 4.1 Given a recursive instance of the algorithm with parameters W' and C'.
Let $E \subseteq W' \times W'$, such that $w_1 E w_2$ iff $\forall X \in C'\ w_1 \in X \leftrightarrow w_2 \in X$.
E defines an equivalence relation.
Data compaction consists of chosing of each equivalence class one representative and removing all other elements from W'.

Whenever an element $w \in W$ is chosen to be covered in the algorithm, all elements in the same equivalence class as w are covered. It follows that only one element of each equivalence class is needed in W in the computation, and all other elements of W can be omitted.

In the following subsections we will discuss how (subsets of)
$\bigcap(MC(D', C')) - D'$ and $C' - \bigcup(MC(D', C'))$ can be computed. Computing $C' - \bigcup(MC(D', C'))$ will be preceded by the introduction of *dominated elements.*

4.2 Computing $\bigcap(MC(D', C')) - D'$

If a set X in C contains an element that is not contained in any other set in C, then X is needed in any minimal cover, otherwise this element in X would never be covered. So X is a member of every minimal cover and therefore X is also in the intersection of all minimal covers.
We shall prove that these sets, which contain some element exclusively, exactly define $\bigcap(MC(\emptyset, C))$, but that they in general only form a subset of $\bigcap(MC(D', C'))$, if D' is not empty.

Lemma 4.1 Let X be an element of $C' \subseteq C$.
$X \in MC(D', C')$ iff for all minimal covers $D \supseteq D'$,
$\exists w \in X\ \ \mathrm{cov}(w, D) = 1$.
Proof. $X \in MC(D', C')$ iff (by definition) for all minimal covers D, such that D' is a subset of D, is X an element of D
iff (by Theorem 2.3 and Lemma 2.1) for all minimal covers D, such that D' is a subset of D: $\exists w \in X\ \mathrm{cov}(w, D) = 1$. □

Now we prove that $\bigcap(MC(\emptyset, C))$ consists of exactly those sets that contain some element exclusively.

Theorem 4.1 Let X be an element of C.
Then $X \in \bigcap(MC(\emptyset,\mathrm{C}))$ iff $\exists w \in X$ $\mathrm{cov}(w, C) = 1$.
Proof.
"$\Rightarrow$": From $X \in \bigcap(MC(\emptyset,\mathrm{C}))$ and Lemma 4.1, we have
$\forall D \in MC(\emptyset,\mathrm{C})$ $\exists w \in X$ such that $\mathrm{cov}(w, D) = 1$.
Assume $\neg\ \exists w \in X$ $\mathrm{cov}(w, C) = 1$. Then $\forall w \in X\ \exists Y \in C, Y \neq X : w \in Y$.
It follows that $\exists D \subset C - \{X\}$, such that D is a minimal cover of W (by Theorem 2.1) and $X \notin D$, so $X \notin MC(\emptyset, C)$. Contradiction.
"$\Leftarrow$": $\exists w \in X$ $\mathrm{cov}(w, C) = 1$ implies $\forall Y \in C$, if $Y \neq X$ then $w \notin Y$.
Also $\forall D \in MC(\emptyset, C)$ $\bigcup D = W$. It follows that $\forall D \in MC(\emptyset, C)$ $X \in D$.
Therefore $X \in \bigcap(MC(\emptyset, C))$. □

Theorem 4.2 Let X be an element of $C' \subset C$.
Then $\exists w \in X$ $\mathrm{cov}(w, C') = 1$ implies $X \in \bigcap(MC(D', C'))$, $D' \neq \emptyset$, but not conversely.

Proof.
"$\Rightarrow$": $\exists w \in X$ $\mathrm{cov}(w, C') = 1$ implies $\forall Y \in C'$, if $Y \neq X$ then $w \notin Y$.
Also $\forall D \in MC(D', C')$ $\bigcup D = W$. It follows that
$\forall D \in MC(D', C')$ $X \in D$. Therefore $X \in \bigcap(MC(D', C'))$.
"$\not\Leftarrow$": We give a counterexample. We shall use the ILP-formulation of the minimal cover problem. Look at the matrix A: $A = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$.
Each column is a set. Each row represents an element. Assume we have a partial cover D' that consists of the set A_1. Now $C' = \{A_2, A_3, A_4\}$. $MC(D', C')$ consists of exactly one minimal cover, namely $\{A_1, A_2\}$. So $A_2 \in \bigcap(MC(D', C')) - D'$. But there is no element in A_2 that has a coverrate of 1 in C'. □

Now we have proven that whenever in the remaining set of sets used to extend a partial cover D'', there is a set X that contains an element with a coverrate of 1, D'' can be extended with X.

Reduction 1 Given a recursive instance of the algorithm with parameters W', C' and D'. In every iteration of the **for**-loop, let D'' be the extension

of D' and C''' the remaining set of sets. for all $X \in C'''$, such that $\exists w \in X$ $\text{cov}(w, C''') = 1$, do $D'' := D'' \cup \{X\}$, $C''' := C''' - \{X\}$ and $W'' := W'' - \{X\}$.

This reduction is in general only capable of finding a subset of $\bigcap(MC(D', C')) - D'$, if D' is not empty.

4.3 Dominated Elements

Suppose, for some recursive instance, that for some $w_1, w_2 \in W'$, for every X set in C', if $w_1 \in X$ implies $w_2 \in X$. Then whenever w_1 is an element of the union of an extension of the current partial cover, then also is w_2.
In literature [9,10] this is called *domination* and defines a relation "$<_{C'}$".

Definition 4.2 Let $<_{C'} \subseteq W' \times W'$, such that $w_1 <_{C'} w_2$ (w_1 dominates w_2) iff $\forall X \in C'$ $w_1 \in X \rightarrow w_2 \in X$.

Lemma 4.2 If data compaction has been applied, then "$<_{C'}$" is a strict partial order.
Proof.
Transitivity is trivial. We look at antisymmetry:
$w_1 <_{C'} w_2 \rightarrow w_2 \not<_{C'} w_1$:
Assume $w_1 <_{C'} w_2$ and $w_2 <_{C'} w_1$. Then $\forall X \in C'$ $w_1 \in X \leftrightarrow w_2 \in X$.
Contradiction, for no such elements exist after data compaction has been applied. □

It can be concluded that in the algorithm only minimal elements of W' in the partial order "$<_{C'}$" are needed. So every non-minimal (that is, dominated) element can be omitted from W'.

Definition 4.3 Let in every iteration of the **for**-loop in the algorithm C'''' be the remaining set of sets and W'' the remaining set to be covered. **Dominated elements removal** consists of removing every element $w \in W''$ such that there is another element $w' \in W''$ and $w' <_{C''''} w$. This can be done using the following algorithm:

begin
  while $\exists w, w' \in W''$ such that $w' <_{C'''} w$
  do
    select $w \in W''$ such that $\exists w' \in W'' : w' <_{C'''} w$;
    $W'' := W'' - \{w\}$;
    $C''' := \{X - \{w\} \mid X \in C'''\}$
  od
end

4.4 Computing $C' - \bigcup(MC(D', C'))$

In every recursive instance of the algorithm we could start searching for dominated elements. When a set in C' only consists of such elements, this set need not be used in any extension of a partial cover. So this set can be removed.

We prove that $C - \bigcup(MC(\emptyset, C))$ is exactly defined by such sets. Whenever $D' \subset C$ is not empty, they form only a subset of $C' - \bigcup(MC(D', C'))$. In this case, all sets in C' that cannot be used in an independent extension of D' are also in $C' - \bigcup(MC(D', C'))$.

Theorem 4.3 Let X be an element of C.
Then $X \in C - \bigcup(MC(\emptyset, C))$ iff
$\forall w \in X\ \exists w' \notin X$, $w' \in W$, such that $w' <_C w$.
Proof.
"$\Rightarrow$": Assume by contradiction there is a $w \in X$ such that $\forall w' \notin X$
$\exists Y \in C$ such that $w \notin Y$ and $w' \in Y$.
Let $S_w = \{Y \mid \exists w' \notin X : w \notin Y \text{ and } w' \in Y\}$.
Then S_w covers $W - X$, but $w \notin \bigcup S_w$, and there is a minimal cover $S'_w \subset S_w$ of $W - X$ by Theorem 2.1.
So $S'_w \cup \{X\}$ is a minimal cover of W, and $X \in \bigcup(MC(\emptyset, C))$. Contradiction.
"$\Leftarrow$": Let w be an arbitrary element of X and let $D \in MC(\emptyset, C)$.
Assume $\exists w' \notin X$, $w' \in W'$ and $w' <_C w$. So $\forall Y \in C\ w' \in Y \rightarrow w \in Y$.
Now $\exists Y \in D$ such that $w' \in Y$, for $\bigcup D = W$. So also $w \in Y$. Now assume by contradiction $X \in D$. But then $\forall w \in X\ \exists Y \in D$ such that $w \in Y$, for w was arbitrarily chosen. It follows that D is not independent. Contradiction, so $X \notin D$. Therefore $X \in C - \bigcup(MC(\emptyset, C))$, for D is arbitrarily chosen. □

Theorem 4.4 Let X be an element of C' and $D' \neq \emptyset$.

Then $\forall w \in X\ \exists w' \notin X\ w' <_{C'} w$ implies $X \in C' - \bigcup(MC(D', C'))$, but not conversely.

Proof.

"$\Rightarrow$": Let $D \in MC(D', C')$, and let w be an arbitrary element of X. Assume $\exists w' \notin X$, $w \in W'$ and $w' <_{C'} w$. So $\forall Y \in C'\ w' \in Y \rightarrow w \in Y$. Now $\exists Y \in D - D'$ such that $w' \in Y$. So also $w \in Y$. Now assume by contradiction $X \in D$. Then $\forall w \in X\ :\ \exists Y \in D$ such that $w \in Y$. It follows that D is not independent. Contradiction, so $X \notin D$. Therefore $X \in C' - \bigcup(MC(D', C'))$, for D is arbitrarily chosen. □

"$\not\Leftarrow$": We use the same counterexample as in Theorem 4.2. Assume we have a partial cover D' that consists of the set A_1. Now $C' = \{A_2, A_3, A_4\}$. $MC(D', C')$ consists of exactly one minimal cover, namely $\{A_1, A_2\}$. So A_3 and A_4 are elements of $C' - \bigcup(MC(D', C'))$, but they do not only exist of non-minimal elements. □

This results in the following reduction:

Reduction 2 In every iteration of the **for**-loop in the algorithm with C'''' as remaining set of sets, apply *dominated elements removal.* For all $X \in C''''$, if $X = \emptyset$, do $C'''' := C'''' - \{X\}$.

Every set X in C', such that $D' \cup \{X\}$ is not an independent extension of partial cover D', does not appear in any minimal cover that is a superset of D', by Corollary 2.1. So these sets are not included in C'''', the subset of C' that is passed to a next recursive instance of the algorithm. This has already been included in our improved enumeration algorithm.

Theorem 4.5 Let D'' be a partial cover and C'' the set of sets that can be used to extend D''. Then we have for all $X \in C''$, if $D'' \cup \{X\}$ is not independent then $X \in C'' - \bigcup(MC(D'', C''))$.

Proof. Immediately by Corollary 2.1. □

Reduction 3 In every iteration of the **for**-loop in the algorithm, with D'' as partial cover and C'' as set of sets that can be used to extend D'', restrict C'' to $C'''' = \{X \in C'' \mid D'' \cup \{X\}$ independent $\}$.

4.5 Partitioning the Remaining Search Space

If in a recursive instance of the algorithm, with parameters W', C' and D', the set C' can be partitioned into subsets, the union of which is pairwise dis-

joint, then all minimal covers of W' can be found by combining all minimal covers of all partitions, and omitting the ones that are not independent.

Lemma 4.3 Let $\bigcup C' \supseteq W'$, and $C' = C'_1 \cup \ldots \cup C'_p$, such that $\bigcup C'_i \cap \bigcup C'_j = \emptyset$, for all $i, j = 1, \ldots, p$, $i \neq j$.
Then $W' = W'_1 \cup \ldots \cup W'_p$, where $W'_i = \bigcup C'_i \cap W'$, $i = 1, \ldots, p$.
Proof. $W' = \bigcup C' \cap W' = \bigcup (C'_1 \cup \ldots \cup C'_p) \cap W' =$
$(\bigcup C'_1 \cap W') \cup \ldots \cup (\bigcup C'_p \cap W') = W'_1 \cup \ldots \cup W'_p$. □

By definition, it follows that the W_i are pairwise disjunct, therefore W is partitioned into p pairwise disjoint subsets .

Theorem 4.6 Let $\bigcup C' \supseteq W'$, and $C' = C'_1 \cup \ldots \cup C'_p$, such that $\bigcup C'_i \cap \bigcup C'_j = \emptyset$, for all $i, j = 1, \ldots, p$, $i \neq j$.
Then for all $D \in MC(D', C')$, $D = D' \cup D_1 \cup \ldots \cup D_p$, such that $D_i \subseteq C'_i$ and D_i is a minimal cover of W'_i.
Proof. From Lemma 4.3 follows that for all j, $j = 1, \ldots, p$, only subsets of C'_j can cover W'_j.
Therefore, by Theorem 2.1 every minimal cover D_j of W'_j is a subset of C'_j.
So every minimal cover $D \in MC(D', C')$ consists of the union of the partial cover D' and of minimal covers D_j of W_j, $j = 1, \ldots, p$. □

If the set C' can be partitioned, then finding all minimal covers comes down to computing every possible combination of D' with a partial cover D_j of each partition, $j = 1, \ldots, p$, provided that such a combination is independent.

Reduction 4 In a recursive instance of the algorithm with D' as partial cover of W' and C' as set of sets, if $C' = C'_1 \cup \ldots \cup C'_p$, such that the $\bigcup C'_i$ are pairwise disjoint, then report as a minimal cover every D, such that $D = D' \cup D_1 \cup \ldots \cup D_p$, where D_j is reported as a minimal cover by a recursive instance of the algorithm with parameters $\bigcup C'_j$, C'_j, $\emptyset$, for each $j = 1, \ldots, p$, provided that D is an independent extension of D'.

4.6 Applicability of the reductions

In all reductions it is stated that they should be applied in every recursive instance of the procedure and in each iteration of the **for**-loop. This does not mean that they always yield a reduction of the search space, but that they *might* yield such a reduction:
with every iteration, a set in C' is removed. Also sets that cannot be used

for an independent extension of the partial cover are not taken into account. Therefore, the structure of C' is changed, and applying the reductions might be fruitful.

Example:

Look at the matrix $A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$,

which defines a set W consisting of 5 elements and 6 sets that cover it. We omit the subscript from the relation "$<$", as long as no ambiguity occurs.

Assume element w_1 (represented by the first row) is selected first. Look at the recursive instance of the algorithm where $\{A_1\}$ is taken as partial cover. Then $\{A_1, A_2\}$ is not independent, so A_2 is omitted, that is, $A_2 \in \{A_2, \ldots, A_6\} - \bigcup(MC(\{A_1\}, \{A_2, \ldots, A_6\}))$. Now $w_2 < w_4$ and $w_3 < w_5$, therefore w_4 and w_5 are dominated elements and are removed. But now A_6 consists completely of dominated elements and is removed. Now assume w_2 is selected next, and the partial cover is extended to $\{A_1, A_3\}$. Then $\{A_1, A_3, A_5\}$ is not independent, so A_5 is removed. As a consequence, $\mathrm{cov}(w_3, \{A_4\}) = 1$, so $A_4 \in \bigcap(MC(\{A_1, A_3\}, \{A_4\}))$, and the partial cover is extended to the minimal cover $\{A_1, A_3, A_4\}$.

This example shows that applying the reductions as described in the previous subsections can reduce the search space very quickly.

5 Heuristics

Now we introduce two heuristics, one of which has shown to reduce the number of recursive instances of the algorithm considerably in practice [9,11]. This does not mean, of course, that by using these heuristics, it can be proven that a minimal number of recursive instances is generated.
The first heuristic selects some special element of W' is selected, and the second in what order the elements of C' are dealt with in the **for**-loop.

Good results are obtained by chosing an element of W' that has a minimal coverrate in C'. Informally, this can be justified as follows: in each recursive instance another element of W is covered. When to achieve this, a minimal number of recursive instances is generated, it seems plausible that

for the whole cover of W a number of recursive instances is generated that is nearly minimal. Although we are not able to prove this, we will show that by applying this strategy dominated elements are removed automatically. Therefore, sets that consist of dominated elements only, which are element of $C' - \bigcup((MC(D', C'))$, are never used to extend a partial cover. As a consequence, recursive instances with such an extended partial cover as third parameter, which would provably yield no minimal cover, are not generated. This forms a possible explanation for the succes of this heuristic in practice.

Lemma 5.1 Let C' and W' be given, and $\bigcup C' \supseteq W'$.
Then for all $w_1, w_2 \in W'$ $w_1 <_{C'} w_2$ implies $\text{cov}(w_1, C') < \text{cov}(w_2, C')$.
Proof. Let $w_1 <_{C'} w_2$. $\forall w \in W'$ let $S(w) = \{X \in C' \mid w \in X\}$. Then $\text{cov}(w, C') = |S(w)|$. Now $\forall X \in S(w_1)$ $w_2 \in X$. Therefore, $|S(w_2)| \geq |S(w_1)|$. Assume $|S(w_2)| = |S(w_1)|$. Then $S(w_2) = S(w_1)$, and as a consequence, $w_2 <_{C'} w_1$. Contradiction, for $<_{C'}$ is a partial order. So $|S(w_1)| <_{C'} |S(w_2)|$ and $\text{cov}(w_1, C') < \text{cov}(w_2, C')$. □

This leads to the following heuristic:

Heuristic 1 *Greedy Search*: In each recursive instance of the algorithm with parameters W', C' and D', select an element in W' with a minimal coverrate in C'.

Theorem 5.1 Given a recursive instance of the algorithm with parameters W', C' and D'. If Greedy Search is applied, then D' is never extended to $D' \cup \{X\}$, $X \in C'$, such that X only consists of dominated elements.
Proof. Assume that as selected element an element $w \in W'$ is chosen such that $w \in X$, for some $X \in C'$, and X consists of dominated elements only. But then there is a $w' \in W'$, such that $w' <_{C'} w$. By Lemma 5.1, we know that $\text{cov}(w', C')) < \text{cov}(w, C')$. So if Greedy Search is applied, w would not be selected. Contradiction.
As a consequence, a set $X \in C'$ that consists of dominated elements is not used for extending the partial cover D'. □

It seems that when Greedy Search is applied, reduction 2 becomes redundant. However, this is not the case. If, using reduction 2, sets consisting of dominated elements are removed, possibly more partitions can be found, which reduces the search space. Therefore reduction 2 is still useful.

Now we deal with a second heuristic, that supports Greedy Search. Until now, the order in which the sets in C' are tried for extending a partial cover is not specified. We can try to optimize this order, by chosing such a set X in C', among the set of all sets in C' that contain the selected element, that X contains an element $w' \in \bigcup L$ with a minimal coverrate in C'. In the recursive instances that are created using sets in L selected after the selection of X, X is not an element of the set C'''. As a consequence, the coverrate of w', which already was the lowest in $\bigcup L$, is reduced by 1. It is possible that now its coverrate becomes 1, in which case reduction 1 can be applied. Otherwise we have an element with a decreased coverrate, which might be of use in a future recursive instance.

Heuristic 2 *Greedy_Supporting Order*: Given a recursive instance of the algorithm with parameters W', C' and D'. Let $w \in W'$ be the selected element by Greedy Search. $L = \{Y \mid w \in Y \text{ and } Y \in C'\}$.
If $\exists Y \in L$ such that $\exists w' (\neq w) \in Y : \forall w'' \in \bigcup L - \{w\} :$
$\text{cov}(w', C') \leq \text{cov}(w'', C')$, then choose Y as next set in the **for**-loop.
Otherwise, choose an arbitrary $Y \in L$.

The chosen set Y is removed from C', so in the next iteration L its size has decreased. It follows that each element of L is chosen once to extend the partial cover D'.

6 Conclusion

We have shown a set covering algorithm to find all minimal covers, consisting of an enumeration method, improved by four reductions and two heuristics. The enumeration method is more efficient than similar methods presented in the literature, and is proven to be correct.

Given a partial cover, two sets are identified using three reductions: sets that appear in any minimal cover containing the partial cover, and sets that appear in no such minimal cover. These reductions can be seen as supplementing the enumeration method.
Another reduction is used to partition the search space in parts that can be treated independently.

The reductions can identify $\bigcap(MC(\emptyset, C))$ and $C - \bigcup(MC(\emptyset, C))$ completely. However, if D' is not empty, they only find subsets of $\bigcap(MC(D', C')) - D'$ and $C' - \bigcup(MC(D', C'))$, so probably some redundant processing is still performed.

The effectiveness of the reductions can be order dependent. It seems wise to start applying reductions that remove as much redundant sets as possible, that is, apply reductions 3 and 2 first. Hereafter reduction 1 can be applied. Reduction 4, the partitioning reduction, can be applied before or after the other three reductions.

Two heuristics are given, one of which has been applied succesfully in practice. This heuristic is proven to be as least as strong as the second reduction, which indicates that it can be useful in any practical set covering application.
The second heuristic is entirely new, and supports the first one.

The major advantage of taking "increasing cover search" as basic enumeration method is that we can apply heuristics to reduce the number of recursive instances that are generated. If "subset enumeration search" is used, no effective heuristics are known.

Further research is needed to find stronger reductions that can exactly identify $\bigcap(MC(D', C')) - D'$ and $C' - \bigcup(MC(D', C'))$. Then the role of the heuristics will possibly become less important.

Acknowledgement

In this paper a lot of use has been made of work (to be published) by
H.J. ter Horst on the EQUIP expert system.

References

[1] Y. Peng (1986). *A Formalization of Parsimonious Covering and Probabilistic Reasoning in Abductive Diagnostic Inference*, Ph.-D. Thesis, Department of Computer Science, TR-1615, University of Maryland.

[2] J.A. Reggia, D.S. Nau, P.Y. Wang (1985). A Formal Model of Diagnostic Inference I. *Information Sciences*, vol. 37, pp. 227-256.

[3] J.A. Reggia, D.S. Nau, P.Y. Wang (1985). A Formal Model of Diagnostic Inference II. *Information Sciences*, vol. 37, pp. 257-285.

[4] H.J. Ter Horst (1990). *Paper on EQUIP*, Philips Research Laboratories Eindhoven, The Netherlands.

[5] A.M. Geoffrion (1967). Integer Programming by Implicit Enumeration and Balas' Method, *SIAM Review*, Vol. 9, No. 2, pp. 178-190.

[6] N. Christofides (1975). *Graph Theory, An Algorithmic Approach*, Academic Press, New York, pp. 30-57.

[7] R. Garfinkel, G.L. Nemhauser (1972). *Integer Programming*, John Wiley & Sons, New York. Class of Set-Covering Algorithms, *SIAM Journal of Computing*, Vol. 12, No. 2, pp. 329-346.

[8] V. Lifschitz, B. Pittel (1983). The Worst and the Most-Probable Performance of a Class of Set-Covering Algorithms, *SIAM Journal of Computing*, Vol. 12, No. 2, pp. 329-346.

[9] M.H. Young, S. Muroga (1985). Minimal Covering Problem and PLA Minimization, *International Journal of Computer and Information Sciences*, Vol. 14, No. 6, pp. 337-364.

[10] C.E. Lemke, H.M. Salkin, K. Spielberg (1971). Set Covering by Single Branch Enumeration with Linear-Programming Subproblems, *Operations Research*, vol. 19, pp. 998-1022.

[11] J.W. Spee (1989). *A Parallel Implementation of a Part of EQUIP in POOL2*, Philips Internal Report, Philips Research Laboratories Eindhoven, The Netherlands.