Provadores de Teoremas baseados em contagem

Eduardo Menezes de Morais

Orientador: Prof. Marcelo Finger

Instituto de Matemática e Estatística
Universidade de São Paulo
lenin@linux.ime.usp.br

1. O Problema

Neste trabalho estudamos provadores automáticos de Teoremas na lógica Proposicional. Na lógica proposicional as variáveis podem ter apenas dois valores: verdadeiro ou falso.
Toda fórmula na lógica proposicional pode ser representada como uma disjunção de conjunções (forma conhecida como Forma Norma Disjuntiva) com apenas três literais por conjunção.

Definição 1. *Uma interpretação é uma atribuição de valores Verdadeiro e Falso às variáveis da fórmula*

Definição 2. *Dizemos que uma fórmula na lógica proposicional é um teorema, ou tautologia, se todas as interpretações possíveis tornam a fórmula verdadeira.*

Os métodos tradicionais de se testar se uma fórmula é uma tautologia são variantes do DPLL, baseado em backtrack, e algorítmos genéticos. Nesse trabalho o autor procurou explorar outros métodos de se provar teoremas. Mas especificamente, provar teoremas contando (não necessariamente enumerando) o número de interpretações que satisfazem uma fórmula.

2. A complexidade de contar

Quando se tem o número exato de interpretações que satisfazem uma fórmula, fica fácil saber se essa fórmula é uma tautologia, se é satisfatível, etc. Além do mais essa informação é útil como heurística.
Porém, o problema de contar o número de interpretações em uma fórmula possivelmente é mais difícil que um problema NP-Completo. Não faz muito sentido comparar esse problema diretamente com NP pois ele não é um problema de *decisão*, mas sim um problema de *contagem*. L. G. Valiant classificou esse problema na classe #P, sendo que ele e #P-Completo [5]. Se #P puder ser resolvido em tempo polinomial, então temos que $P = NP$, porém mesmo que $P = NP$, não podemos garantir que podemos resolver #SAT em tempo polinomial.

3. Os algoritmos

3.1 Tabela-Verdade

O jeito mais trivial de se contar o número de interpretações é olhar uma por uma. Esse e o famoso método da Tabela-Verdade

3.2 Algoritmo de Iwama

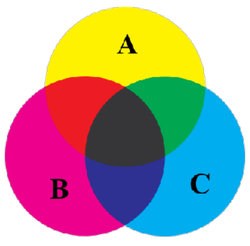
Kazuo Iwama criou um algoritmo[4] para contar sem enumerar que serviu de base e influenciou diversos outros algoritmos[1].
A idéia é que se temos n variáveis e apenas i literais em uma conjunção, essa conjunção valida 2^{n-i} interpretações.
Ex.: $a \wedge b$ é verdadeiro com as interpretações abc e $ab\bar{c}$
Infelizmente não podemos simplesmente somar o número de interpretações de cada conjunção pois algumas interpretações tornam verdade mais de uma conjunção. Por isso temos que somar as interpretações de cada conjunção, subtrair as que aparecem em duas conjunções, somas as que aparecem em três, ..., do mesmo modo que na Teoria dos Conjuntos.

$$S = \sum_{\Omega \subset C} (-1)^{\|\Omega\|} 2^{(n-\phi(\Omega))}$$

3.3 Algoritmo de Dubois

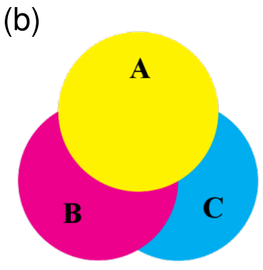
Olivier Dubois em [2] partiu da mesma idéia que Iwama, mas escolheu um caminho diferente: ao invés somar e subtrair interpretações de modo a evitar que se conte duas vezes a mesma interpretações, Dubois modifica a fórmula original de maneira a tentar fazer com que cada conjunção seja "independente" das demais.

(a)



(Interpretações originais)

(b)



(Após modificações)

Exemplo 1. $(a \wedge b \wedge c) \vee (d \wedge e)$ *se torna* $(a \wedge b \wedge c) \vee (d \wedge e \wedge \neg a) \vee (d \wedge e \wedge a \wedge \neg b) \vee (d \wedge e \wedge a \wedge b \wedge \neg c)$

3.4 Um novo algoritmo

O autor também criou o seu próprio algoritimo. A idéia é baseada na transformação do algoritimo de Dubois, mas ao invés de fazer essa transformação explicitamente, apenas calcula-se quantas conjunções e com que tamanho apareceriam, usando análise combinatória.
É fácil calcular esse número quando se assume que nenhum literal se repete. Então inicialmente assumimos que este é caso.
Cada literal repetido chamamos de "conflito". Para cada conflito calculamos em quantas conjunções isso ocorre e fazemos as modificações necessárias para corrigir o número de literais.
Para mais detalhes sobre o algoritimo consulte a monografia desse trabalho.

4. A transição de fase

Diversos autores [3] apontam o fato de que em muitos provadores de teoremas, parece ocorrer uma dificuldade a mais em verificar se a fórmula é um teorema depende da proporção Conjunções/variáveis. Se o número de conjunções é muito pequeno comparado ao número de variáveis, as fórmulas tendem a não ser teoremas e serem resolvidas rápido. Se essa número é muito grande, as fórmulas tendem a ser teoremas e também serem resolvidas rápido. Porém se esse número está entre 8 e 16, as fórmulas podem ser ou não teoremas e tendem a demorar muito mais para serem resolvidas.

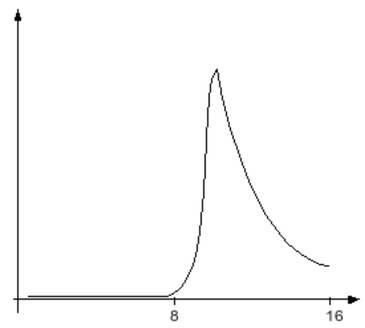


Figura 2: *Baseado nos dados do artigo de Ian P. Gent e Toby Walsh*

Em nenhum dos algoritmos baseados em contagem esse comportamente foi visto, mostrando que eles são alternativas a serem pesquisadas.

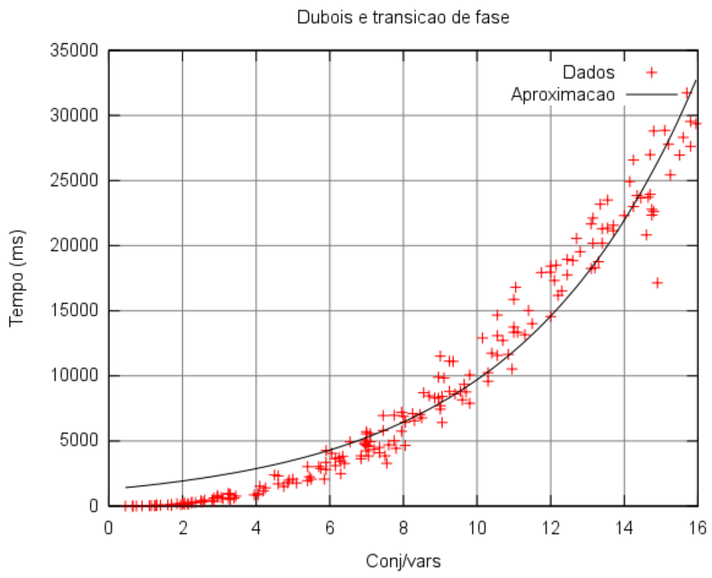


Figura 3: *Gráfico do algoritimo de Dubois quanto a transição de fase. Outros algoritimos tem comportamento semelhante*

Referências

[1] Ştefan Andrei. Counting for satisfiability by inverting resolution. *Artificial Intelligence Review*, 22(4):339–366, 2004.

[2] Olivier Dubois. Counting the number of solutions for instances of satisfiability. *Theor. Comput. Sci.*, 81(1):49–64, 1991.

[3] Ian P. Gent and Toby Walsh. The sat phase transition. In *Proceedings of 11th ECAI*, pages 105–109, 1994.

[4] Kazuo Iwama. Cnf satisfiability test by counting and polynomial average time. *SIAM J. Comput.*, 18(2):385–391, 1989.

[5] L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.