ALGORITMOS PARA O PROBLEMA DE SATISFAZIBILIDADE

Isis Burmeister Pericolo
Leonardo Holtz de Oliveira
Milena Lopes Maciel

Simple solutions seldom are. It takes a very unusual mind to undertake analysis of the obvious.
Alfred North Whitehead

RESUMO: Este trabalho tem por objetivo estudar o problema da satisfazibilidade booleana, e se aprofundar em algoritmos que o resolvam, na complexidade computacional de cada um deles e descrição de métodos de construção de código, bem como manipulações lógicas necessárias para a satisfação de fórmulas quaisquer. Sendo um problema muito relevante tanto para a teoria quanto para questões práticas da ciência da computação, entender as restrições necessárias para achar resultados satisfatórios se torna interessante atualmente.
PALAVRAS-CHAVE: Satisfazibilidade, Algoritmos, Ciência da Computação, Lógica Formal, Complexidade.

1 INTRODUÇÃO

Neste trabalho serão apresentadas as características e algoritmos usados comumente para resolução do problema de satisfazibilidade booleana (abreviado como SAT), o qual se propõe a determinar se uma fórmula booleana pode ser satisfeita por um conjunto de valorações de seus literais. Simplificadamente, tratamos da questão de descobrir a existência de valores (verdadeiro ou falso) para as variáveis de uma fórmula proposicional qualquer tal que ela resulte verdadeira.

O problema de satisfazibilidade foi o primeiro a ser categorizado como NP-completo, em questões de otimização computacional; isso quer dizer que não há uma solução rápida e eficiente conhecida para ele, e suspeita-se que nem ao menos existe um algoritmo tal - apenas sabemos que tais resoluções computam em tempo exponencial no pior caso possível. No entanto, como tais problemas necessitam ser resolvidos, usamos ferramentas como métodos heurísticos, algoritmos de aproximação, restrições estruturais ou parametrização das entradas para solucionar de maneira mais rápida.

1.1 MOTIVAÇÃO

Algoritmos SAT se tornam extremamente importantes dentro da ciência da computação, sendo utilizados em diversas áreas, tanto teóricas e de estudo de complexidade algorítmica, como em design de algoritmos e circuitos, inteligência artificial, criptografia e até quando simplesmente precisamos de um método automatizado para provar teoremas. Por isso, eles são estudados a fundo e no presente temos algoritmos que conseguem avaliar fórmulas que envolvem dezenas de milhares de literais e milhões de símbolos, o que é suficiente para a grande maioria dos problemas práticos. Além disso, em questões de complexidade, por ser parte da categoria NP-completo, a descoberta de um algoritmo rápido e eficiente para ele viria a resolver o problema de P vs. NP, um dilema famoso e até hoje sem resolução, mas extremamente importante na ciência da computação.

2 RESTRIÇÕES NECESSÁRIAS PARA A RESOLUÇÃO DE SATs

Neste trabalho trabalharemos em grande parte com algoritmos que exigem certas restrições à fórmula de entrada, para assim determinar a satisfazibilidade dela. Uma fórmula qualquer e aleatória possivelmente não conseguiria ser resolvida por um computador, dada que esta fosse muito grande, pois nenhuma máquina teria capacidade de computá-la, logo, essas restrições se tornam necessárias para que possamos manipular a entrada e achar uma resposta. Ainda que hajam outros métodos de construir algoritmos para o problema da satisfazibilidade booleana que não envolvam restringir as instâncias a determinados formatos, julgamos que este modo, por lidar com a manipulação dos conectivos e organização de fórmulas formais, seja o mais interessante para os objetivos deste trabalho e os conteúdos que aqui expomos.

2.1 TEMPO POLINOMIAL

Existem alguns modos de fazer algoritmos de resolução de SAT funcionarem em tempo polinomial, ou seja, dadas algumas restrições específicas à fórmula de entrada, a verificação de sua satisfazibilidade por um computador pode ser realmente rápida.

2.1.1 Forma Normal Disjuntiva

Se a fórmula dada estiver no formato de forma normal disjuntiva (FND) e dentro de cada implicante não exista, simultaneamente, um literal e sua negação, poderemos apenas verificar se um dos implicantes é satisfeito pelo nosso conjunto de valorações. Se assim for, a fórmula é satisfazível.

Isso se torna interessante pois várias fórmulas aparentemente complicadas podem ser simplificadas no formato FND e, apresentando-se a segunda condição também, teremos uma resposta rápida para um problema originalmente extenso.

2.1.2 2-SAT

O algoritmo comumente conhecido como 2-SAT, usa como entradas fórmulas em formato FNC (forma normal conjuntiva) com até dois literais em cada cláusula e também, por vezes, grafos direcionados especiais, chamados grafos de implicação. Tais grafos representam a fórmula e são formados usando uma equivalência lógica tripla que mostraremos abaixo, sendo que cada vértice representa um literal ou sua negação e cada arco (aresta direcionada) representará a implicação entre os literais.

$$(x0 \vee \neg x1) \equiv (\neg x0 \Rightarrow \neg x1) \equiv (x1 \Rightarrow x0)$$

Existem algumas formas de escrever algoritmos 2-SAT, incluindo usar recursos de backtracking e análise de componentes fortemente conectados. Neste trabalho, porém nos aprofundaremos um pouco mais na alternativa que trabalha diretamente com a fórmula proposicional e a manipulação dos conectivos lógicos.

Um algoritmo tal foi descrito por Krom em 1967 e este funciona assim: partimos da suposição de que uma instância de entrada tem duas cláusulas que possuem uma mesma variável, exceto que em uma delas esta está negada e na outra não; usaremos estas duas cláusulas para produzir uma terceira do seguinte modo: tendo $(a \vee b)$ e $(\neg b \vee \neg c)$ produzimos $(a \vee \neg c)$, o que equivale a fazermos a operação de transitividade na forma implicativa da fórmula -- as cláusulas originais são apresentadas como $(\neg a \Rightarrow b)$ e $(b \Rightarrow \neg c)$, e então conseguimos $(\neg a \Rightarrow \neg c)$.

A partir daí, podemos tirar conclusões sobre a fórmula ser *consistente* ou não: Krom descreve que se essas repetidas aplicações de inferências não gerarem $\forall x$ presente na fórmula, simultaneamente, as cláusulas $(x \vee x)$ e $(\neg x \vee \neg x)$, a fórmula é consistente. O motivo para isso é que não é possível satisfazer as duas cláusulas descritas acima ao mesmo tempo. Podemos, então, estender a fórmula de entrada,

adicionando sempre um dos formatos de cláusula descritos acima para cada variável presente, mantendo sempre a consistência. Assim que este processo estiver completo, podemos ver a satisfazibilidade da fórmula ao marcar todas as variáveis que estão em formato $(x \vee x)$ como verdadeiras e as que estão em formato $(\neg x \vee \neg x)$ como falsas.

2.1.3 HORN-SAT

Há ainda outro algoritmo para o problema da satisfazibilidade booleana que funciona em tempo polinomial: a satisfazibilidade de Horn. Nele, as restrições feitas são simplesmente que a fórmula necessita ser uma fórmula de Horn, que por sua vez é uma conjunção de cláusulas de Horn.

Cláusulas de Horn são termos em que existe apenas um literal positivo ligado a um número indefinido de literais negativos por disjunções, ou seja, algo nesse formato: $(\neg x1 \vee \neg x2 \vee ... \vee \neg xn \vee y)$, onde y seria a "cabeça" da cláusula, o literal positivo dela (isso se formata assim pois uma cláusula tal é logicamente equivalente a $(x1 \wedge x2 \wedge ... \wedge xn) \Rightarrow y$). Pode-se dizer que as fórmulas de Horn são então modificações da forma normal conjuntiva.

O algoritmo de satisfazibilidade usando fórmulas de Horn utiliza da regra de propagação unitária: se na fórmula existe uma cláusula com um único literal x, então todas as cláusulas contendo x (exceto a unitária de origem) e todas as cláusulas contendo $\neg x$ são removidas; após esse passo, pode-se gerar uma cláusula unitária, que será tratada da mesma maneira que a de x, caso venha a ocorrer.

E então, se não existem cláusulas unitárias na fórmula, podemos satisfazê-la marcando todas as variáveis restantes como falsas. A instância será insatisfazível se as transformações feitas geram um par de cláusulas x e $\neg x$.

> This algorithm also allows determining a truth assignment of satisfiable Horn formulae: all variables contained in a unit clause are set to the value satisfying that unit clause; all other literals are set to false. The resulting assignment is the minimal model of the Horn formula, that is, the assignment having a minimal set of variables assigned to true, where comparison is made using set containment.
>
> (Cook, S.; Nguyen, P.. *Logical foundations of proof complexity.*)

Diz-se que a satisfazibilidade de Horn, categorizada como P-completa, é um dos problemas mais difíceis e mais expressivos dos problemas categorizados como resolvíveis em tempo polinomial.

2.2 3-SAT

O problema 3-SAT é outra restrição da satisfazibilidade booleana, mas ele é menos rígido que os outros exemplos mostrados: tendo uma fórmula em forma normal conjuntiva e com no máximo três variáveis, podemos provar a satisfação dela se ao menos um dos literais em cada cláusula for verdadeiro.

Um dos problemas normalmente resolvidos com o uso de 3-SAT é o de cliques em um dado grafo, com n nodos, onde cada um é um dos literais da fórmula. O grafo só terá n-clique se a fórmula resultante desse processo for satisfazível.

O 3-SAT e suas derivações, diferentes dos outros algoritmos que falamos sobre neste trabalho, são NP-completos, difíceis de resolver de forma prática e rápida e estudados a fundo em busca de uma prova de que um algoritmo ligeiro exista, tal que P = NP.

2.2.1 Exatamente-1 3-SAT

Esse algoritmo é uma especificação e variação do 3-SAT, em que o problema é determinar se numa fórmula com as restrições normais do 3-SAT, existe satisfação quando exatamente uma das três variáveis em cada cláusula é verdadeira e as outras duas são obrigatoriamente falsas. Para resolver um caso desses, usamos um operador ternário (que chamaremos de R), o qual será marcado como verdadeiro somente se a cláusula a que ele está ligado tiver as características de exatamente-1 3-SAT.

2.2.2 Não-Iguais 3-SAT

Também chamado de NAE 3-SAT (Not-All-Equal 3-SAT), essa é mais uma variação do 3-SAT, onde, dada uma fórmula em FNC com três literais por cláusula, tentamos determinar se existe uma valoração para as variáveis de modo que em nenhuma cláusula as três tenham o mesmo valor verdadeiro.

2.3 O TEOREMA DA DICOTOMIA DE SCHAEFER

Falamos sobre vários tipos de restrições que podem ser feitas para resolver o problema da satisfazibilidade e muitas delas tratam em específico de conjunções de sub-fórmulas, como é o caso do 2-SAT, 3-SAT e HORN-SAT. Embora cada uma delas trate de uma especificação diferente, todas elas entram na descrição e categorização de Schaefer: ele coloca que para qualquer restrição a operadores booleanos (nossos conhecidos conectivos lógicos) que podem ser usados para criar essas subfórmulas, seu problema de satisfazibilidade correspondente ou faz parte do grupo de complexidade P ou é NP-completo. Alguns casos especiais desse teorema poderiam ser o 2-SAT, HORN-SAT, exatamente-1 3-SAT e o não-iguais 3-SAT.

3 EXTENSÕES DO SAT

A partir de 2003, uma extensão do SAT vem ganhando alguma popularidade e sendo mais usada: é chamada SMT (Satisfiability Modulo Theory, que traduziremos livremente para Teoria de Satisfazibilidade Modular). Uma instância em SMT é uma generalização de um problema de satisfazibilidade booleano, na qual as variáveis permitem predicados e quantificadores. Para o SMT, uma entrada válida poderia ser $\forall x \forall y \exists z \, (x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z)$, por exemplo, algo que os problemas de SAT até então discutidos nesse trabalho não conseguiriam abordar.

Enquanto que os algoritmos de SAT estudados aqui se preocupam em achar uma resposta qualquer que satisfaça uma fórmula proposicional dada, existem outros que tem funções mais específicas e se propõem a resultar informações diferentes a partir dessa fórmula, ainda que ligados à satisfazibilidade dela. O #SAT, por exemplo, é um algoritmo que conta quantos conjuntos de valorações satisfazem a fórmula; ele é um problema de contagem, não de decisão, e faz parte do grupo de complexidade #P-completo. Outro algoritmo interessante que podemos enumerar é o UNIQUE-SAT, que é o problema de determinar se existe exatamente uma valoração que satisfaça a fórmula.

4 CONSIDERAÇÕES FINAIS

Aprofundando conhecimentos no assunto do problema da satisfazibilidade, podemos ver melhor as diferentes conexões entre as áreas teóricas e aplicadas da

ciência da computação. A lógica formal permite a manipulação de fórmulas e instâncias booleanas e predicativas em diferentes estruturas de dados, que poderão ser usadas em algoritmos práticos. Embora a complexidade desse problema impossibilite, atualmente, acharmos um algoritmo rápido para o problema em questão, usamos de restrições nas entradas para conseguir desenvolver e estudar resoluções para um número vasto de instâncias. Finalmente, mesclamos conhecimentos diversos para encontrar e apresentar algoritmos que consigam, dadas suas devidas restrições, tratar uma fórmula booleana e encontrar uma valoração satisfazível para ela.

REFERÊNCIAS BIBLIOGRÁFICAS

HUTH, Michael; RYAN, Mark. L**ógica em ciência da computação**: Modelagem e Argumentação sobre Sistemas. 2 ed. Rio de Janeiro: LTC, 2008. 322 p.

FUX, Jaques. **Análise de Algoritmos SAT para Resolução de Problemas Multivalorados**. 2004. 86 p. Dissertação (Mestrado em Ciência da Computação)- Universidade Federal de Minas Gerais, [S.l.], 2004. Disponível em: <https://www.dcc.ufmg.br/pos/cursos/defesas/23M.PDF>. Acesso em: 09 dez. 2017.

TAVARES, Cláudia Fernanda O. K. **Prova Automática de Satisfatibilidade Módulo Teoria Aplicada ao Método B**. 2007. 136 p. Dissertação (Mestrado em Sistemas e Computação)- Universidade Federal do Rio Grande do Norte, [S.l.], 2007. Disponível em: <http://ftp://ftp.ufrn.br/pub/biblioteca/ext/bdtd/ClaudiaFCKT.pdf>. Acesso em: 10 dez. 2017.

DA SILVA NUNES, Igor Ferreira; DIAS, Henrique Santos; SILVA, Welton Aristides. **Solucionando o problema da satisfatibilidade booleana por meio de algoritmo genético**. 2015. 9 p. Dissertação (Bacharel em Engenharia Elétrica e Mecânica)- Universidade Federal de Goiás, [S.l.], 2015. Disponível em: <https://www.trabalhosgratuitos.com/Exatas/Inform%C3%A1tica/Problema-da-Satisfatibilidade-Booleana-988040.html>. Acesso em: 10 dez. 2017.

ANDREW GIBIANSKY. **Writing a sat solver**. Disponível em: <http://andrew.gibiansky.com/blog/verification/writing-a-sat-solver/>. Acesso em: 04 dez. 2017.

ARQUIVO. **Basics of sat solving algorithms**. Disponível em: <http://www.cs.utexas.edu/users/moore/acl2/seminar/2008.12.10-swords/sat.pdf>. Acesso em: 07 dez. 2017.

ARQUIVO. **Capítulo 34 - análise e síntese de algoritmos**. Disponível em: <https://fenix.tecnico.ulisboa.pt/downloadfile/3779571834099/ch34.pdf>. Acesso em: 06 dez. 2017.

ARQUIVO. **Satisfiability solvers**. Disponível em: <https://ece.uwaterloo.ca/~vganesh/teaching/w2013/satsmt/satsolvers-survey.pdf>. Acesso em: 05 dez. 2017.

MATH U CODE. **Understanding sar by implementing a simple sat solver in python**. Disponível em: <http://sahandsaba.com/understanding-sat-by-implementing-a-simple-sat-solver-in-python.html>. Acesso em: 06 dez. 2017.

OA. The boolean satisfiability problem. Disponível em: <http://0a.io/boolean-satisfiability-problem-or-sat-in-5-minutes/>. Acesso em: 05 dez. 2017.

STACK OVERFLOW. **Algoritmo - o que é um problema np completo?**. Disponível em: <https://pt.stackoverflow.com/questions/34104/o-que-%c3%a9-um-problema-np-completo>. Acesso em: 03 dez. 2017.

WIKIPEDIA. **2-satisfiability**. Disponível em: <https://en.wikipedia.org/wiki/2-satisfiability>. Acesso em: 08 dez. 2017.

WIKIPEDIA. **Boolean satisfiability problem**. Disponível em: <https://en.wikipedia.org/wiki/boolean_satisfiability_proble>. Acesso em: 03 dez. 2017.

WIKIPEDIA. **Forma normal disjuntiva**. Disponível em: <https://pt.wikipedia.org/wiki/forma_normal_disjuntiva>. Acesso em: 05 dez. 2017.

WIKIPEDIA. **Horn-satisfiability**. Disponível em: <https://en.wikipedia.org/wiki/horn-satisfiability>. Acesso em: 07 dez. 2017.

WIKIPEDIA. **Not-all-equal 3-satisfiability**. Disponível em: <https://en.wikipedia.org/wiki/not-all-equal_3-satisfiability>. Acesso em: 09 dez. 2017.

WIKIPEDIA. **Np-completeness**. Disponível em: <https://en.wikipedia.org/wiki/np-completeness>. Acesso em: 03 dez. 2017.

WIKIPEDIA. **Satisfazibilidade de horn**. Disponível em: <https://pt.wikipedia.org/wiki/satisfatibilidade_de_horn>. Acesso em: 07 dez. 2017.

WIKIPEDIA. **Satisfiability modulo theories**. Disponível em: <https://en.wikipedia.org/wiki/satisfiability_modulo_theories>. Acesso em: 11 dez. 2017.

WIKIPEDIA. **Schaefer's dichotomy theorem**. Disponível em: <https://en.wikipedia.org/wiki/schaefer%27s_dichotomy_theorem>. Acesso em: 11 dez. 2017.

WIKIPEDIA. **Unit propagation**. Disponível em: <https://en.wikipedia.org/wiki/unit_propagation>. Acesso em: 07 dez. 2017.

YOUTUBE. P vs. np and the computational complexity zoo. Disponível em: <https://www.youtube.com/watch?v=yx40hbahx3s>. Acesso em: 03 dez. 2017.

YOUTUBE. T**he boolean satisfiability problem: advanced math**. Disponível em: <https://www.youtube.com/watch?v=sbttrwpjwi0>. Acesso em: 02 dez. 2017.