Universidade Federal de Ouro Preto
Departamento de Computação
Projeto e Análise de Algoritmos

TRABALHO PRÁTICO

Lívia Stéffanny de Sousa
Pedro Augusto de Sousa
Thiago

Ouro Preto
September 23, 2024

Contents

1 Introdução

Este trabalho prático tem como objetivo a implementação de três algoritmos fundamentais para a resolução de problemas clássicos da computação: o problema de Satisfabilidade, o problema de Clique e o problema do Conjunto Independente. Através da implementação desses algoritmos, aplicaremos técnicas como Backtracking e Branch and Bound, além de realizar uma redução polinomial entre o problema de Clique e Conjunto Independente.

Os problemas serão resolvidos conforme descrições específicas:

- O problema da **Satisfabilidade** (SAT) será resolvido utilizando a técnica de Backtracking.
- O problema do **Clique** será resolvido com o algoritmo Branch and Bound.
- O problema do **Conjunto Independente** será abordado via redução ao problema do Clique.

Este relatório apresenta a metodologia aplicada, a descrição dos algoritmos, os resultados obtidos e uma análise das decisões tomadas.

2 Metodologia

O trabalho foi dividido em três partes principais, cada uma relacionada aos problemas propostos. Implementamos os algoritmos utilizando a linguagem de programação C++. As entradas e saídas foram formatadas de acordo com a especificação do trabalho, e as instâncias de cada problema foram testadas para verificar a corretude dos algoritmos.

2.1 Problema da Satisfabilidade

O problema da Satisfabilidade foi resolvido utilizando o algoritmo de Backtracking. O algoritmo busca, de forma recursiva, atribuir valores-verdade às variáveis de forma que a fórmula booleana na forma normal conjuntiva (CNF) seja satisfeita.

$$(x \vee y \vee z)(x \vee \overline{y})\,(y \vee \overline{z})(\overline{x} \vee z)$$
$$x = \top,\ y = \top,\ z = \top$$

Figure 1: Exemplo de uma fórmula CNF e sua solução

A entrada do programa é fornecida em um arquivo de texto, onde cada linha corresponde a uma cláusula, e os números indicam a presença ou ausência das variáveis em cada cláusula.

2.2 Problema do Clique

O problema do Clique foi resolvido utilizando o algoritmo de Branch and Bound. Este método é eficaz para encontrar o maior subconjunto de vértices em um grafo em que todos são adjacentes entre se. O algoritmo foi implementado de forma que a entrada seja um grafo representado por uma matriz de adjacência.

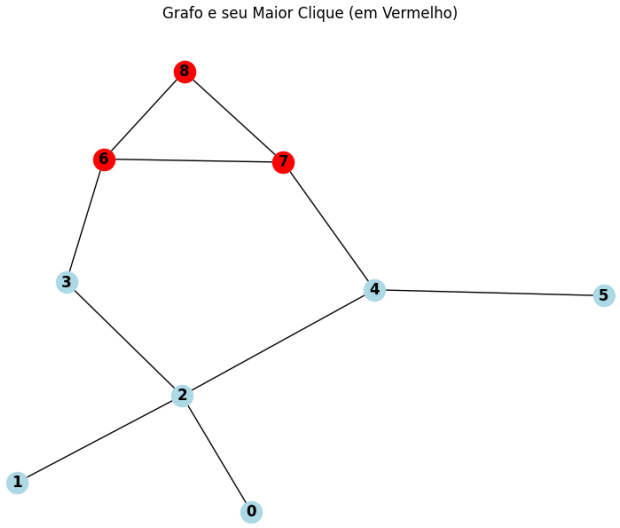

Figure 2: Exemplo de um grafo e o clique máximo correspondente

2.3 Problema do Conjunto Independente

Para o problema do Conjunto Independente, foi realizada uma redução polinomial ao problema do Clique que foi resolvido anteriormente. Um conjunto independente é retornado pelo clique com o grafo complementar, ou seja, ele irá retornar o maior subconjunto de vértices em um grafo que nenhum é adjacente a qualquer outro do subconjunto.

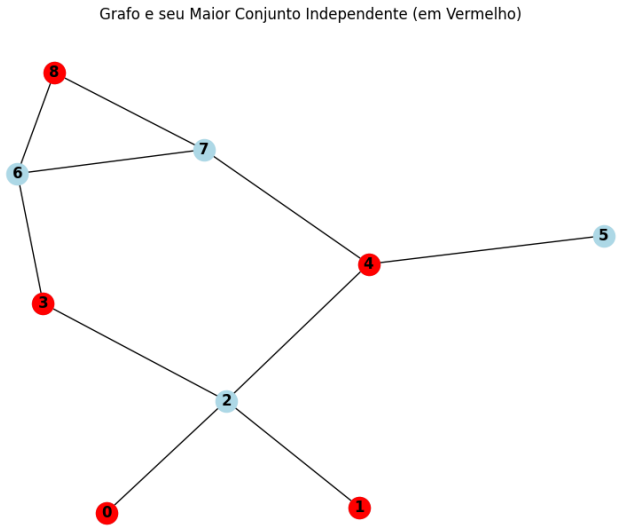

Figure 3: Ilustração da redução do conjunto independente ao problema do Clique

3 Algoritmos

A seguir, apresentamos os algoritmos implementados.

```
função backtracking(sat, position):
    // Verifica se a fórmula foi satisfeita
    se position == sat.getVarsNum() então
        retornar verdadeiro

    // Atribui falso à variável atual
    sat.vars.adicionar(falso)

    // Testa as variáveis já atribuídas
    se sat.checkVars() então
        se backtracking(sat, position + 1) então
            retornar verdadeiro

    sat.vars.removerÚltimo() // remove o valor atribuído

    // Atribui verdadeiro à variável atual
    sat.vars.adicionar(true)

    // Testa as variáveis já atribuídas
    se sat.checkVars() então
        se backtracking(sat, position + 1) então
            retornar verdadeiro

    sat.vars.removerÚltimo() // remove o valor atribuído

    // Nenhuma atribuição levou a uma solução
    retornar falso
```

Figure 4: Ilustrando o pseudocódigo do algoritmo de Backtracking

3.1 Algoritmo de Backtracking para Satisfatibilidade

O algoritmo de Backtracking percorre recursivamente todas as possíveis atribuições de valores para as variáveis da fórmula CNF. Contudo ele usa uma função que avalia os valores das variáveis em cada atribuição, dessa forma, ele consegue identificar uma atribuição invalida antes de atribuir outra, dessa forma podando a arvore de busca. A abordagem é viável para instâncias pequenas, mas seu tempo de execução cresce exponencialmente.

3.2 Algoritmo de Branch and Bound para Clique

O algoritmo de Branch and Bound é utilizado para explorar o espaço de busca, limitando subárvores que não podem conter soluções ótimas. Para o problema do Clique, o algoritmo seleciona vértices e constrói cliques progressivamente, descartando subconjuntos de vértices que não podem formar um clique maior.

3.3 Redução do Conjunto Independente ao Clique

A redução do problema do Conjunto Independente ao problema do Clique explora a estrutura complementar dos grafos. Isso permite a resolução eficiente do problema, reaproveitando o algoritmo de Clique implementado.

```
função branchAndBound(clique, profundidade, candidatos):
    // Se não há mais candidatos
    se candidatos estiver vazio então
        // Atualiza o clique máximo se o clique atual for maior
        se tamanho de clique.currClique > tamanho de clique.maxClique então
            clique.maxClique = clique.currClique
        retornar

    // Condição de poda: se o clique atual mais os candidatos restantes não puder ser melhor
    se tamanho de clique.currClique + tamanho de candidatos <= tamanho de clique.maxClique então
        retornar

    // Pegar as arestas do grafo
    arestas = clique.getArestas()

    // Tentar expandir o clique atual
    para cada vértice v em candidatos faça:
        adicionar v ao clique.currClique
        novosCandidatos = conjunto vazio

        // Novos candidatos são os vértices adjacentes ao vértice atual
        para cada vértice u em candidatos faça:
            se arestas[v][u] então
                adicionar u a novosCandidatos

        // Chamar recursivamente para expandir o clique
        branchAndBound(clique, profundidade + 1, novosCandidatos)

        remover v de clique.currClique
```

Figure 5: Exemplo de um pseudocódigo do algoritmo Branch and Bound

4 Resultados

Os algoritmos foram testados com 8 instâncias para cada problema. A seguir, apresentamos os resultados obtidos em 3 instancias para cada problema.

4.1 Satisfatibilidade

- **Instância 1**: Fórmula CNF com 3 variáveis, tempo de execução: 4.3221e-05 s.
- **Instância 2**: Fórmula CNF com 20 variáveis, tempo de execução: 0.0007509 s.
- **Instância 3**: Fórmula CNF com 100 variáveis, tempo de execução: 0.00203452 s.

4.2 Clique

- **Instância 1**: Grafo com 9 vértices, tempo de execução: 3.0664e-05 s.
- **Instância 2**: Grafo com 50 vértices, tempo de execução: 0.664854 s.
- **Instância 3**: Grafo com 100 vértices, tempo de execução: 51.5461 s.

4.3 Conjunto Independente

- **Instância 1**: Grafo com 9 vértices, tempo de execução: 0.000298246 s.
- **Instância 2**: Grafo com 50 vértices, tempo de execução: 0.247918 s.
- **Instância 3**: Grafo com 100 vértices, tempo de execução: 102.436 s.

5 Conclusão

Este trabalho prático permitiu implementar e analisar três algoritmos clássicos da ciência da computação. O uso das técnicas de Backtracking e Branch and Bound se mostrou eficaz na resolução dos problemas propostos, e a redução do problema do Conjunto Independente ao Clique simplificou sua resolução.

Os resultados demonstram que os algoritmos são eficientes para as instâncias testadas, com tem pos de execução aceitáveis, considerando o número de variáveis e vértices envolvidos.

6 Referências

- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms*. MIT Press.
- Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company.
- Karp, R. M. (1972). Reducibility Among Combinatorial Problems. *Complexity of Computer Computations*.