On Lower Bounding Minimal Model Count

Mohimenul Kabir[a] and Kuldeep S Meel[b]

[a]National University of Singapore
[b]University of Toronto

Minimal Models

- Propositional variable: v takes value either 0 or 1
- Literal: ℓ is either v or $\neg v$
- Clause: C is a disjunction of literals $\bigvee_i \ell_i$
- Formula: F is a conjunction of clauses $\bigwedge_j C_j$
- Assignment: τ assigns each variables $\tau : \mathrm{Var}(F) \to \{0, 1\}$ where $\mathrm{Var}(F)$ denotes the variable set of F
- Model: $\tau \models F$ when τ evaluates F to be 1

Minimal Models

- Propositional variable: v takes value either 0 or 1
- Literal: ℓ is either v or $\neg v$
- Clause: C is a disjunction of literals $\bigvee_i \ell_i$
- Formula: F is a conjunction of clauses $\bigwedge_j C_j$
- Assignment: τ assigns each variables $\tau : \text{Var}(F) \rightarrow \{0, 1\}$ where $\text{Var}(F)$ denotes the variable set of F
- Model: $\tau \models F$ when τ evaluates F to be 1
- Set Notation: $\{a \rightarrow 1, b \rightarrow 0, c \rightarrow 1\} \equiv \{a, c\}$
- Minimal Model: τ is a subset minimal model of F if $\nexists\, \tau_2 \models F$ such that $\tau_2 < \tau$. Intuitively, the minimal set of variables assigned to 1 to satisfy the formula F
- Consider $F = (a \vee b) \wedge (a \vee c)$. The formula F has five models.
  The minimal models of F: $\{a\}, \{b, c\}$.
  Note that the models $\{a, b\}, \{a, c\}, \{a, b, c\}$ are NOT minimal
- Applications: Diagnosis [Reiter1987], Database System [Zaki2004], etc.

Minimal Models

- Propositional variable: v takes value either 0 or 1
- Literal: ℓ is either v or $\neg v$
- Clause: C is a disjunction of literals $\bigvee_i \ell_i$
- Formula: F is a conjunction of clauses $\bigwedge_j C_j$
- Assignment: τ assigns each variables $\tau : \text{Var}(F) \rightarrow \{0,1\}$
  where $\text{Var}(F)$ denotes the variable set of F
- Model: $\tau \models F$ when τ evaluates F to be 1
- Set Notation: $\{a \rightarrow 1, b \rightarrow 0, c \rightarrow 1\} \equiv \{a, c\}$
- Minimal Model: τ is a subset minimal model of F if $\nexists\, \tau_2 \models F$ such that $\tau_2 < \tau$.
  Intuitively, the minimal set of variables assigned to 1 to satisfy the formula F
- Consider $F = (a \vee b) \wedge (a \vee c)$. The formula F has five models.
  The minimal models of F: $\{a\}, \{b, c\}$.
  Note that the models $\{a, b\}, \{a, c\}, \{a, b, c\}$ are NOT minimal
- Applications: Diagnosis [Reiter1987], Database System [Zaki2004], etc.
- Property: *Each of the variables within a minimal model must be justified.*
  For formula $F = (a \vee b) \wedge (a \vee c)$,
  - $\tau = \{a\}$, if a is flipped to false, then it falsifies both of the clauses
- Goal: Lower bounding the number of minimal models of F
  (i.e., lower bounding $|\text{MM}(F)|$)

Answer Set Programming

- Roots in logic programming and nonmonotonic reasoning
- A rule-based language for problem encoding

$$\underbrace{h_1 \vee \ldots \vee h_\ell}_{head} \leftarrow \underbrace{b_1, \ldots, b_k, \sim b_{k+1}, \ldots, \sim b_{k+m}.}_{body}$$

Answer Set Programming

- Roots in logic programming and nonmonotonic reasoning
- A rule-based language for problem encoding

$$\underbrace{h_1 \vee \ldots \vee h_\ell}_{head} \leftarrow \underbrace{b_1, \ldots, b_k, \sim b_{k+1}, \ldots, \sim b_{k+m}.}_{body}$$

- An ASP program $P \equiv$ set of rules.
- The model of P is an *answer set* (denoted as AS(P)).

Answer Set Programming

- Roots in logic programming and nonmonotonic reasoning
- A rule-based language for problem encoding

$$\underbrace{h_1 \vee \ldots \vee h_\ell}_{head} \leftarrow \underbrace{b_1, \ldots, b_k, \sim b_{k+1}, \ldots, \sim b_{k+m}.}_{body}$$

- An ASP program $P \equiv$ set of rules.
- The model of P is an *answer set* (denoted as $\mathrm{AS}(P)$).
- Answer set programming follows the *default negation* — everything is false unless there are some justifications.
- Consider an ASP program, $P = \{\underbrace{a \leftarrow b.}_{r_1}\ \underbrace{b \leftarrow a.}_{r_2}\ \underbrace{s \leftarrow \sim a.}_{r_3}\ \underbrace{a \leftarrow t.}_{r_4}\}$
  - $\{s\} \in \mathrm{AS}(P)$, since s is justified by r_3
  - $\{a, b\} \notin \mathrm{AS}(P)$, since a and b are not justified

From Minimal Models to Answer Set Programming

- We can compute minimal models of a formula by answer set solving
- For a Boolean formula F, we can compute an ASP program $\mathrm{DLP}(F)$ such that $\mathrm{AS}(\mathrm{DLP}(F)) = \mathrm{MM}(F)$

From Minimal Models to Answer Set Programming

- We can compute minimal models of a formula by answer set solving
- For a Boolean formula F, we can compute an ASP program DLP(F) such that AS(DLP(F)) = MM(F)

DLP(F):

- for each clause $C = \ell_1 \vee \ldots \vee \ell_k \vee \neg\ell_{k+1} \vee \ldots \vee \neg\ell_{k+m} \in F$, we introduce a rule to DLP(F) as follows:

$$\ell_1 \vee \ldots \ell_k \leftarrow \ell_{k+1}, \ldots, \ell_{k+m}.$$

From Minimal Models to Answer Set Programming

- We can compute minimal models of a formula by answer set solving
- For a Boolean formula F, we can compute an ASP program DLP(F) such that AS(DLP(F)) = MM(F)

DLP(F):

- for each clause $C = \ell_1 \vee \ldots \vee \ell_k \vee \neg\ell_{k+1} \vee \ldots \vee \neg\ell_{k+m} \in F$, we introduce a rule to DLP(F) as follows:

$$\ell_1 \vee \ldots \ell_k \leftarrow \ell_{k+1}, \ldots, \ell_{k+m}.$$

Example:

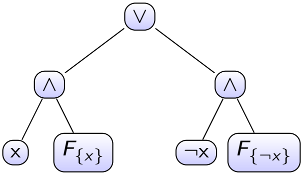
- Consider $F = (a \vee b) \wedge (a \vee c)$
- $\text{DLP}(F) = \{a \vee b \leftarrow .\ a \vee c \leftarrow .\}$
- $\text{AS}(\text{DLP}(F)) = \{\{a\}, \{b, c\}\}$

Knowledge Compilation

- Knowledge compilation [Thurley2006] is an ingredient for model counters (model counting is polytime over the knowledge compilation)

Knowledge Compilation

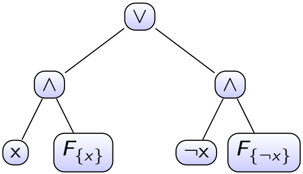
- Knowledge compilation [Thurley2006] is an ingredient for model counters (model counting is polytime over the knowledge compilation)
- **Shannon Expansion**



Unit Propagation: For $x \in \mathrm{Var}(F)$, $\tau \models F_{|\{x\}}$ if and only if $\{x\} \cup \tau \models F$

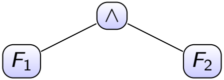
Knowledge Compilation

- Knowledge compilation [Thurley2006] is an ingredient for model counters (model counting is polytime over the knowledge compilation)
- **Shannon Expansion**



Unit Propagation: For $x \in \text{Var}(F)$, $\tau \models F_{|\{x\}}$ if and only if $\{x\} \cup \tau \models F$

- **Component Decomposition**: For two formulas F_1 and F_2 where $\text{Var}(F_1) \cap \text{Var}(F_2) = \emptyset$, it holds that $\tau_1 \models F_1$ and $\tau_2 \models F_2$ if and only if $\tau_1 \cup \tau_2 \models F_1 \wedge F_2$



Challenges in Knowledge Compilation: Minimal Model Counting

Consider the Boolean formula $F = (a \vee b \vee c) \wedge (\neg a \vee \neg b \vee d) \wedge (\neg a \vee \neg b \vee e)$

- $\mathsf{MM}(F) = \{\{a\}, \{b\}, \{c\}\}$
- $\mathsf{MM}(F_{|\{e\}}) = \{\{a\}, \{b\}, \{c\}\}$. But $\{b\} \cup \{e\} \notin \mathsf{MM}(F)$

✕ Unit propagation on minimal model counting DOES NOT work

Challenges in Knowledge Compilation: Minimal Model Counting

Consider the Boolean formula $F = (a \vee b \vee c) \wedge (\neg a \vee \neg b \vee d) \wedge (\neg a \vee \neg b \vee e)$

- $\mathsf{MM}(F) = \{\{a\}, \{b\}, \{c\}\}$
- $\mathsf{MM}(F_{|\{e\}}) = \{\{a\}, \{b\}, \{c\}\}$. But $\{b\} \cup \{e\} \notin \mathsf{MM}(F)$

✕ Unit propagation on minimal model counting DOES NOT work

- $\mathsf{MM}(F) = \{\{a\}, \{b\}, \{c\}\}$
- $F_{|\{a,b\}}$ decomposes into two components containing variables d and e.
- $F_{|\{a,b\}} = d \wedge e$
- $\mathsf{MM}(d) = \{d\}$ and $\mathsf{MM}(e) = \{e\}$
- However, $\{a, b\} \cup \{d\} \cup \{e\} \notin \mathsf{MM}(F)$

✕ Simple Component Decomposition on minimal model counting DOES NOT work

Challenges in Knowledge Compilation: Minimal Model Counting

Consider the Boolean formula $F = (a \vee b \vee c) \wedge (\neg a \vee \neg b \vee d) \wedge (\neg a \vee \neg b \vee e)$

- $\mathrm{MM}(F) = \{\{a\}, \{b\}, \{c\}\}$
- $\mathrm{MM}(F_{|\{e\}}) = \{\{a\}, \{b\}, \{c\}\}$. But $\{b\} \cup \{e\} \notin \mathrm{MM}(F)$

✗ Unit propagation on minimal model counting DOES NOT work

- $\mathrm{MM}(F) = \{\{a\}, \{b\}, \{c\}\}$
- $F_{|\{a,b\}}$ decomposes into two components containing variables d and e.
- $F_{|\{a,b\}} = d \wedge e$
- $\mathrm{MM}(d) = \{d\}$ and $\mathrm{MM}(e) = \{e\}$
- However, $\{a, b\} \cup \{d\} \cup \{e\} \notin \mathrm{MM}(F)$

✗ Simple Component Decomposition on minimal model counting DOES NOT work

Reason: The assignment to variables is NOT justified.
In minimal model counting, unit propagation and component decomposition must be applied on justified assignment

Knowledge Compilation over Justified Assignment

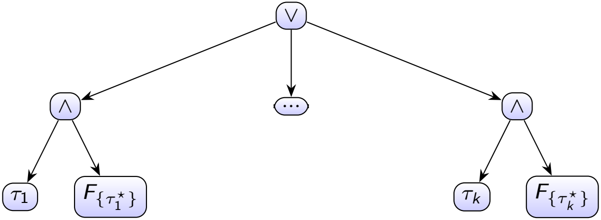
- **Justified Assignment:** Given an assignment τ, the justified assignment $\tau^{\star} = \tau_{\downarrow\{v \in \mathsf{Var}(F) \mid \tau(v)=0\}}$, where "$\downarrow$" denotes the *projection*

Knowledge Compilation over Justified Assignment

- **Justified Assignment:** Given an assignment τ, the justified assignment $\tau^\star = \tau_{\downarrow\{v \in \mathsf{Var}(F) \mid \tau(v)=0\}}$, where "$\downarrow$" denotes the *projection*
- **Cut:** $\mathcal{C} \subset \mathsf{Var}(F)$ such that for every $\tau \in 2^{C}$, the formula $F_{|\tau}$ efficiently *decomposes* into disjoint components (heuristically)

Knowledge Compilation over Justified Assignment

- **Justified Assignment:** Given an assignment τ, the justified assignment $\tau^\star = \tau_{\downarrow\{v \in \mathsf{Var}(F) \mid \tau(v)=0\}}$, where "$\downarrow$" denotes the *projection*
- **Cut:** $\mathcal{C} \subset \mathsf{Var}(F)$ such that for every $\tau \in 2^{C}$, the formula $F_{|\tau}$ efficiently *decomposes* into disjoint components (heuristically)



For each $i \in [1, k]$

- $\tau_i \in 2^{\mathcal{C}}$ and $\tau_1 \vee \ldots \tau_k = \top$
- ✓ Unit propagation and Component decomposition on $F_{\{\tau_i^\star\}}$ preserve the minimal models

Knowledge Compilation: Details

```
Data: Formula F and cut C
Result: |MM(F)|
Algorithm Proj-Enum(F, C)
    cnt ← 0
    B ← ∅ // blocking assignments
    while ∃σ ∈ MinModelswithBlocking(F, B) do
        τ ← σ↓C, d ← 1
        foreach comp ∈ Components(F|τ*) do
            // each disjoint components
            d ← d × |ProjMinModels(F, τ, Var(comp))| // projected enumeration
        end
        cnt ← cnt + d // increment the number of models
        B.add(τ)
    end
    return cnt
```

Knowledge Compilation: Details

Data: Formula F and cut $\mathcal{C}$
Result: $|\mathsf{MM}(F)|$
Algorithm Proj-Enum($F, \mathcal{C}$)

```
cnt ← 0
𝓑 ← ∅ // blocking assignments
while ∃σ ∈ MinModelswithBlocking(F, 𝓑) do
    τ ← σ↓𝓒, d ← 1
    foreach comp ∈ Components(F|τ⋆) do
        // each disjoint components
        d ← d × |ProjMinModels(F, τ, Var(comp))| // projected enumeration
    end
    cnt ← cnt + d // increment the number of models
    𝓑.add(τ)
end
return cnt
```

Implementation Details:

- MinModelswithBlocking($F, \mathcal{B}$): Finding a minimal model of F, where $\mathcal{B}$ denotes all blocking assignments
- ProjMinModels(F, τ, Y): Enumerate $\sigma \in \mathsf{MM}(F)$ such that $\sigma \models \tau$ while projecting onto the variable set of Y
- $\mathcal{C}$: employing tree decomposition

Hashing-based Approximate Minimal Model Counting

Approximate Minimal Model Counting [CMV2013]

- (ϵ, δ)-approximate counting:
  Input: formula F, tolerance ϵ, and confidence δ
  Output: a count c such that

$$\Pr[|\mathsf{MM}(F)|/(1+\epsilon) \leq c \leq |\mathsf{MM}(F)| \times (1+\epsilon)] \geq 1-\delta$$

- **Counter**: invoke ApproxASP [KESHFM2022] on DLP(F)

Hashing-based Approximate Minimal Model Counting

Approximate Minimal Model Counting [CMV2013]

- (ϵ, δ)-approximate counting:
  Input: formula F, tolerance ϵ, and confidence δ
  Output: a count c such that

$$\Pr[{}^{|\mathsf{MM}(F)|}/_{(1+\epsilon)} \leq c \leq |\mathsf{MM}(F)| \times (1+\epsilon)] \geq 1-\delta$$

- **Counter**: invoke ApproxASP [KESHFM2022] on DLP(F)

Probabilistic Lower Bound on Minimal Models

- **Input**: formula F and confidence δ
  Output: a count c such that

$$\Pr[c \leq |\mathsf{MM}(F)|] \geq 1-\delta$$

- **Counter**: invoke a modified ApproxASP on DLP(F)

Hashing-based Minimal Model Counting

Data: Formula F, independent support $\mathcal{X}$, and confidence δ
Result: $|\mathsf{MM}(F)|$
Algorithm HashCount($F, \mathcal{X}, \delta$)

$\alpha \leftarrow -\log_2(\delta) + 1$
generate $|\mathcal{X}|-1$ XORs, namely $Q^1, \ldots, Q^{|\mathcal{X}|-1}$
$\hat{m} \leftarrow$ max k s.t. $\exists \tau \in \mathsf{MM}(F)$ s.t. $\tau \models Q^1 \wedge \ldots Q^k$
return $2^{\hat{m}-\alpha}$

Hashing-based Minimal Model Counting

Data: Formula F, independent support $\mathcal{X}$, and confidence δ
Result: $|\mathsf{MM}(F)|$
Algorithm HashCount($F, \mathcal{X}, \delta$)

$\alpha \leftarrow -\log_2(\delta) + 1$
generate $|\mathcal{X}|-1$ XORs, namely $Q^1, \ldots, Q^{|\mathcal{X}|-1}$
$\hat{\mathsf{m}} \leftarrow \max k$ s.t. $\exists \tau \in \mathsf{MM}(F)$ s.t. $\tau \models Q^1 \wedge \ldots Q^k$
return $2^{\hat{\mathsf{m}}-\alpha}$

Implementation Details:

- HashCount is similar to ApproxASP with thresh $= 1$
- HashCount utilizes a *log search* technique to find the value of $\hat{\mathsf{m}}$
- Compute independent support following the Padoa's theorem [Padoa1901]

Hashing-based Minimal Model Counting

Data: Formula F, independent support $\mathcal{X}$, and confidence δ
Result: $|\mathsf{MM}(F)|$
Algorithm HashCount($F, \mathcal{X}, \delta$)
    $\alpha \leftarrow -\log_2(\delta) + 1$
    generate $|\mathcal{X}|-1$ XORs, namely $Q^1, \dots, Q^{|\mathcal{X}|-1}$
    $\hat{m} \leftarrow \max k$ s.t. $\exists \tau \in \mathsf{MM}(F)$ s.t. $\tau \models Q^1 \wedge \dots Q^k$
    return $2^{\hat{m}-\alpha}$

Implementation Details:

- HashCount is similar to ApproxASP with thresh $= 1$
- HashCount utilizes a *log search* technique to find the value of $\hat{m}$
- Compute independent support following the Padoa's theorem [Padoa1901]

Combining Both Algorithms: MinLB

- If $|\mathsf{Cut}(F)|$ is small, then invoke Proj-Enum
- Otherwise, invoke HashCount

Experimental Result

Baselines:

- Clingo
- ApproxASP
- #MinModels: subtractive approach:
  the number of all models (#P) − the number of non-minimal models (#NP)

[1] https://dtai.cs.kuleuven.be/CP4IM/datasets/

Experimental Result

Baselines:

- Clingo
- ApproxASP
- #MinModels: subtractive approach:
  the number of all models (#P) − the number of non-minimal models (#NP)

Benchmarks:

- Model Counting Competition Benchmarks
- Minimal Generators Benchmark[1]

[1] `https://dtai.cs.kuleuven.be/CP4IM/datasets/`

Experimental Result

Baselines:

- Clingo
- ApproxASP
- #MinModels: subtractive approach:
  the number of all models (#P) − the number of non-minimal models (#NP)

Benchmarks:

- Model Counting Competition Benchmarks
- Minimal Generators Benchmark[1]

Evaluation Metric

$$\mathrm{TQP}(t, C) = \begin{cases} 2 \times \mathcal{T}, & \text{if no lower bound is returned} \\ t + \mathcal{T} \times \frac{1+\log(C_{\min}+1)}{1+\log(C+1)}, & \text{otherwise} \end{cases}$$

[1] https://dtai.cs.kuleuven.be/CP4IM/datasets/

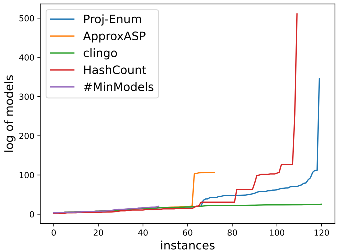
Model Counting benchmark

Clingo	ApproxASP	#MinModels	MinLB (our prototype)
6491	6379	7743	5599

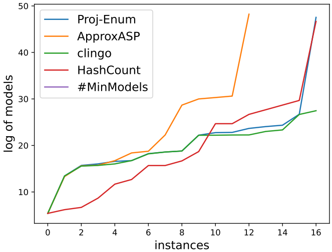
Minimal Generator benchmark

Clingo	ApproxASP	#MinModels	MinLB (our prototype)
6944	5713	9705	5043

Visualization of Lower Bounds

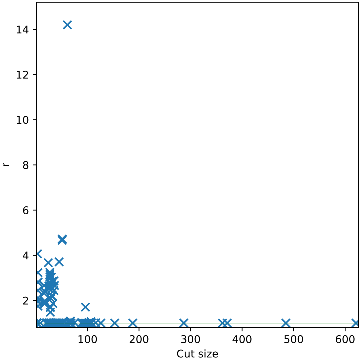


Model counting competition

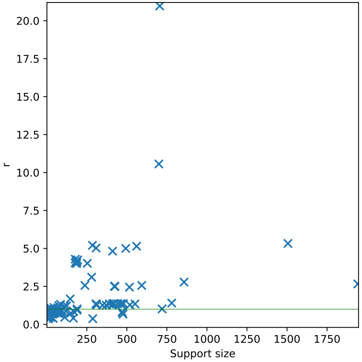


Minimal generator benchmark

Strengths and Weaknesses



Proj-Enum



HashCount

The greater the value of r, the higher the quality.

Conclusion

We propose two methods for lower bounding minimal model count

- Proj-Enum outperforms when the cut size is small
- HashCount scales from small to medium independent support
- MinLB computes better lower bounds than existing systems.

`https://github.com/meelgroup/minLB`