Das Técnicas e da Importância do SAT

Cristiano Medeiros Dalbem
Universidade Federal do Rio Grande do Sul – UFRGS
Ciência da Computação
INF05508 - Lógica para computação

Dezembro de 2008
Porto Alegre, Rio Grande do Sul, Brasil

Introdução

Não é pretendido aqui destrinchar a questão da satisfatibilidade, e nem também fugir do problema e estudar apenas a ligação entre as diferentes versões deste mesmo problema que se encontram por toda a Ciência da Computação e é um dos problemas matemáticos e computacionais mais importantes hoje em aberto. Não. O que farei aqui é introduzir conceitos de NP-completude, demonstrar por exemplos a importância de se questionar "P=NP?" e como é interessante e ao mesmo tempo amedrontador pensar que há quase 50 anos não se tem uma solução para isto.

"SAT is the Drosophila
of
Complexity Theory" [1]

"*Drosophila melanogaste*r (conhecida por mosca da fruta) constitui um organismo modelo em estudos de genética tendo permitido obter dados importantes sobre estrutura e funcionamento de genes, evolução e desenvolvimento embrionário."[2] "*Melanogaster*" significa, grosso modo, "ventre colorido", característica deste tipo de mosca que tanto ajudou neste tipo de estudo, importância comparada a das ervilhas das Leis de Mendel.

Por esta frase temos a essência do problema da satisfatibilidade em termos do relacionamento com todos os outros problemas: sabemos que todo problema NP-completo, que é da série do problemas mais difíceis, e dos mais estudados, podem ser reduzidos a instâncias do problema da satisfatibilidade booleana em lógica. Esta descoberta não simplesmente revolucionou, mas deu vida à discussão dos problemas NP-completos, assim como de outras classes, depois estudadas, de problemas ainda mais complexos.

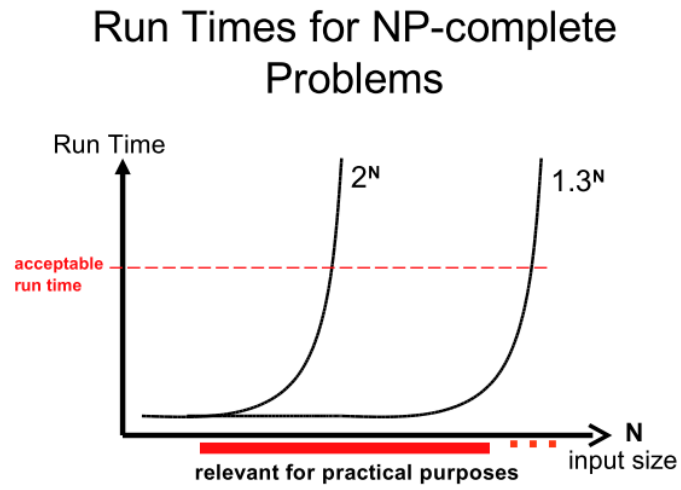


Gráfico que demonstra o comportamento de algoritmos de tempo exponencial em função do tamanho da entrada - mesmo uma otimização deles ainda apresenta um comportamento "desconfortável".

[1] Uwe Schöning – "Quest for the Fastest SAT Algorithm" - University of Ulm, Germany
[2] http://www.fc.up.pt/foco/drosophila_2008.htm

SAT

Definição?

O problema de satisfatibilidade booleana, conhecido mundialmente como simplesmente SAT (sigla?), é notável pela sua tanto enorme simplicidade quanto enorme dificuldade de resolução eficiente. Aliás, o SAT é "O" problema - não só na Ciência da Computação, especificamente na área de Complexidade de Algoritmos (de forma mais branda, na área de Algoritmos), como também em uma vasta zona de influência da matemática; afinal de contas, o que é que do que tange à teoria da Ciência da Computação não se trata também de matemática?. O problema do SAT foi não só o primeiro a ser descoberto, mas também o principal de uma classe de problemas chamada NP-completos, subconjunto dos problemas NP - uma sigla mal ajeitada de *Non-Deterministic Polynomial Time*, referindo-se ao tempo necessário para resolução utilizando-se dos algoritmos mais eficientes já encontrados. Mais sobre problemas NP-completos mais adiante.

Falando-se especificamente da satisfatibilidade booleana, ou simplesmente satisfatibilidade (por definição, binária no prisma da Lógica "clássica"), é uma propriedade presente em uma forma lógica, propriedade esta dada no momento em que é provado que uma certa combinação de atribuições a proposições lógicas de uma fórmula a faz ser verdadeira. Por conseguinte, temos que uma fórmula não é satisfatível no momento em que provamos que não há combinação de atribuições passível de torná-la verdadeira.

Fórmulas Normais Conjuntivas e k-SAT

O conceito mais comum envolvendo problemas de satisfatibilidade está relacionado com o conceito de k-SAT. Denomina-se k-SAT o problema de resolução da satisfatibilidade de uma fórmula lógica proposicional que se encontra normalizada em FNC. Uma fórmula em FNC é uma fórmula cujas cláusulas estão relacionadas por operadores lógicos "E", e cada cláusula é constituída de proposições, negadas ou não, operadas entre si por "OU". O "k", de k-SAT, está relacionado com o tamanho das cláusulas de uma certa fórmula normalizada.

$$(A \vee B) \wedge (\neg B \vee C \vee \neg D) \wedge (D \vee \neg E)$$

Exemplo de fórmula normalizada em FNC

É interessante notar que uma fórmula normalizada em FND torna o trabalho de determinação da satisfatibilidade muito mais simples: em FND, todas cláusulas de proposições entre "E"s estão operadas por "OU"s; assim, a fórmula é satisfatível se e somente se uma de suas cláusulas o são, e uma cláusula é satisfatível se e somente se nela não há tanto um proposição *p* quanto sua forma negada ¬*p*. Certamente este trabalho é um trabalho de tempo polinomial, então por que não ter uma fórmula lógica sempre normalizada desta maneira? Simples: porque o trabalho necessário pra realizar esta transformaçõe pode e é tão complicado quanto um problema NP-completo.

A razão, então, de se trabalhar com k-SAT tendo fórmulas em FNC é pelo simples fato de se ter uma estrutura pela qual percorrer. Ainda que todo algoritmo já encontrado para esta resolução faça de alguma maneira sim um caminhamento randômico e exponencial, uma estrutura, assim como técnicas que veremos a seguir pode diminuir de maneira considerável a complexidade do problema.

Uma luz em meio à escuridão: o 2-SAT, ou 2-satisfatibilidade

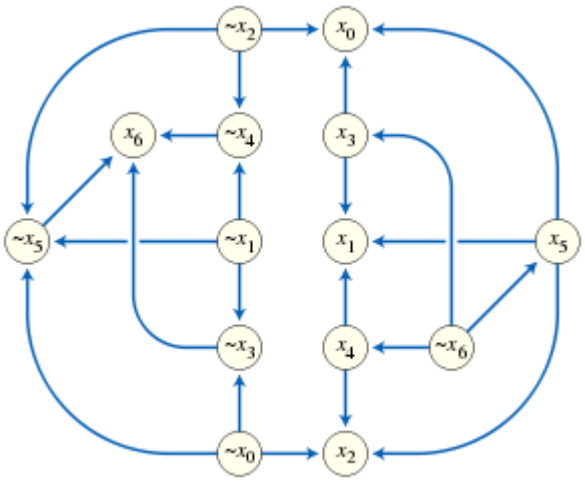
Provavelmente a interpretação mais correta do uso do título desta seção seja a do sensacionalismo. Afinal de contas, o problema do 2-SAT não chega a ser uma luz, mas talvez apenas um certo nicho iluminado da vastidão escura que é o problema da satisfatibilidade, ou de maneira geral, dos problema NP-completos. Sim, é isso o que o leitor está cogitando ser: o 2-SAT é um problema NL-completo. Falaremos mais sobre o que é um problema NL-completo mais adiante; por enquanto, serve aos nossos propósitos dizer que é um problema que pode ser resolvido em tempo polinomial, ou seja, há algoritmo eficiente o suficiente para que o problema seja considerado resolvido para soluções rápidas.

Em 2-SAT, é comum se fazer uma transformação para uma fórmula normal implicativa, fórmula esta que nos guiará à resolução daquela. A figura abaixo demonstra como funciona a transformação.

$$(x_0 \vee \neg x_3) \equiv (\neg x_0 \Rightarrow \neg x_3) \equiv (x_3 \Rightarrow x_0).$$

A partir de tal normalização, é formado um "grafo de implicações". As figuras abaixo mostram uma fórmula em FNC e o correspondente grafo de implicações, respectivamente.

$$(x_0 \vee x_2) \wedge (x_0 \vee \neg x_3) \wedge (x_1 \vee \neg x_3) \wedge (x_1 \vee \neg x_4) \wedge (x_2 \vee \neg x_4) \wedge$$
$$(x_0 \vee \neg x_5) \wedge (x_1 \vee \neg x_5) \wedge (x_2 \vee \neg x_5) \wedge (x_3 \vee x_6) \wedge (x_4 \vee x_6) \wedge (x_5 \vee x_6).$$



O principal método de resolução foi demonstrado em 1979 por Aspvall, Plass & Tarjan e consiste na afirmação de que a fórmula é satisfatível se e somente NÃO há componentes fortemente conexos no grafos de implicações. Como são conhecidos

algoritmos de tempo linear para esta verificação, foi concluído que o 2-SAT é um problema resolvível, assim, também em tempo linear.

Ser um problema NL-completo significa dizer que é um dos problemas principais de uma classe NL, sigla para *Nondeterministic Logarithmic-space*, que significa que o problema pode ser resolvido por uma máquina de Turin não-determinística utilizando-se de uma quantidade logarítmica de memória. Um outro problema de expressiva importância desta classe é o de ST-conectividade (ST-conctivity, tradução livre), que consiste em, dados dois nodos S e T e um grafo direcionado (dígrafo), perguntar se T pode ser alcançado por S.

Uma aplicação interessante destes conceitos é da montagem das legendas dos pontos em um mapa, por exemplo. Imagine que cada ponto em um mapa em duas dimensões seja uma cidade, e que gostaríamos de ter um sistema automatizado para a geração deste mapa. O algoritmo para colocação das legendas será uma das partes mais complicadas da implementação desse sistema, pois pense em tudo que ele deve analisar: justaposição de legendas, distância da legenda do ponto, ambigüidade, tamanhos, cores, etc.

Exemplo de caso de posicionamento de nomes de cidades em um mapa

Ocorre que tal problema é na realidade um problema NP-completo, mas pode ser resolvido em tempo linear aplicando-se uma restrição: o número de possibilidades de montagem de uma legenda é binário - digamos, por exemplo, que o algoritmo deve decidir se a legenda ficará do lado esquerdo ou direito do ponto. Neste caso, o problema pode ser convertido em uma fórmula lógica proposicional em FNC na forma de um 2-SAT, e assim resolvido como qualquer outro deste tipo.

De volta ao lado negro: 3-SAT

O 3-SAT não é nada mais do que uma generalização do problema SAT, mas é um guia ao seu entendimento. Um guia não muito simpático é bem verdade, pois dele sabemos, comprovadamente, que:

- Um problema 3-SAT é um problema NP-completo;
- Um problema k-SAT pode ser reduzido, sem modificar sua propriedade satisfatível, a um 3-SAT.

Estas foram notícias tristes, mas somente para aqueles que tinham necessidade de uma solução rápida e eficaz ao SAT. Na realidade, a descoberta do 3-SAT ser NP-completo é equivalente à descoberta do SAT o ser, o que foi uma descoberta de vasta influência feita por Cook-Levin, conhecido pelo resultante Teorema de Cook-Levin. Mais. Em seu teorema ele afirma que, se há um algoritmo determinístico de tempo polimial que solucione o 3-SAT, assim também haverá para qualquer outro problema NP-completo. O “completo” desta denominação afirma justamente esta equivalência deste conjunto de problemas.

A NP-completude

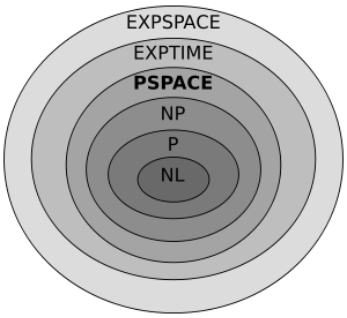
A descoberto de Cook, feita em 1971, implicou em um famoso e não menos importante artigo por Richard Karp, em 1972, que levou a descoberta de Cook um passo à frente ao provar que 21 de alguns dos mais importantes problemas computacionais daquele tempo eram problemas NP-completos. Fascinante, e aterrorizante, é ver que todos estes problemas, e muitos outros que hoje já são provados serem NP-completos, permanecem sem resposta. Tanto Cook quando Karp ganharam o Turing Award pelos seus respectivos artigos *"The complexity of theorem proving procedures"*e *"Reducibility among combinatorial problems"*.

Dos *Karp's 21 NP-complete problems*, lista que renovou o interesse neste campo, e assim o manteve até hoje, alguns dos exemplos são de alta popularidade mesmo para amadores da ciência da computação, e são todos impressionantes pela sua simplicidade e aparente facilidade. Dentre eles, estão:

- Número cromático;
- Cobertura de vértices;
- Cliques;
- Conjuntos independetes;
- Grafos Hamiltonianos;
- Etc.

Há jogos clássicos que também tem em sua essência problemas NP-completos, como o FreeCell – uma generealização com 4*n cartas mostra que encontrar configurações de vitória é um problema NP-Completo -, ou o Sudoku – linhas, colunas e seções da tabela são conjuntos de nodos interligados, e encontrar a combinação certa que torna o faz "vencer" o jogo é equivalente a um problema de Coloração de Grafos.

E os predicados, onde ficam nisso tudo? QSAT!



Uma representação da relação entre as classes de complexidade

Até agora estivemos falando só dos casos mais comuns de SAT, que são os que envolvem fórmula em FNC constituídas apenas de proposições não quantificadas. Estas instâncias podem ser chamadas de PSAT.

São, na realidade, bem mais recentes as pesquisas envolvendo as QSAT (Quantified SAT).

"SAT por si mesmo usa apenas quantificadores $\exists$. Se permitirmos apenas quantificadores $\forall$, torna-se um *Co-NP-complete tautology problem*. Permitindo ambos,

o problema denomina-se *quantified Boolean formula problem* (QBF), o qual pode ser demonstrado ser PSPACE-completo. Acredita-se fortemente que problemas PSPACE-completos são mais difíceis que qualquer problema em NP, ainda que isto não tenha sido provado."[3]

Problemas PSPACE-completos são definidos como problemas solúveis por máquinas de Turing não-determinísticas com memória de tamanho polinomial em tempo polinomial, o que aparentemente mostra-se ser realmente mais complexo que o NP.

As Soluções

Não é necessário dizer que, sendo um problema da matemática (é bom pontuar a quase multidisciplinariedade da questão) em aberto desde tempos anteriores às descobertas de Cook, pelos dios dos anos 60, muitas coisas interessantes já foram observadas e provadas a este respeito. Além do mais, com o exponencial (com o perdão do trocadilho) crescimento da computação nos últimos tempos, o problema da NP-completude se tornou um gargalo de uma enormidade de problemas que não poderiam ficar sem tratamento.

São duas as classes de algoritmos utilizados e desenvolvidos hoje em dia: modernas variantes do bastante antigo DPLL e os *local search*, como o GSAT e o WalkSAT.

Davis-Putnam-Logemann-Loveland algorithm

O DPLL é um algoritmo desenvolvido em 1962 por uma equipe encima de outro ainda mais antigo, desenvolvido por Davis e Putnam em 1960. É um algoritmo bastante poderoso e eficiente, que mesmo hoje ainda é base para o desenvolvimento de outros ainda mais potentes, tanto para os problemas SAT quanto para problemas com teoremas de lógica de primeira-ordem.

O método utilizado pra a solução é o de *backtracking*, que não é nada mais do que a possibilidade de detecção de um resultado parcial comprovadamente incorreto, que é logo descartado, o que pode mudar drasticamente o desenvolvimento do processamento de um problema. Um exemplo clássico é o do quebra-cabeça das oito rainhas (*eight queens puzzle*, tradução livre), onde queremos as distruibuir em um tabuleiro de chadrez 8x8 de maneira que nenhuma possa atacar a outra. "Na abordagem em backtracking, os candidatos parciais são distruições de k rainhas nas primeiras k colunas do tabuleis, cada uma em uma diferente coluna e em uma diferente linha. Qualquer solução parcial que contenha duas rainhas a se atacar mutuamente pode ser abadandonado, já que não poderá resultas em uma solução válida".[4]

[3] http://en.wikipedia.org/wiki/Boolean_satisfiability_problem
[4] http://en.wikipedia.org/wiki/Backtracking

GSAT e WalkSAT

Ambos são algoritmos que tratam o problema primeiramente com atribuições randômica, testam se a fórmula é verdade e então aplicam uma mudança alguma variável.

"O GSAT muda a variável que minimiza o número de cláusulas não satisfeitas, provavelmente utilizando probabilidades ao invés de pegar uma variável aleatória.

O WalkSAT primeiramente muda uma variável de uma cláusula não satisfeita com as atribuições atuais. A variável escolhida geralmente resulta no menor número possível de cláusulas anteiormente satisfeitas tornando-se não-satisfeitas, com alguma chance de usar uma variável aleatória. Quando se usa uma variável que pode ser a opção ótima, o WalkSAT faz menos cálculos que o GSAT foi considera menos possibilidades."[5]

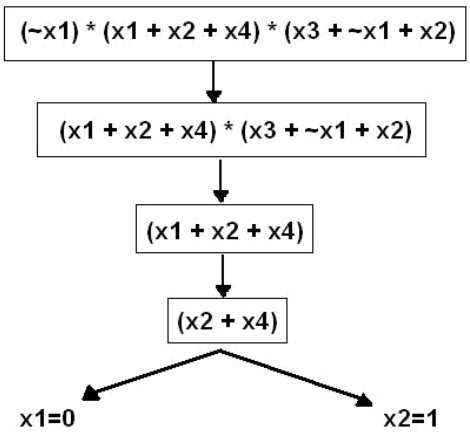
Algoritmo de Davis-Putnam

O algoritmo de Davis-Putnam foi o pioneiro na empreitada pelos métodos mais eficazes de resolução de SAT. É um método, assim como seu predecessor, o DPLL, baseado em *backtracking*.

É um algoritmo recursivo que cria uma árvore de busca como método para Divisão e Conquista. Esta estrutura também torna o processo de backtracking mais acessível.

Seu funcionamento é simples: se a associação a uma variável torna uma cláusula falsa, então o resultado parcial é descartado, assim como todo provindos deste. "O algoritmo procura identificar a não satisfatibilidade em um assinalamento parcial de forma que não haja necessidade de tentar outras variáveis ainda sem valor atribuído. Isso normalmente resulta em uma procura mais eficiente do que simplesmente enumerar cada uma das combinações possíveis de valoração."[6]

A idéia, como pode-se notar, é bastante simples, mas até hoje faz parte de qualquer método eficiente de resolução.



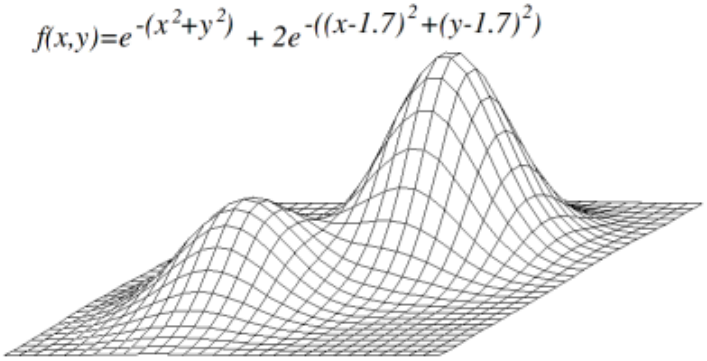
Exemplo de execução do algoritmo. Aqui, o problema foi resolvido utlizando-se principalmente da Propagação Unitária, que é a decisão de associação de valor a uma cláusula unitária (¬x1, no exemplo).

[5] http://en.wikipedia.org/wiki/WalkSAT
[6] http://pt.wikipedia.org/wiki/Algoritmo_de_Davis-Putnam

O *hill climbing*

O *hill climbing* não é um algoritmo em si, mas uma técnica de resolução com aproximação utlizado geralmente em problemas de crescimento exponencial. Ele pertence a uma classe de técnicas chamada *local search*, e é a que melhor a exemplifica.

O *hill climbing* inicia com uma série de associações aleatórias para depois começar a analisar instâncias locais das variáveis onde pretende "mexer" a fim de encontrar uma solução ótima. Não é garantido que chegaremos na solução ótima do problema tratando-o desta maneira, e muitas vezes encontramos soluções ótimas locais que são condições de parada do algoritmo, mas podem estar ainda longe de uma solução ótima global. O *hill climbing*, como qualquer outro algoritmo de aproximação, utiliza-se de meta-heurística, que pode ser definida grosso modo como uma forma de heurística onde as especulações são mais difíceis.



Exemplo onde um algoritmo de hill climbing *pode encontrar condições de parada em solução ótimas locais mais não globais, como num caso de maximização de uma função. Por esta visualização também podemos entender o porquê do nome* hill climbing.

Mais ainda, um algoritmo de *hill climbing* por si só pode encontrar condições de parada determinadas por uma faixa de tempo ou de passos onde as soluções encontradas não diferem muito entre si, como no caso de um "planalto" em um gráfico de soluções (também como uma "planície" no gráfico acima).

Micelânia: Horn-SAT

Uma formula Horn é uma série de conjunções de cláusulas Horn, estas definidas como cláusulas onde no máximo uma variável é "positiva" enquanto as outras são todas negadas.

O problema da satisfatibilidade em uma cláusula Horn é tido como um dos mais difíceis de ordem P (ele é P-completo), ou seja, pode ser resolvido em tempo polinomial.

O algoritmo utilizado para este problema baseia-se basicamente no conceito já aqui exposto da Propagação Unitária., e a satisfatibilidade é comprovada se uma série de transformações deste tipo não gera um par p e $\neg p$.

Já especulou-se sobre a possibilidade de uso do Horn-SAT para comprovação de P=NP, transformando-se um problema NP em uma fórmula Horn. Hoje sabe-se que isto não é possível, pois o procedimento de transformação ocorre em tempo exponencial, além de que a fórmula resultante seria de tamanho exponencial.

Conclusão

A conclusão na qual chegamos depois de uma breve passagem pelo problema da satisfatibilidade booleana e da NP-completude é a de que não há grandes conclusões sobre este assunto.

Há quase 50 anos Estados Unidos e União Soviética, as duas maiores potências militares (e intelectuais, por que não?) que já foram vistam, trabalhavam e faziam as primeiras descobertas neste campo da Teoria da Complexidade – também não é errado dizer que há quase 50 anos NASCIA a Teoria da Complexidade, pois ela está fortemente vinculada (e completamente dependente) das descobertas no campo da computação, área esta extremamente nova e que deu um sentido completamente novo à lógica, apresentando formas de aplica-la jamais pensadas antes.

Até hoje muito do que se descobriu diz respeito apenas à dificuldade de responder à pergunta "P = NP?", e os algoritmos mais eficazes utilizados hoje em grande escala para resolução de problemas SAT e afins são muito fortemente baseados nos primeiros algoritmos de Davis e Putnam dos anos 60.

Finalizando, segue uma tabela de alguns dos algoritmos mais utlizados hoje em dia, principalmente *estocásticos*, que demonstra muito bem o quanto este é ainda um problema mal resolvido (veja gráfico da Introdução).

Best Algorithms for 3-SAT[7]
(worst case bounds)

backtracking, truth-table……2n
Monien, Speckenmeyer 1985……..1.619n
*** Paturi, Pudlak, Zane 1997……...1.588n**
Kullmann 1992……1.505n
Goerdt, Schöning 1999……1.481n
*** Paturi, Pudlak, Saks, Zane 1998……1.447n**
*** Schöning 1999……(4/3)n**
*** Hofmeister, S., Schuler, Watanabe 00……1.330n**
*** Baumer, Schuler 2002……...1.329n**
*** Rolf 2003……...1.328n**
*** Iwama, Tamaki 2003……1.324n**

* Stochastic algorithms

[7] Uwe Schöning – "Quest for the Fastest SAT Algorithm" - University of Ulm, Germany

Bibliografia

Toda pesquisa pra este trabalho foi realizada pela Internet, usando principalmente informações de acesso público das redes da Wikipedia. É notável a quantidade e qualidade de informações encontradas em português nesta rede.

Segue lista de sites consultados.

1. en.wikipedia.org/wiki/Boolean_satisfiability_problem
2. www.ti.inf.ethz.ch/ew/mise/Abstracts/2005/SchoeningQuestFastSat21-5-05.pdf
3. anytime.cs.umass.edu/aimath06/proceedings/P34.pdf

As páginas abaixo são facilmente acessíveis pelo "portal" da SAT na wikipedia, cujo link é o de número 1. Sugiro-as no caso do leitor querer mais informações sobre o assunto. Os títulos dos links são auto-explicativos.

4. en.wikipedia.org/wiki/NL-complete
5. en.wikipedia.org/wiki/2-satisfiability
6. en.wikipedia.org/wiki/Cook%27s_theorem
7. en.wikipedia.org/wiki/Conjunctive_normal_form
8. en.wikipedia.org/wiki/P_%3D_NP_problem
9. en.wikipedia.org/wiki/PSPACE-complete
10. en.wikipedia.org/wiki/DPLL_algorithm
11. en.wikipedia.org/wiki/WalkSAT
12. en.wikipedia.org/wiki/Quantified_Boolean_formula_problem
13. en.wikipedia.org/wiki/MAX-SAT
14. en.wikipedia.org/wiki/Local_search_(optimization)
15. pt.wikipedia.org/wiki/Algoritmo_de_Davis-Putnam
16. en.wikipedia.org/wiki/Horn-satisfiability
17. en.wikipedia.org/wiki/Hill_climbing