# Algorithms for Computing *X*-Minimal Models

Chen Avin and Rachel Ben-Eliyahu – Zohary

Communication Systems Engineering Department,
Ben-Gurion University of the Negev
Beer-Sheva 84105, Israel
{avin,rachel}@bgumail.bgu.ac.il

**Abstract** The problem of computing *X*-minimal models, that is, models minimal with respect to a subset *X* of all the atoms in a theory, is very relevant for computing circumscriptions and diagnosis. Unfortunately, the problem is NP-hard. In this paper we present two novel algorithms for computing *X*-minimal models. The advantage of these new algorithms is that, unlike existing ones, they are capable of generating the models one by one. There is no need to compute a superset of all minimal models before finding the first *X*-minimal one. Our procedures may use local serach techniques, or, alternatively, complete methods. We have implemented and tested the algorithms and the preliminary experimental results are encouraging.

## 1 Introduction

Minimal model computation is a crucial task in many reasoning systems in Artificial Intelligence, including Logic Programming, Nonmonotonic Reasoning, and Diagnosis [Re87,Mc80,dKW87]. Indeed, a considerable effort has been made to analyze the complexity of this task and to build efficient algorithms and systems that solve it [e.g. BD96,KL99,.JNS00].

In this paper, we consider a more general computational problem- the problem of computing *X*-minimal models. When we look for *X*-minimal models, we search for models that are minimal with respect to a subset *X* of all the atoms in the theory. The task of computing minimal models is a special case of generating *X*-minimal models, taking *X* to be all the atoms in the theory. *X*-minimal models are particularly relevant in Diagnosis and Circumscription [Re87,Mc80,Li85]. In the logical approach to Diagnosis, the artifact to be diagnosed and the behavior of the system are encoded as a set of logical sentences called the system description and the observations, respectively. The components of the system are represented by constants, and their status – whether or not they are functioning well - is indicated by a special predicate called an abnormality predicate and denoted *ab(.)*. Normally, we assume that all the abnormality predicates are **false**, that is, that components in the system well behave. Once there is a fault, the theory composed of the system description and the observations becomes logically inconsistent if we assume that none of the components is abnormal. To resume consistency, we must assume that some components are malfunctioning. We prefer to explain the inconsistency with a

T. Eiter, W. Faber, and M. Truszczyński (Eds.): LPNMR 2001, LNAI 2173, pp. 322-335, 2001.

minimal subset of abnormalities. It does not make sense to assume that a set of components are broken when the assumption that only a subset of this set is malfunctioning can explain the faulty behavior of the system. This is called "The Principle of Parsimony". By this principle, we assume that only minimal subsets of components are faulty, or in other words, we look for models that are minimal with respect to the abnormality predicates.

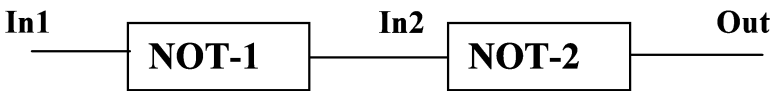


**Fig. 1.** An example circuit

The systems descriptions and observations are usually represented in first order logic. For the sake of simplicity, we will use propositional logic in this paper. The algorithms presented here can be used for function-free first-order minimal model computation by first grounding the theories involved. Alternatively, the algorithms shown here can serve as a basis for developing algorithms tailored for first-order logic.

As an example for model-based diagnosis using minimal models, consider the simple circuit shown in Figure 1. Assuming *AB1* and *AB2* mean that gate "not-1" and "not-2", respectively, are malfunctioning, the system description (*SD*) for this gate is:

$$\neg AB1 \rightarrow [In1 \leftrightarrow \neg In2]$$
$$\neg AB2 \rightarrow [In2 \leftrightarrow \neg Out]$$

Now, assume that In1 is 0 and Out is 1, indicating that the circuit is faulty. The observations (*OBS*) in this case are $\{\neg In1, Out\}$. If we assume that both gates are normal and take the theory that is the union of *SD*, *OBS*, and the literals $\{\neg AB1, \neg AB2\}$, - we get an inconsistent theory. However, if we consider the theory consisting of the union of *SD* and *OBS* alone, and we look at the *X*-minimal models of this theory taking *X* to be $\{AB1, AB2\}$, we obtain two such models, in each of which only one gate is abnormal. Hence the diagnosis for the above system and observations is that either the first or the second (but not both) circuit is faulty.

The circuit example also illustrates why a demand-driven computation of the *X*-minimal models is advantageous. Each *X*-minimal model explains the faulty behavior of the system by suggesting a minimal set of components that may be abnormal. If we are given the models one by one, we can test the suspect components while the next model is being computed.

The paper is organized as follows. After presenting some basic definitions and known results in Section 2, we present two demand-driven algorithms for *X*-minimal model computation in Section 3. Both algorithms may be implemented either using local search methods or complete methods. *X*-minimal models are also very relevant in computing Circumscription. We elaborate on that in Section 4. In Section 5 we report on experiments done on the algorithms developed and in Sections 6 and 7 we present related work and concluding remarks.

## 2 Preliminary Definitions

In this section we provide some basic terminology used throughout the paper.

- *Literal* – propositional symbol (atom) (positive literal) or its negation. (negative literal).
- *Clause* – disjunction of literals.
- *CNF theory* – conjunctive normal form, a conjunction of clauses. All the theories in this paper are assumed to be in CNF. Hence by *theory* we mean a set of clauses. A theory is *Horn* if and only if each clause in the theory contains at most one positive literal.
- *Positive graph of a theory* - Let $T$ be a theory. The positive graph of $T$ is an undirected graph (V, E) defined as follows: $V = \{ P | P$ is a positive literal in some clause in $T\}$, E= $\{(P, Q) | P$ and $Q$ appear positive in the same clause$\}$.
- *Vertex cover* - Let $G = (V, E)$ be a graph. A vertex cover of G is a set $V' \subseteq V$ such that for each $e \in E$ there is some $v \in V'$ such that $v \in e$.
- *Vertex cover of a theory* – Vertex cover of the poistive graph of the theory. Note that if all the atoms of a vertex cover of a theory are instantiated, the theory becomes Horn.
- *Model* – a truth assignment to all the atoms in the theory that makes the theory true.
- *Pos (M)* – the set of the atoms assigned **true** in a model $M$.
- *Lit(v)* – A representation of a truth assignment $v$ as a set of literals. For example, if $v$=*{P*=**true,** *Q*=**false,** *R*=**false***}*, then *Lit(v)={P,¬Q,,¬R}*.
- *Unit clause* – clause that contains only one literal.
- *Unit propagation* - the process where given a theory $T$, you do the following until there is no change in the theory (no new clauses are generated and no clause is deleted): you pick a unit clause C from $T$, delete the negation of C from each clause and delete each clause that contains C.
- $T \otimes S$ – is the result of unit propagation on $T \bigcup$ S.
- {**true** (**false**)}– set of atoms that are assigned with **true** (**false**).
- $Int_x(M)$ – the value (integer) of a model $M$ over a given ordered set of variables $X=\{P_r,\ldots,P_0\}$, seen as a binary number where $P_r$ is the most significant bit (MSB or MSV - most significant variable) and $P_0$ is the least significant bit (LSB or LSV - least significant variable). So, for example, if *M={P*=**true,** *Q*=**false,** *R*=**false***}* then $Int_{\{P,Q\}}(M)$=(10 in binary code) = 2; $Int_{\{P,Q,R\}}(M)$=(100 in binary code) = 4.
- *X-Largest (Smallest) model* – the model with the largest (smallest) value ($Int_x(M)$) with respect to a given ordered set of variables $X=\{P_r,\ldots,P_0\}$.

### *X*-minimal model

Let $T$ be a theory over a set of atoms $L$, $X \subseteq L$, and $M$ a model for $T$. $M$ is an *X-Super* of another model $M'$ if and only if pos($M$ ')$\cap X$ is a proper subset of pos$(M) \cap X$.

*M* is an *X-minimal model* for *T* if and only if there is no other model *M*' for *T* such that *M* is an *X-Super* of *M*'. If *M* is an *X*-minimal model for $X = L$, it will be called simply a minimal model. It has been shown that a Horn theory has a unique minimal model that can be found in linear time [DG84].

**Find-*X*-minimal** (*T,X,M*)

**Input:** A theory *T*, an ordered set $X=\{P_r,...P_1\}$ which is a subset of the atoms in *T*.

**Output:** true if *T* is satisfiable, false otherwise. In case *T* is satisfiable, the output variable *M* is a smallest *X* –minimal model of *T* w.r.t. the ordering $\{P_r,...P_1\}$.

**1.** If ¬sat(*T,M*) return false;

**2.** For i : = r downto 1 do

a.If isHorn(*T*) then *M*=HornMinimalModel(*T*), Goto 4.

b.If sat ($T \bigcup \{\neg P_i\}$ , M) then $T: =T \otimes \{\neg P_i\}$
else $T : =T \otimes \{P_i\}$

**3.** sat(*T,M)*

**4.** Return true;

**Fig. 2.** Algorithm Find-*X*-minimal

**Example 2.1** Suppose a theory $T_0$ has variables $P_6...P_0$, and $X=\{P_6,...P_3\}$. Assume further that $T_0$ has exactly four models (ordered from the *X*-smallest to the *X*-largest): $M_1=0010110$, $M_2=0011000$, $M_3= 0100111$, and $M_4=1100000$. $M_1$ and $M_3$ are the only *X*-minimal models of $T_0$.

Throughout this paper, unless stated otherwise, models that agree on the truth assignments given to all the variables in *X* are considered identical.

The following theorem is quite interesting. Its proof is based on results from Combinatorics [Bo86]. We provide in the appendix a proof suggested by Lomonosov [Lo00].

**Theorem 2.2**: Let *T* be a theory and let *X* be a subset of the atoms that are used in *T*. The number of *X*-minimal models of *T* is at most $\binom{n}{\lfloor n/2 \rfloor}$, where $|X|=n$.

**S-*X*-Min** ($T$, $X$)

**Input**: A theory $T$. An ordered set $X=\{P_r,...P_1\}$ which is subset of the variables in $T$.

**Output**: **true** if $T$ is satisfiable, **false** otherwise. If $T$ is satisfiable, output one by one all $X$-minimal models of $T$ from the smallest to the largest w.r.t the order $\{P_r,...P_1\}$.

1. If $\neg$Sat$(T,M)$ return **false**.
2. *ModelsTable* = $\varnothing$.
3. For $i$ : = 0 to $2^r$-1 do:
   a. $v$: = instantiation of $X$ that equal $i$. ($P_1$ least significant, $P_r$ most significant).
   b. If $v$ is not an $X$-super of a model in *ModelsTable* then
      if Sat$(T \bigcup Lit(v), M)$
         Output ($M$);
         Add $M$ to *ModelsTable;*

**Fig. 3.** Algorithm S-*X*-Min

## Find *X*-Minimal

In Figure 2 we show an algorithm for computing one $X$-minimal model for a theory. The algorithm takes $O(|X|)$ steps and uses $O(|X|)$ calls to a satisfiability testing procedure. A similar algorithm was shown in [BD96]. Find-$X$-minimal tries to assign as many **false** as possible and calls a Horn satisfiability checker once there are enough instantiations so that the theory becomes Horn.

Notes on *Find-X-Minimal* (for future use):

***Note 1:*** if the theory $T$ is not satisfiable then $M$ is returned with the value it was initialized with.

***Note 2:*** The algorithm uses a procedure *sat(T,M)* that returns **true** iff $T$ is satisfiable. In case $T$ is satisfiable, $M$ is a model of $T$. Each model $M$ is an array of booleans, *M[i]* being the truth value assigned to $P_i$. It might be the case that $M$ has entries for variables not appearing in $T$. These variables will be assigned **false** by *sat(T,M).* We do not always use the model that *sat* returns. In implementations, we can use a version of *sat* that does not return a model when we do not need it.

***Note 3:*** If *sat*($T$,$M$) is complete then Find-$X$-Minimal is also complete, otherwise Find-$X$-Minimal is not complete.

## 3 New Algorithms

In this section we will present two algorithms for computing *X*-minimal models. The correctness of these algorithms can be proved only if a complete SAT procedure is assumed. Otherwise the algorithm is not complete and may generate a model which is not *X-minimal.*

The first algorithm, called *S-X-min,* goes over all possible instantiations to *X*, using an ordering having the property that whenever a model is not *X*-minimal, then it must be an *X*-super of an *X*-minimal model already generated. S-*X*-min is shown in Figure 3.

***J-X-min*** (*T*, *X*)

**Input**: A theory *T*, a subset *X* of all the variables in *T*. We assume that $|X|=r$.

**Output**: **true** if *T* is satisfiable, **false** otherwise. In case *T* is satisfiable, output one by one all *X*-minimal models of *T* from the smallest to the largest. Each model is an array of booleans *M, M[i]* is the truth value assigned to $P_i$.

1. Let $P_{n-1}, ...,P_{n-r},...P_0$ be an ordering on the variables in *T* such that the first *r* variables are all the variables from *X*. Variable $P_0$ will be considered the least significant and the variable $P_{n-1}$ *will be* considered the most significant.
2. If Find-*X*-minimal(*T*, $\{P_{n-1}, ...,P_{n-r}\}$, *M*) == **false** return **false**;
3. *ModelsTable* = ∅.
4. Output (*M*); Add *M* to *ModelsTable*.
5. Let *i* be the index of the least significant variable that satisfies:
   1. $P_i \in X$
   2. *M[i]* = **false**
   3. $P_i$ is more significant than another variable $P_j$ such that $P_j \in X$ and *M[j]*=**true;**.

   if there is no such variable return **true**.
6. *M[i]* = **true;**
7. If Find-*X*-minimal ($T \otimes Lit(M[n-1,...i])$, $\{P_{n-1},...P_{n-r}\}$, *M)*=**false** then goto 5.
8. If *M* is not an *X*-super of a model in *ModelsTable* then goto 4. Else goto 5.

**Fig. 4.** Algorithm J-*X*-Min

**Lemma 3.1**: Algorithm S-*X*-Min is correct.

The proof is omitted due to space constraints. The basic argument is that a model which is not *X*-minimal must be an *X*-super of a model that is *X*-smaller. Since the models are generated in an increasing integer ($Int_X$) order, all and only the *X*-minimal models will be generated.

The second algorithm that we present is algorithm *J-X-min* shown in Figure 4. For each theory *T* there is a (possibly empty) set • of all the *X*-minimal models of T. You can order the *n X*-minimal models in • in order $\{M_1,\ldots, M_n\}$ where $M_1$ is the smallest *X*-minimal model and $M_n$ is the largest. The algorithm is based on the following observation:

**Lemma 3.2**: Algorithm Find-*X*-minimal (Figure 2) will always return the smallest *X*-minimal model for some variable ordering. (If exists).

The proof is omitted due to space constraints. Intuitively, the Lemma is true because Find-*X*-minimal tries to assign as many **false** as possible and backtracks on this choice at as less significant bit as possible.

Once we find the first, smallest, *X*-minimal model, we serach for the next one. Suppose that $|X|=4$ and the smallest model is 0100…. There is no point in checking if truth assignmnets starting with 0101 or 0110 are models because it is obvious that they are *X*-super of the first model. Hence the algorithm will "jump" to check whether truth assignmnets starting with 1000 may be models. Hence the "J" in the algorithm name.

**Theorem 3.3**: Algorithm J-*X-Min* is correct.

***Proof****(sketch)*: Let *T* be a theory and let *X* be a subset of its variables. If *T* is inconsistent, the algorithm is clearly correct. Assume *T* is consistent. First, observe that models generated in Step 7 are always generated from the *X*-smallest to the *X*-largest. Let $M_0,\ldots M_k$ be all the *X*-minimal models of *T,* ordered from the *X*-smallest to the *X*-largest according to an ordering $\{P_{n-1},\ldots P_{n-r}\}$ of *X*. We will show by induction on $0 \le t \le k$ that the *t*th model that J-*X*-min outputs is $M_t$.

Base case: Follows from Lemma 3.2.

Case *t>0:* Assume by contradiction that $M' \neq M_t$ is the *t'*th model that the algorithm outputs. By the induction hypothesis, it must be the case that $Int_X(M_{t-1}) < Int_X(M') < Int_X(M_t)$.

Let us look at the last time that Step 5 of the algorithm is exacuted just before model *M'* is sent to output.

Let *i* is the index that the algorithm finds at this last step.

Let *M** be an instantiation of the variables in *T* defined as follows: $M^*[n-1,\ldots,i+1]=M_{t-1}[n-1,\ldots,i+1]$, $M^*[i] = true$, $M^*[i-1,\ldots,n-r]=\{\textbf{false}\}$. It is clear that $Int_X(M_{t-1}) < Int_X(M^*) \le Int_X(M_t)$. There are 2 cases:

1. $Int_X(M_{t-1}) < Int_X(M') < Int_X(M^*)$, then there is contradiction because in this case *M'* must be an *X*-super of $M_{t-1}$ (Some of the variables $(P_{i-1},\ldots,P_{n-r})$ that were **false** become **true** instead), hence the algorithm will not output it.
2. $Int_X(M^*) \le Int_X(M') < Int_X(M_t)$. Then the following must be true:
   2.1 $M'[n-1,\ldots, i] = M_t[n-1,\ldots, i]$
   2.2 $Int(M'[i-1,\ldots,n-r]) < Int(M_t[i-1,\ldots,n-r])$.

But by Lemma 3.1 when we execute *Find-X-minimal* with $X=\{P_{i-1},\ldots, P_{n-r}\}$ it returns the smallest possible value of $\{P_{i-1},\ldots, P_{n-r}\}$ and therefore *M'* cannot be a minimal model that satisfies 2.1 & 2.2, a contradiction.

It is left to show that $M_k$ is the last model sent to output. This is obvious because all the models generated after $M_k$ are not *X*-minimal and hence must be *X*-super of at least one of all the *X*-minimal models that are already in *ModelsTable*. □

**Example 3.4** Consider again theory $T_0$ from Example 2.1, and suppose J-*X*-min is called with $X=\{P_6,\ldots P_3\}$. The first (and smallest) *X*-minimal model returned by Find-*X*-minimal is $M_1=0010110$. After that, at Step 5, we choose $i=5$ and call Find-*X*-minimal ($T_0 \otimes \{\neg P_6, P_5\}$, *X,M)*. Find-*X*-minimal will return model $M_3= 0100111$, which is the 2nd and last minimal model of $T_0$. At Step 5 after that we choose $i=6$ and call Find-*X*-minimal ($T_0 \otimes \{P_6,\}$, *X, M)*. Find-*X*-minimal will return model $M_4= 1100000$. $M_4$ is an *X*-super of $M_3$, and therefore will not be sent to output. In the next iteration, no *i* will be found, and the algorithm will terminate. You can see that out of 16 possible assignments to *X*, only 3 models were considered by J-*X*-min.

## 4 Computing Circumscription

In this section we will show how our algorithms can be used for computing circumscription. First, we will formulate deduction in circumscription in model-theoretic terminology. We will use propositional logic version of definitions from [GPP89,Li85,Mc80]. In this section we assume that *T* is some propositional theory and that there is a partition of all the atoms in *T* into three disjoint sets of atoms: *P,Z,* and *Q*.

**Definition 4.1** [GPP89]. For any two models *M* and *N* of *T* we write $M \bullet N$ mod *(P,Z)* if models *M* and *N* differ only on how they interpret predicates from *P* and *Z* and if pos(*M*)• *P* is a subset of pos(*N*)• *P*. We say that a model *M* is *(P,Z)-minimal* if there is no model *N* such that $N<M$ mod *(P,Z)* (i.e. such that $N \bullet M$ but not $M \bullet N$).

That is, in order for *M* to be *(P,Z)*-minimal, the following must hold: for every model *N* such that *M* and *N* grant the same truth value to all the atoms in *Q,* the set of all atoms in *P* to which *M* assigns **true** must be a subset of the set of all atoms in *P* to which *N* assigns **true**; and we don't care about the truth assignment these models give to atoms in *Z*.

**Theorem 4.2** [GPP89]. For any clause *c,* we say that *c follows from (T,P,Z) by circumscription* if and only if *c* is true in every *(P,Z)*-minimal model of *T*.

**Example 4.3** Assume *T* is the following theory, having the intuitive meaning that children normally like McDonald's, and Itamar is a child:

$$Child \wedge \neg Ab \rightarrow LikesMD.$$
$$Child.$$

Suppose we want to know whether Itamar likes McDonald's. The intuition is that the answer is *yes.* Using classical logic, *LikesMD* does not follow from this theory. However, taking $P=\{Ab\}$ and $Z=\{likesMD\}$ we get that *LikesMD* follows from *(T,P,Z)* by circumscription . To see this, note that *T* has exactly three models:

$M_1$={*Child,Ab,LikesMD*},$M_2$={*Child,Ab,¬LikesMD*},and $M_3$={*Child,¬Ab,LikesMD*}. $M_3$ is the only *(P,Z)*-minimal model of *T*.

The algorithms developed here can be used for a demand-driven computation of (*P,Z)*-minimal models of *T*, in the following way:

1. Use some backtracking algorithm to find all consistent (with *T*) truth assignments for the variables in *Q*
2. For each assignment *v* found in step 1, compute one by one the *P*-minimal models of $T \bigcup Lit(v)$. Each model generated is a *(P,Z)*-minimal model of *T*.

A demand-driven computation is useful here because it may help us refute conclusions before all the models are generated (if a fact does not follow from some model it certainly does not follow from all of the models).

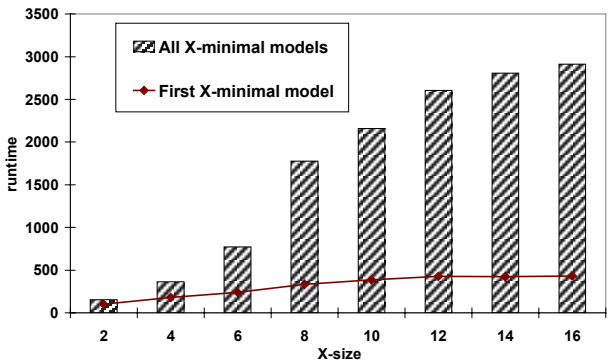


**Fig. 6.** Growing *X* size

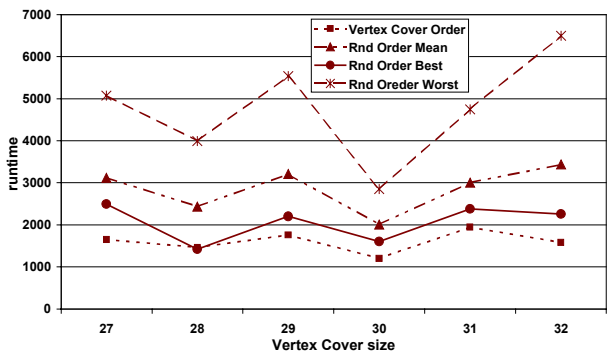


**Fig. 7.** Comparing symbols order heuristics

## 5 Experiments

We have tested algorithm *J-X-min* algorithm on a suite including hard randomly generated CNF problems (theories). The problems are 3CNF difficult random problems as describe in [SKC94] and [SK96]. The Algorithm was tested on theories

of size 50/218 (50 symbols, 218 clauses). The theories were taken from the SATLIB [SAT00]. The algorithms were implemented in JAVA on a PC having Pentium 3 600 MH processor and 128 MB memory. We have chosen JAVA because we had the intention of building an object-oriented library of tools for computing minimal models. We used a JAVA code of *walksat* [SKC94] as a (incomplete) SAT procedure. Since JAVA is a relatively ineffcient language in terms of running time, we did not pay much attention to the absolute running time of the algorithms in these experiments. However, we do believe that running time is an important factor and we plan to implement the algorithms in C in order to improve their time performance.
We have ran three experiments:

1. Compute all the minimal model of the theories and compare the results to results of a complete procedure (the dlv system [KL99] ).
2. Check the growth in run-time as a function of an increasing size of *X*.
3. Compute all the minimal models using different symbols order heuristics.

The first experiment has shown that inspite of using incomplete *sat* algorithms, we have succeeded in computing *all* and *only* the minimal models of the theories. The set of minimal models computed by our algorithm was exactly the same set of minimal models computed by dlv. We expect though that on much larger theories an incomplete algorithm will be less accurate.

Results of the 2nd experiment are shown in Figure 6. As expected, the run-time of the algorithm (given in seconds) is growing as *X* grows. It is encouraging to see that the first *X*-minimal model is generated in about 25% of the time it takes to compute all the models, since one of the goals of this project was to output the models on a demand-driven basis.

The ordering of theory variables before calling algorithm J-*X*-min might be crucial. Once enough instantiations are made so that the theory is Horn, a linear algorithm can be called upon to finish the minimal model computation. In the 3rd experiment we have computed *X*-minimal models where *X* is the set of all variables in the theory. On each theory, we have tested the *J-X-min* five times, four times with random symbol order, and one time with symbol order where the vertex cover of the theory is first in the ordering. In general, the problem of finding a minimum-cardinality vertex cover of a graph is NP-hard. A greedy heuristic procedure that we used for finding a vertex cover simply removes the node with maximum degree from the graph and continues with the reduced graph until all nodes are disconnected. The set of all nodes removed is a vertex cover.

We have compared the run-time results of *J-X-min* in vertex cover order, and in random order. For the random order, we took the best,worst, and mean run time. We have divided the results according to the vertex cover of the theory. The results of this experiment are summarized in Figure 7 (run time is in seconds). We can see that in general the run time of *J-X-min* does not grow as the size of the vertex cover grows. We explain this findings by the fact that we use the *walksat* algorithm. When the *walksat a*lgorithm is checking the satisfiability of an inconsistent theory, it stops after a time-out (measured by number of flips and restarts). This time-out is almost constant and hence the run time of *J-X-min* is more or less constant on theory size with different vertex cover size. We can see that the vertex cover heuristics is quite good, always better than the worst and mean run-time of the *J-X-min* with random order, and usually even better than the best.

## 6 Related Work

During the last few years there have been several studies regarding the problem of minimal model computation. Ben-Eliyahu and Dechter [BD96] have presented several algorithms for computing minimal models, all of them different from the ones proposeded here. One limitation of the algorithms presented there is that they produce a *superset* of all minimal models while we produce the minimal models one by one. Ben-Eliyahu and Palopoli [BP97] have presented a polynomial algorithm for finding a minimal model, but it works only for a subclass of all CNF theories and it finds only one minimal model.

The systems *dlv* [KL99] and *smodels* [JNS00] compute stable models of disjunctive logic programs. If integrity constraints are allowed in the programs, every knowledge base can be represented as a disjunctive logic program such that the set of all minimal models of the first coincide with the set of all stable models of the second. An advatage of our approach compares to theirs is that our algorithms compute *X*-minimal models one at a time while using their approach we have to compute first all minimal models and then select the set of *X*-minimal ones. Another difference is that our implementaion uses local search techniques.

## 7 Conclusions

The task of computing *X*-minimal models is very relevant in many knowledge representation systems, and particularly in Diagnosis and Circumscription. We have presented two new algorithms to perform this task. The algorithms are demand driven, and can be implemented either by using incomplete local search procedures or by using complete procedures. In the future we plan to combine the algorithms presented here with the algorithm developed by [Be00] in order to produce a distributed algorithm for computing *X*-minimal models.

## Acknowledgement

Many thanks to Professor Michael Lomonosov of the Department of Mathematics in Ben-Gurion University, Israel, for helping us with the proof of Theorem 2.2.

## Reference

[Be00] Ben-Eliyahu, R. 2000. A demand-driven algorithm for generating minimal models. In *AAAI-2000: the 17th National Conference on Artificial Intelligence,* pages 101-106.

[BD96] Ben-Eliyahu, R., and Dechter, R. 1996. On computing minimal models. *Annals of Mathematics and Artificial Intelligence* 18:3-27. A short version in AAAI-93: Proceedings of the 11th national conference on artificial intelligence.

[BP97] Ben-Eliyahu, R., and Palopoli, L. 1997. Reasoning with minimal models: Efficient algorithms and applications. *Artificial Intelligence* 96:421-449.

[Bo86] Bollobas, B.; Combinatorics. Cambridge University Press, 1986.

[DG84] William F. Dowling and Jean H. Gallier. Linear time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming*, 3:267-284, 1984.

[dKW87] de Kleer, J., and Williams, B. 1987. Diagnosis multiple faults. *Artificial Intelligence* 32:97-130.

[dKMR92] de Kleer, J.; Mackworth, A.; and Reiter, R. 1992. Characterizing diagnosis and systems. *Artificial Intelligence* 56:197-222.

[GPP89] Gelfond, M.; Przymusinska, H.; and Przymusinski, T. 1989. On the relationship between Circumscription and Negation as Failure, *Artificial Intelligence* 38 75-94.

[JNS00] Janhunen, T.; Niemella, I; Simons, P. and J. You, 2000.Unfolding partiality and disjunctions in stable model semantics. In *KR-2000: Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning*.

[KL99] Koch, C. and Leone, N. 1999. Stable Model Checking Made Easy. In IJCAI-99: Proceedings of the 16th international joint conference on Artificial Intelligence, 70-75.

[Li85] Lifshitz, V. 1985. Computing circumscription. In *IJCAI-85: Proceedings of the international joint conference on AI* 121-127.

[Lo00] Lomonosov, M.; Personal communication., 2000.

[Mc80] McCarthy, J.1980. Circumscription - a form of non-monotonic reasoning. *Artificial Intelligence* 13:27-39.

[Re87] Reiter, R.1987. A theory of diagnosis from first principles. *Artificial Intelligence* 32:57-95.

[SAT00] SATLIB - The Satisfiability Library - http://aida.intellektik.informatik.tu-darmstadt.de/SATLIB/

[SK96] Selman, B. and Kirkpatrick, S. 1996. Critical behavior in the computational cost of satisfiability testing. *Artificial Intelligence*, 81:273-296.

[SKC94] Selman, B. Kautz, H. Cohen, B. Local Search Strategies for Satisfiability Testing. Proceedings of 2nd DIMACS Challenge on Cliques, Coloring and Satisfiability, 1994.

## Appendix

In the following text, unless otherwise is stated, we assume some fixed theory $T$ and some fixed subset $X$ of all atoms in $T$, where $|X|=x$.

We take two truth assignmnets to be different only if they disagree on the set of atoms $X$.

The question we want to raise is: What is the maximum number of different $X$-minimal models that such a theory $T$ may have?

**Definition 1:** An *assignments* (truth assignments*) X-chain (*or $X$-chain in short) is an ordered set of assignments where each assignment is an $X$-s*uper* of the next assignment.

**Definition 2:** an $X$-chain *Set* is a set of $X$-chains such that each possible truth assignment to $X$ appears in exactly one *X-chain.* So there are exactly 2^x assignments in all the $X$-chains all together.

We define $C_r$ as *chains set* that contains exactly *r assignments chains.*

***Lemma 1:*** For any Theory $T$ and set of atoms in $T$, $X$, such that $T$ has a total of $j$ different $X$-minimal models, and for any *X-chain set* $C_r$ for $T$, $r \geq j$.

***Proof:*** We will prove by contradiction that $j$ can't be larger than $r$. It is obvious that two different $X$-minimal models of $T$ must belong to a different $X$-*chain* of $C_r$ (one $X$-minimal model can't be a *Super* of another $X$-minimal model by definition and therefore can't be in the same $X$-*chain*). If $j > r$ then there must be two different $X$-minimal models that belong to the same $X$-*chain* in $C_r$. A contradiction.

***Lemma 2:*** If there is a theory $T$ having exactly $j$ $X$- minimal models and an $X$-*chain set* $C_r$ where $r=j$ then for any theory $T'$ and for any set of atoms $X'$ in $T'$ such that $|X'|=|X|$, $T'$ may have at most $j$ $X'$-minimal models.

***Proof:*** It is easy to see that since $T$ has an $X$-chain set of size $r$, $T'$ must also have an $X'$-chain of size $r$. Assume that $T'$ has $j'$ $X'$-minimal models with $j'>j$. By Lemma 1, $r \geq$ j'. But we also know that $r=j$, so we get that $j \geq$ j'. A contradiction.

**Theorem**: The maximum number of $X$-minimal models of a theory *is* $\binom{n}{\lfloor \frac{n}{2} \rfloor}$, where $|X|=n$.

***Proof:*** First, we will show that there is some theory $T$ having exactly $\binom{n}{\lfloor \frac{n}{2} \rfloor}$ $X$-minimal models for some subset $X$ of all the atoms in $T$ with $|X|=n$. We will define $T$ to be the theory that has exactly $\binom{n}{\lfloor \frac{n}{2} \rfloor}$ models where each model has a different set of $\lfloor \frac{n}{2} \rfloor$ true atoms that belong to some fixed set $X$ of atoms with $|X|=n$, while all the other atoms in the model are assign with **false.** In this case each model is also an x-minimal model.

Next we will show that there is an $X$-*chain set* of size $\binom{n}{\lfloor \frac{n}{2} \rfloor}$. This will complete the proof because it means (according to Lemma 2) that this is the maximum number of $X$-minimal models that a theory may have. We will divide the $2^n$ different assignments to $X$ into the following sets which reflect the number of atoms in $X$ assign true by the assignment:

$$\binom{n}{1}, \binom{n}{2} \ldots \binom{n}{r}, \binom{n}{r+1}, \ldots \binom{n}{\lfloor \frac{n}{2} \rfloor}, \ldots, \binom{n}{n-1}, \binom{n}{n}.$$

We will build the chains in the set as follows. We start with $\binom{n}{1}$ chains, each having one assignment that belongs to the set $\binom{n}{1}$. We then add the assignments in the set $\binom{n}{2}$ to the existing chains, possibly starting a new chain , and so on. The $X$-chains are growing by creating *complete matching* in bipartite graphs where the set of vertices $V$ is the union of the

assignments from $\binom{n}{r}$ and the assignments from $\binom{n}{r+1}$. The edges connecting vertices in this graph reflect the *X*-super relation and we can show that in this case we can find a complete matching.