Satisfatibilidade Booleana: Implementação e Análise Comparativa com e sem Backtracking

***Abstract*—This article presents the implementation and analysis of two algorithms to check the satisfiability of Boolean formulas in conjunctive normal form (CNF). The first algorithm uses a brute force approach to generate all variable assignments, while the second algorithm employs the backtracking technique. The performance comparison between the two algorithms was carried out based on execution time and computational efficiency.**

***Keywords*—Boolean Satisfaction (SAT), Backtracking, CNF Formulas, Computational Complexity, Brute Force Algorithms.**

I. Introdução

A Satisfatibilidade Booleana (SAT) é um dos problemas fundamentais da teoria da complexidade computacional e aparece frequentemente em várias áreas da ciência da computação, incluindo verificação formal, planejamento e inteligência artificial. O problema SAT consiste em determinar se existe uma atribuição de valores de verdade (verdadeiro ou falso) às variáveis de uma fórmula booleana que torne a fórmula verdadeira.

A técnica de backtracking é uma abordagem comum para a resolução de problemas combinatórios como o SAT, sendo mais eficiente do que a busca exaustiva em muitos casos. Este artigo compara dois algoritmos para a resolução de SAT: um que explora todas as possíveis atribuições de variáveis (força bruta) e outro que utiliza backtracking para otimizar o processo de busca por uma solução.

II. Metodologia

2.1 Algoritmo sem Backtracking (Força Bruta)

O primeiro código implementado faz uma verificação exaustiva de todas as atribuições possíveis de variáveis. Ele lê as fórmulas booleanas em formato CNF, representadas por cláusulas disjuntivas de literais, e tenta cada combinação possível de valores binários para determinar se alguma atribuição satisfaz todas as cláusulas.

Funcionamento: Gera todas as atribuições possíveis de variáveis (2^n, onde n é o número de variáveis). Verifica se alguma dessas atribuições torna a fórmula verdadeira. Se uma atribuição for válida, a fórmula é considerada satisfatível (SAT); caso contrário, é insatisfatível (UNSAT).

Complexidade Temporal: O algoritmo sem backtracking possui uma complexidade O(2^n), onde n é o número de variáveis. Isso ocorre porque ele gera e testa todas as combinações possíveis de atribuições, que crescem exponencialmente com o número de variáveis.

A. Algoritmo com Backtracking

O segundo código utiliza a técnica de backtracking para otimizar a busca por uma solução. Ao invés de gerar todas as combinações de variáveis, o algoritmo faz atribuições parciais e interrompe o processo quando detecta que uma atribuição não pode levar a uma solução satisfatória.

Funcionamento: Atribui valores às variáveis uma por uma. A cada atribuição, verifica se a fórmula ainda pode ser satisfatível com base nas cláusulas atuais. Se uma atribuição parcial falhar, o algoritmo "retrocede" (backtrack) e tenta uma nova atribuição. O processo continua até encontrar uma atribuição satisfatória ou até esgotar todas as possibilidades.

Complexidade Temporal: O uso do backtracking pode reduzir significativamente o número de combinações testadas em comparação à força bruta. Em cenários onde o backtracking é efetivo, a complexidade temporal pode ser reduzida para algo próximo de O(b^d), onde b é o fator de ramificação e d é a profundidade máxima da árvore de recursão. No entanto, no pior caso, o algoritmo pode ainda ter desempenho exponencial.

III. Resultados e Discussão

Os dois algoritmos foram testados utilizando o mesmo conjunto de fórmulas CNF extraídas de arquivos no formato DIMACS. Os tempos de execução foram medidos em segundos, e o desempenho dos algoritmos foi comparado com base nos seguintes parâmetros:

Tempo de execução individual de cada fórmula.

Tempo de execução total para todas as fórmulas.

A. Desempenho do Algoritmo Sem Backtracking

O algoritmo sem backtracking apresentou um tempo de execução significativamente maior para fórmulas com um número elevado de variáveis devido ao crescimento exponencial das combinações possíveis. O tempo total de execução para resolver as fórmulas CNF testadas foi, em média, muito maior que o do algoritmo com backtracking.

B. Desempenho do Algoritmo Com Backtracking

O algoritmo com backtracking se mostrou mais eficiente na maioria dos casos, especialmente em fórmulas que contêm variáveis cujas atribuições inviabilizam rapidamente a solução. O tempo de execução total foi drasticamente reduzido em comparação ao método de força bruta.

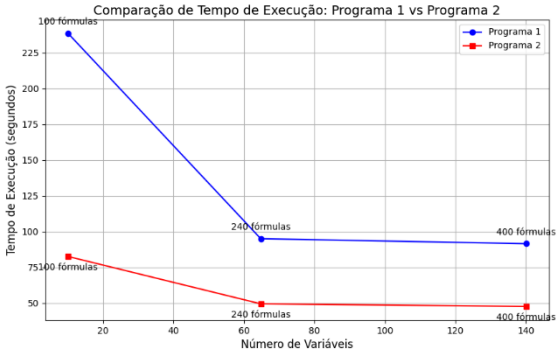

Figura 1 – Gráfico com os Resultados

IV. Análise de Complexidade

Para o primeiro código, que realiza uma busca exaustiva verificando todas as possíveis combinações de atribuições de variáveis, a complexidade pode ser descrita da seguinte maneira:

Número de variáveis: n

Número total de combinações de atribuições: 2^n

Como o código verifica todas as possíveis atribuições para encontrar uma que satisfaça a fórmula, a complexidade total do algoritmo é O(2^n), onde n é o número de variáveis na fórmula CNF. Esse comportamento é esperado em problemas NP-completos como o SAT, onde uma solução determinística para verificar a satisfatibilidade pode demandar tempo exponencial em relação ao número de variáveis.

No segundo código, que implementa backtracking, a complexidade teórica ainda é exponencial no pior caso. O backtracking pode, em alguns casos, reduzir o número de atribuições testadas ao descartar atribuições que claramente não levam a uma solução válida. Entretanto, no pior cenário, todas as possíveis combinações podem ainda precisar ser verificadas.

Complexidade no pior caso: O(2^n)

Complexidade média (com otimização de backtracking): O(k * 2^n), onde k é uma constante que depende do grau de poda realizada pelo algoritmo.

Mesmo com a poda, como o problema SAT é NP-completo, a complexidade permanece exponencial no pior caso. No entanto, o uso de técnicas de backtracking pode melhorar significativamente o desempenho prático para fórmulas com características específicas.

Conclusão

Este artigo apresentou uma análise comparativa de dois algoritmos para a verificação de satisfatibilidade booleana. A abordagem de backtracking demonstrou ser significativamente mais eficiente do que a busca exaustiva para fórmulas CNF com um grande número de variáveis. Embora ambos os algoritmos possam eventualmente determinar a satisfatibilidade de qualquer fórmula, o backtracking é preferível em cenários práticos devido à sua capacidade de podar grandes porções do espaço de busca, tornando-o mais eficiente em muitos casos.

References

[1] Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: an efficient SMT solver. In Proceedings of the Theory and practice of software, 14th international conference on Tools and algorithms for the construction and analysis of systems (TACAS'08/ETAPS'08). Springer-Verlag, Berlin,

[2] https://en.wikipedia.org/wiki/Satisfiability

[3] https://en.wikipedia.org/wiki/2-satisfiability

[4] https://github.com/Z3Prover/z3