MODEL COUNTING WITH DPLL

Ismet Burak Kadron
November 3, 2016

University of California, Santa Barbara
INTRODUCTION & APPLICATIONS
Motivation: Given a propositional logic formula $F$ in Conjunctive Normal Form (CNF), counting the number $\mu(F)$ of models, assignments of truth values to its variables that satisfy $F$.

Conjunctive Normal Form (CNF): A propositional logic formula where it is a conjunction ($\land$) of clauses, where a clause is a disjunction ($\lor$) of literals.
· Computing the number of solutions of a constraint satisfaction problem.
  · **Constraint Satisfaction Problem**: a mathematical problem defined as a set of objects whose state must satisfy a number of constraints or limitations.
· Computing the number of solutions of a constraint satisfaction problem.
  · **Constraint Satisfaction Problem**: a mathematical problem defined as a set of objects whose state must satisfy a number of constraints or limitations.

· Finding node probabilities in Bayesian belief networks.
DPLL ALGORITHM & MODEL COUNTING
DPLL is a decision procedure for satisfiability of Boolean formulas in conjunctive normal form (CNF-SAT).

function DPLL (F: propositional CNF formula):

1. if F is empty; return true
DPLL is a decision procedure for satisfiability of Boolean formulas in conjunctive normal form (CNF-SAT).

function DPLL (F: propositional CNF formula):

1. if F is empty; return true
2. if F contains an empty clause; return false
DPLL is a decision procedure for satisfiability of Boolean formulas in conjunctive normal form (CNF-SAT).

function DPLL (F: propositional CNF formula):

1. if F is empty; return true
2. if F contains an empty clause; return false
3. if there exists a pure literal l in F;
   · return DPLL(F \land l)
DPLL is a decision procedure for satisfiability of Boolean formulas in conjunctive normal form (CNF-SAT).

function DPLL (F: propositional CNF formula):

1. if F is empty; return true
2. if F contains an empty clause; return false
3. if there exists a pure literal l in F;
   · return DPLL(F ∧ l)
4. if F contains a unit clause {l};
   · F₁ = {C − {¬l} | C ∈ F, l \in C}
   · return DPLL(F₁)
DPLL is a decision procedure for satisfiability of Boolean formulas in conjunctive normal form (CNF-SAT).

function DPLL (F: propositional CNF formula):

1. if F is empty; return true
2. if F contains an empty clause; return false
3. if there exists a pure literal l in F;
   · return DPLL(F ∧ l)
4. if F contains a unit clause {l};
   · F₁ = {C − {¬l} | C ∈ F, l ∉ C}
   · return DPLL(F₁)
5. Choose a variable x of F:
   · return DPLL(F ∧ l) ∨ DPLL(F ∧ ¬l)
function CDP (F: propositional CNF formula, n: integer):

1. if F is empty; **return** $2^n$
function CDP (F: propositional CNF formula, n: integer):

1. if F is empty; return $2^n$
2. if F contains an empty clause; return 0
function CDP (F: propositional CNF formula, n: integer):

1. if F is empty; return $2^n$
2. if F contains an empty clause; return 0
3. if F contains a unit clause $\{l\}$;
   - $F_1 = F_1 = \{ C - \{\neg l\} \mid C \in F, \neg l \notin C\}$
   - return CDP($F_1$, n - 1)
function CDP (F: propositional CNF formula, n: integer):

1. if F is empty; return $2^n$
2. if F contains an empty clause; return 0
3. if F contains a unit clause \{l\};
   - $F_1 = F_1 = \{C - \{\neg l\} \mid C \in F, l \notin C\}$
   - return $\text{CDP}(F_1, n - 1)$
4. Choose a variable x of F:
   - $F_1 = \{C - \{\neg x\} \mid C \in F, x \notin C\}$
   - $F_2 = \{C - \{x\} \mid C \in F, \neg x \notin C\}$
   - return $\text{CDP}(F_1, n - 1) + \text{CDP}(F_2, n - 1)$
AVERAGE RUNNING TIME ESTIMATION
· **Lemma 1:** If $F$ contains $m$ clauses and $n$ variables, then there is a constant $c$ such that $cmn$ bounds the time of executing one iteration of function CDP.
· **Lemma 1**: If $F$ contains $m$ clauses and $n$ variables, then there is a constant $c$ such that $cmn$ bounds the time of executing one iteration of function CDP.

· **Lemma 2**: Each of the formulas $F_1, F_2$ produced by splitting $F$ contains $m - k$ clauses with probability $\binom{m}{k} p^k (1 - p)^{m-k}$.

· $p$ is probability of a literal $l$ occurring in any clause.
For execution time bound of CDP, we can define some upper bounds based on the splitting step in CDP.

\[
T(m, n) \leq \begin{cases} 
    cmn + \\
    \sum_{k=1}^{m} \binom{m}{k} p^k (1 - p)^{m-k} T(m - k, n - 1) + \\
    \sum_{k=0}^{m} \binom{m}{k} p^k (1 - p)^{m-k} T(m - k, n - 1) \\
    1 
\end{cases} \\
n, m \geq 1 \\
n = 0 \text{ or } m = 0
CASE FOR $p = 1/3$

$$T(m, n) \leq \begin{cases} \text{cmn +} \\ \left(\frac{2}{3}\right)^m T(m, n - 1) + \\ 2 \sum_{k=1}^{m} \binom{m}{k} \left(\frac{1}{3}\right)^k \left(\frac{2}{3}\right)^{m-k} T(m - k, n - 1) \quad n, m \geq 1 \\ 1 \quad n = 0 \text{ or } m = 0 \end{cases}$$

- **Theorem 1**: For $p = 1/3$, $T(m, n) = O(m^2 n)$.
- **Theorem 2**: For any $p$, $T(m, n) = O(m^d n)$ where $d = \left\lceil \frac{-1}{\log_2(1-p)} \right\rceil$.

Proof time!
IMPROVEMENTS
CDP IMPROVEMENTS

- **Variable Selection in Splitting**: Try to minimize number of clauses $m_1, m_2$ in $F_1 \& F_2$. 
- **Variable Selection in Splitting**: Try to minimize number of clauses $m_1, m_2$ in $F_1 \& F_2$.
- One approach is minimizing $m_1 + m_2$ by choosing a variable appearing in maximal number of clauses of $F$. 
· **Variable Selection in Splitting**: Try to minimize number of clauses $m_1, m_2$ in $F_1 \& F_2$.

· One approach is minimizing $m_1 + m_2$ by choosing a variable appearing in maximal number of clauses of $F$.

· Another is minimizing $\max(m_1, m_2)$ by maximizing over a variable $x$ the quantity $\min(pos(x), neg(x))$ where $pos(x)$ ($neg(x)$) denotes the number of clauses $x$ ($\neg x$) appears.
· **Small Formula Problem**: What is \( F \) consists of a single clause with \( k \) literals?
CONCLUSION
We have presented a model counting algorithm (CDP) based on a SAT solver, DPLL.

We presented the average execution time with an upper bound.

We have presented some possible improvements on CDP.
Questions?