Universidade Federal do Rio Grande do Sul
Instituto de Informática
Complexidade de Algoritmos

Coloração de Grafos

Complexidade de Algoritmos (INF05510) – Turma A
Profª Mariana Kolberg

13 de junho de 2012

Rafael Vanz
Tiago Biazus

Resumo

Neste artigo mostraremos que o Problema 3-Colorível é NP-Completo. Será mostrado que esse é um problema que pertence à classe NP e que é possível fazer uma redução de 3-SAT para 3-Colorível em tempo polinomial (3-SAT $\leq_p$ 3-COL).

1. Introdução

Dado um grafo G, podemos colorir os vértices deste grafo com k cores de tal forma que as extremidades de cada aresta tenha uma cor diferente? Este é o problema de coloração de grafos.

Para um grafo ser 1-colorível ele deve ser um grafo vazio, ou seja, um grafo sem arestas, e não nulo, ou seja, um grafo com pelo menos um vértice.

Para um grafo ser 2-colorível basta ele ser bipartido, ou seja, um grafo cujos vértices podem ser divididos em dois conjuntos, nos quais não há arestas entre vértices de um mesmo conjunto.

Para $k \geq 3$ temos um problema NP-Completo.

2. Classes de Complexidade

As classes de complexidade de problemas de grande importância para a Ciência da Computação são:

- **Classe P (Polynomial-time Algorithms)**: É o conjunto de todos os problemas que podem ser resolvidos por algoritmos determinísticos em tempo polinomial.
- **Classe NP (Nondeterministic Polynomial-time Algorithms)**: É o conjunto de todos os problemas que podem ser resolvidos por algoritmos não-determinísticos em tempo polinomial.
- **Classe NP-Difícil**: É o conjunto de todos os problemas que possuem uma redução para ele de um outro problema em tempo polinomial.
- **Classe NP-Completo**: É o conjunto de todos os problemas que podem, ou ser resolvidos por algoritmos não-determinísticos, ou ser verificados por algoritmos determinísticos, em tempo polinomial (pertence a NP).

Para mostrar que um problema está em NP, podemos apresentar um algoritmo não-determinístico que execute em tempo polinomial para resolver o problema (muito mais difícil de ser formalizado, geralmente é feito usando-se Máquinas de Turing Não-Determinísticas), ou um algoritmo determinístico que execute em tempo polinomial para verificar se uma dada solução é válida (este será o método usado nesse artigo por ser mais simples e de igual eficácia).

Os problemas NP-Completos têm grande importância na Ciência da Computação, pois essa classe tem a propriedade de que se qualquer problema NP-Completo puder ser resolvido em tempo polinomial, então todo problema em NP tem uma solução em tempo polinomial e P = NP, entretanto ainda não foi descoberto nenhum algoritmo de tempo polinomial que resolva um problema NP-Completo.

Para que o problema seja considerado NP-Completo ele deve satisfazer a duas condições:

i. Pertencer à classe NP.

ii. Ter um outro problema NP-Completo que possa ser reduzido a ele em tempo polinomial (pertente à NP-Difícil).

Neste artigo aplicaremos esses conceitos para mostrar que 3-Colorível é NP-Completo. Para a redução usaremos o problema NP-Completo 3-SAT e aplicaremos a seguinte redução 3-SAT $\leq_p$ *3-COL*.

3. Caracterização do problema

Uma k-coloração é uma função $f : V \rightarrow \{1, \ldots, k\}$ onde para cada aresta $\{u, v\}$ nós temos $f(u) \neq f(v)$. Se tal função existe para um dado grafo G, então G é k-colorível.

Para provar que 3-Colorível pertence a NP-Completo vamos primeiro provar que ele pertence à classe NP. Para fazer isso precisamos de uma coloração válida para um dado grafo, ou seja, um certificado. Também será necessário um algoritmo verificador de complexidade polinomial.

O segundo passo será provar que 3-Colorível pertence a classe NP-difícil. Para fazer isso faremos a redução de uma instância 3-SAT para 3-Colorível.

4. 3-Colorível pertence à classe NP

- Certificado: para cada nó uma cor de {1, 2, 3}
- Verificador: checar se para cada aresta (u, v), a cor de u é diferente da cor de v

Algoritmo de verificação

Entrada:

- Um grafo G = (V, A) onde V é um conjunto de vértices e A um conjunto de pares de vértices
- S[1 ... n]: um certificado, uma sequência de cores para os vértices
  Observações: cada vértice é representado, nesta implementação, como uma estrutura de dados que possui informações como nome do vértice (V[i].nome) , cor do vértice (V[i].cor) e número do vértice no array (V[i].numero). E varia de i = 1 até n, sendo n o número de vértices. O certificado é representado, nesta implementação, como um array de tamanho n, onde n representa o número de vértices de G, e que possui a informação cor para cada vértice. A sequência S do certificado possui somente três cores diferentes. A, o conjunto de pares de vértices, é representado como um array (A[1 … m]) de estruturas de dados que representam pares de vértices, onde m representa o número de arestas do grafo G. Exemplo: A[1].u e A[1].v representam os vértices da aresta 1.
- m e n, onde m representa o número de arestas e n o número de vértices

Saída:

- Uma resposta: SIM ou NÃO

```
1.  Para i:=1 até m faça
2.     se ( S[A[i].u.numero]   =  S[A[i].v.numero] )
3.     senão retorne NÃO
4.  Fim-Para
5.  retorne SIM
```

Observações: A[1].u representa um vértice da aresta 1. A[1].u.numero retorna o número do vértice no array V. S[A[1].u.numero] vai retornar a cor do vértice neste certificado. Na linha 1, i varia até o número de arestas do grafo G. Na linha 2 ocorre o teste para verificar se os vértices da aresta possuem corres diferentes no certificado. Se todos os pares de vértices das arestas tiverem cores diferentes, na linha 3 retorna SIM.

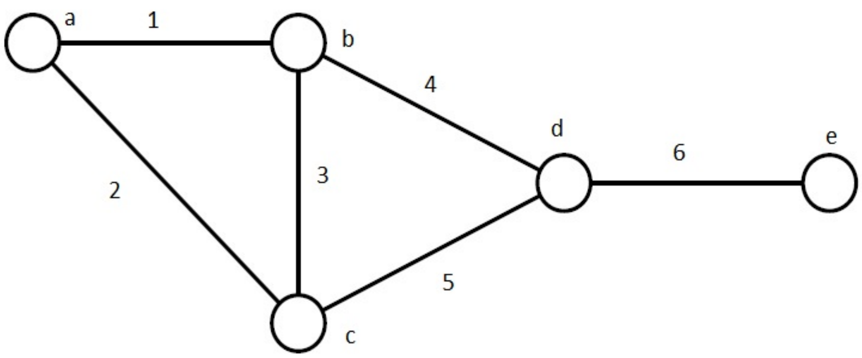
Cálculo da Complexidade

desemp[todas linhas] = desemp[1..5] = $\left(\sum_{i=1}^{m}(1)\right)$= *O(m)*

A complexidade O(m), linear, significa que o nosso algoritmo de verificação se encaixa dentro da definição de um problema NP, como queríamos demonstrar.

Exemplo

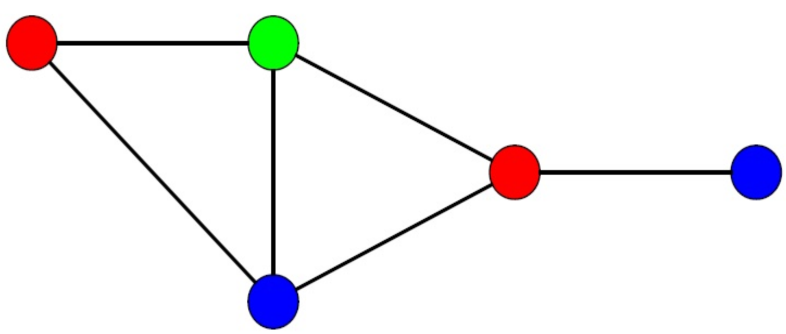
Seja G = (V, A), onde V é representado por um array de vértices V[1..5] e A um array de arestas A[1..6]



grafo G

Pergunta: Dado um certificado S = {vermelho, verde, azul, vermelho, azul}, ou seja, o vértice V[1] terá cor vermelha, V[2] verde, V[3] azul, V[4] vermelha e V[5] azul, e o grafo G, ele é 3-colorível?

Resposta: Aplicando o algoritmo verificador para este grafo e este certificado, obtemos a resposta SIM.



grafo G 3-colorido
nome dos vértices e número das arestas foram omitidos

5. 3-Colorível pertence à classe NP-Difícil (3SAT $\leq_p$ 3COL)

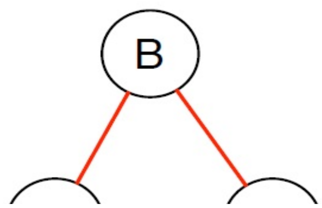
Seja x1, …, xn, C1, … , Ck uma instância de 3-SAT. Será mostrado como usar 3-coloração para resolver isso.

Vamos construir um grafo G que será 3-Colorível se a instância 3-SAT é satisfazível.

Para cada variável xi são criados dois nós em G, um para xi e um para ~xi.

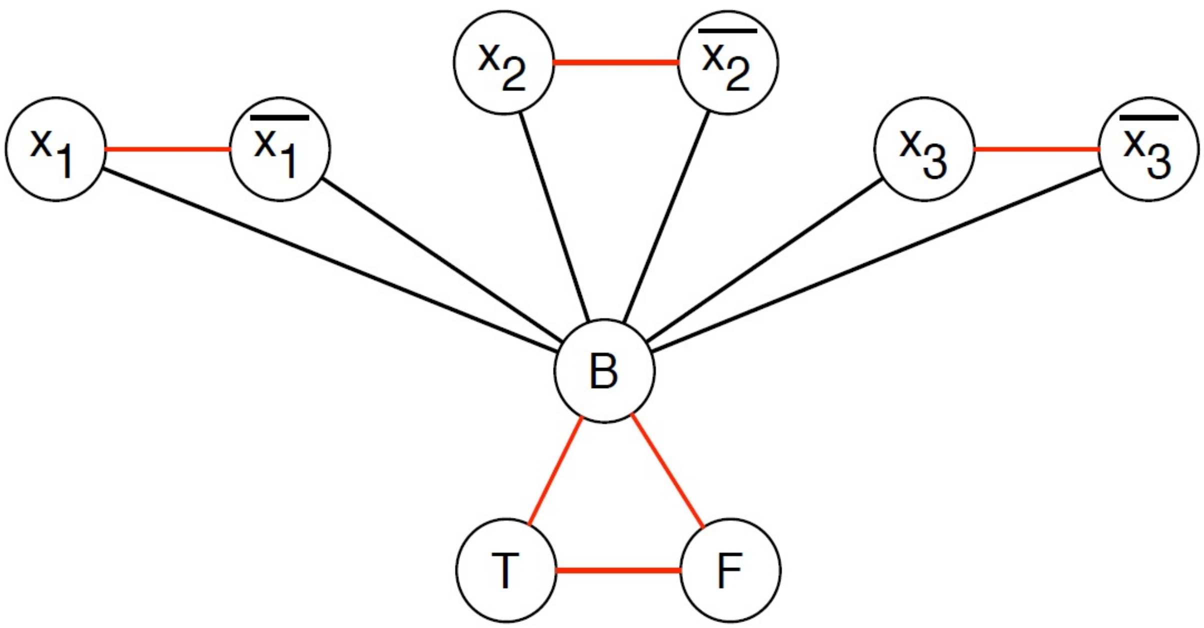
Conectar esses nós por uma aresta:

Criar três nós especias T, F e B unidos por um triângulo:



gadget

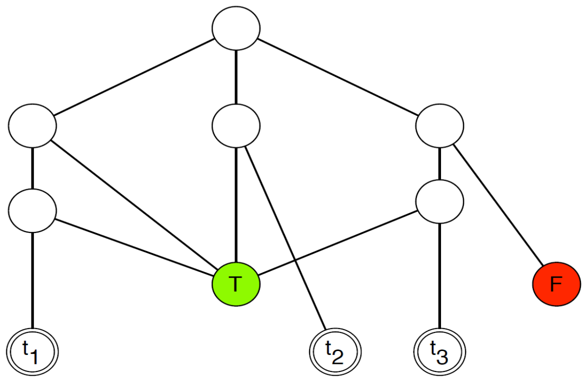
Conectar cada variável nó com o nó B:



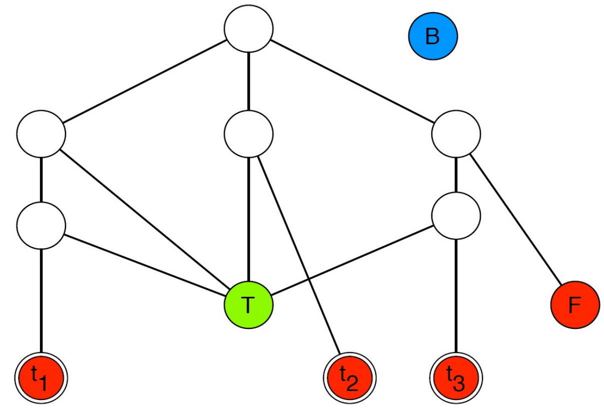
Propriedades:

- Cada xi e ~xi é colorido com cores diferentes
- Cada xi e ~xi deve ter cor diferente do nó B
- B, T e F devem receber cores diferentes
- O nó topo é colorível se um dos termos receber a cor para “true”

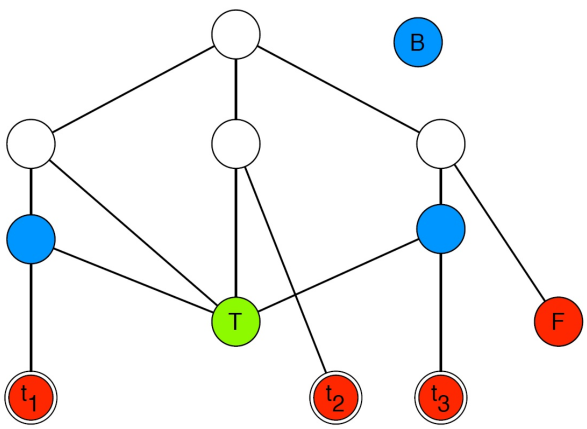
Seja C uma cláusula na fórmula. Pelo menos um dos seus termos deve ser “true”, porque se eles forem todos “false”, nós não conseguiremos colorir, como mostrado abaixo:



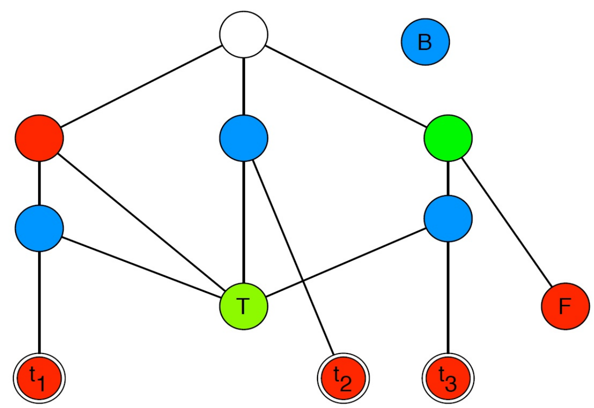
passo 1



passo 2



passo 3



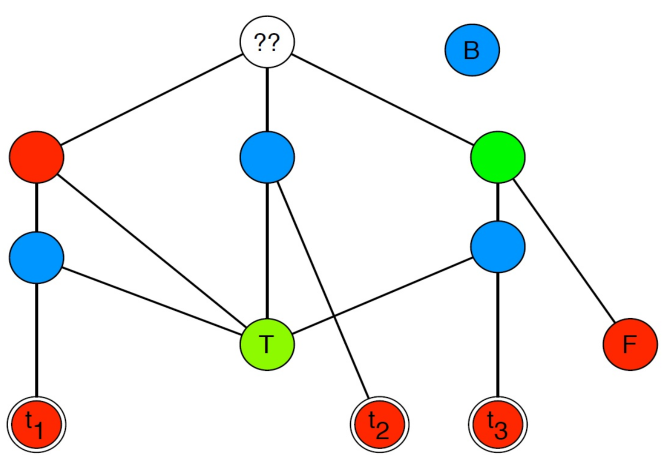
passo 4

o nó topo

Suponha que fórmula obteremos um de G fazendo:

•

T, F e B,



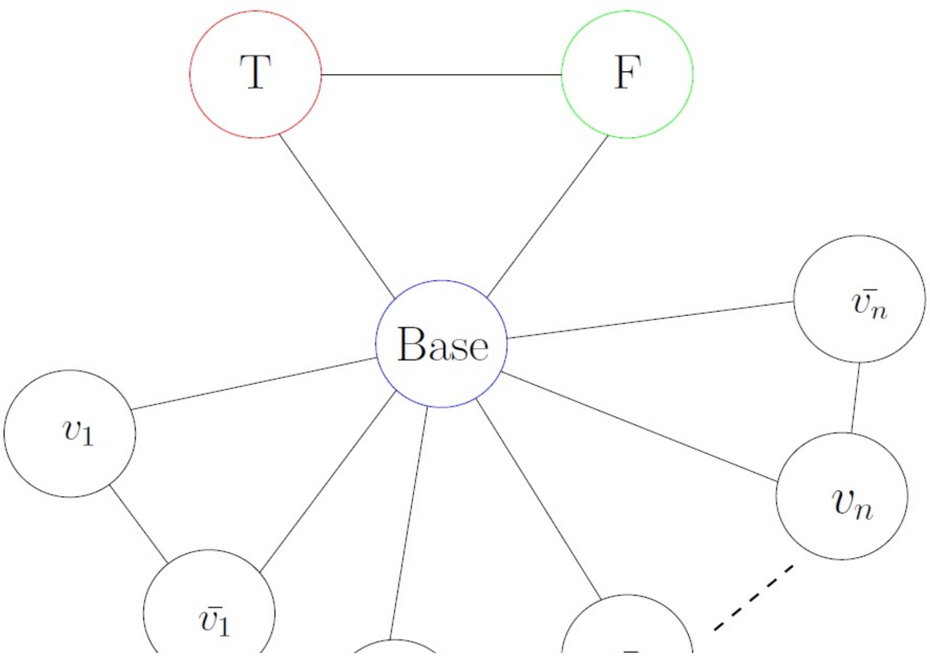
impossível colorir

tenhamos uma satisfazível. Nós grafo 3-Colorível

Colorindo os nós arbitrariamente,

com três cores diferentes

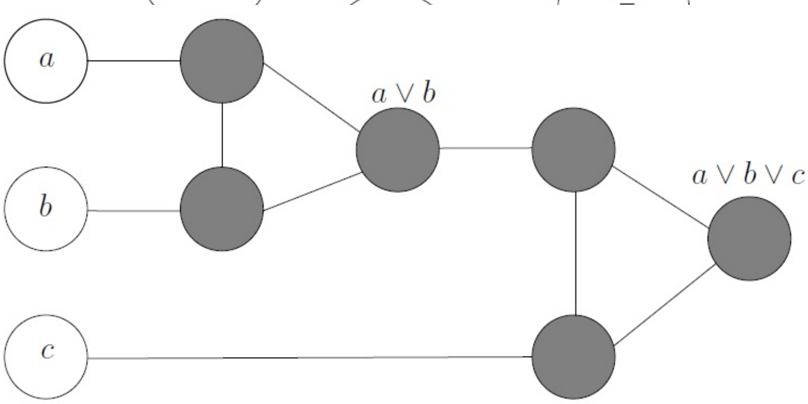
- Se xi = true, colorir vi com a mesma cor que T e ~vi com a mesma cor que F
- Se xi = false, fazer o oposto
- Estender essa coloração para as cláusulas "gadgets"



Para cada cláusula Cj = (a v b v c), criar um grafo gadget:

o grafo gadget é conectado aos nós correspondentes a a, b, c

precisamos implementar o grafo OR-gadget

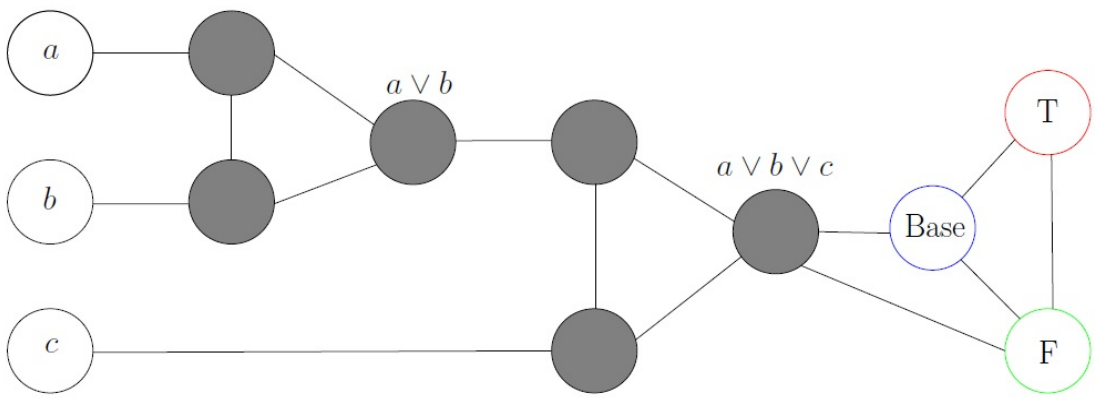


Propriedades:

Se a, b, c são coloridos com a cor de False em um grafo 3-Colorível então o nó saída do OR-gadget tem que ser colorido com a cor de False

- Se um dos nós a, b ou c é colorido com a cor de True então o OR-gadget é 3-Colorível de tal forma que o nó saída do OR-gadget é colorido com a cor de True

- Criar um triângulo com o nós True, False e Base
- Para cada variável xi dois nós vi e ~vi conectados em um triângulo com o nó Base em comum
- Para cada cláusula Cj = (a v b v c), adicionar um grafo OR-gadget com nós de entrada a, b, c e conectar o nó saída do OR-gadget aos nós False e Base



Corretez da Redução:

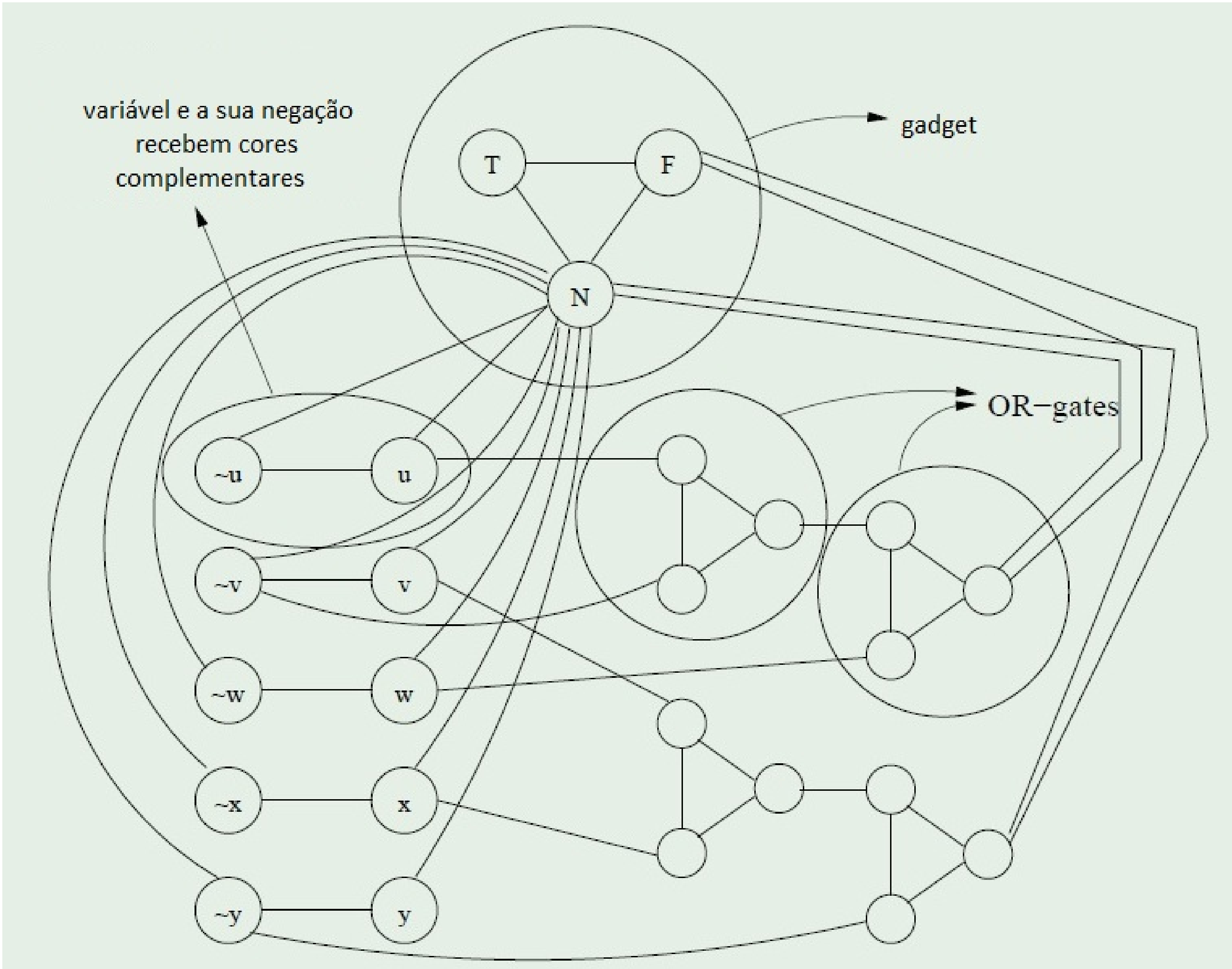
Se a fórmula 3-SAT é satisfazível implica que o grafo G é 3-Colorível

- Se xi recebe True, colorir o nó vi com a cor de True e ~vi com a cor de False
- Para cada cláusula Cj = (a v b v c) ao menos um dos nós a, b ou c é colorido com a cor de True. OR-gadget para Cj pode ser 3-Colorível de tal forma que a saída é True

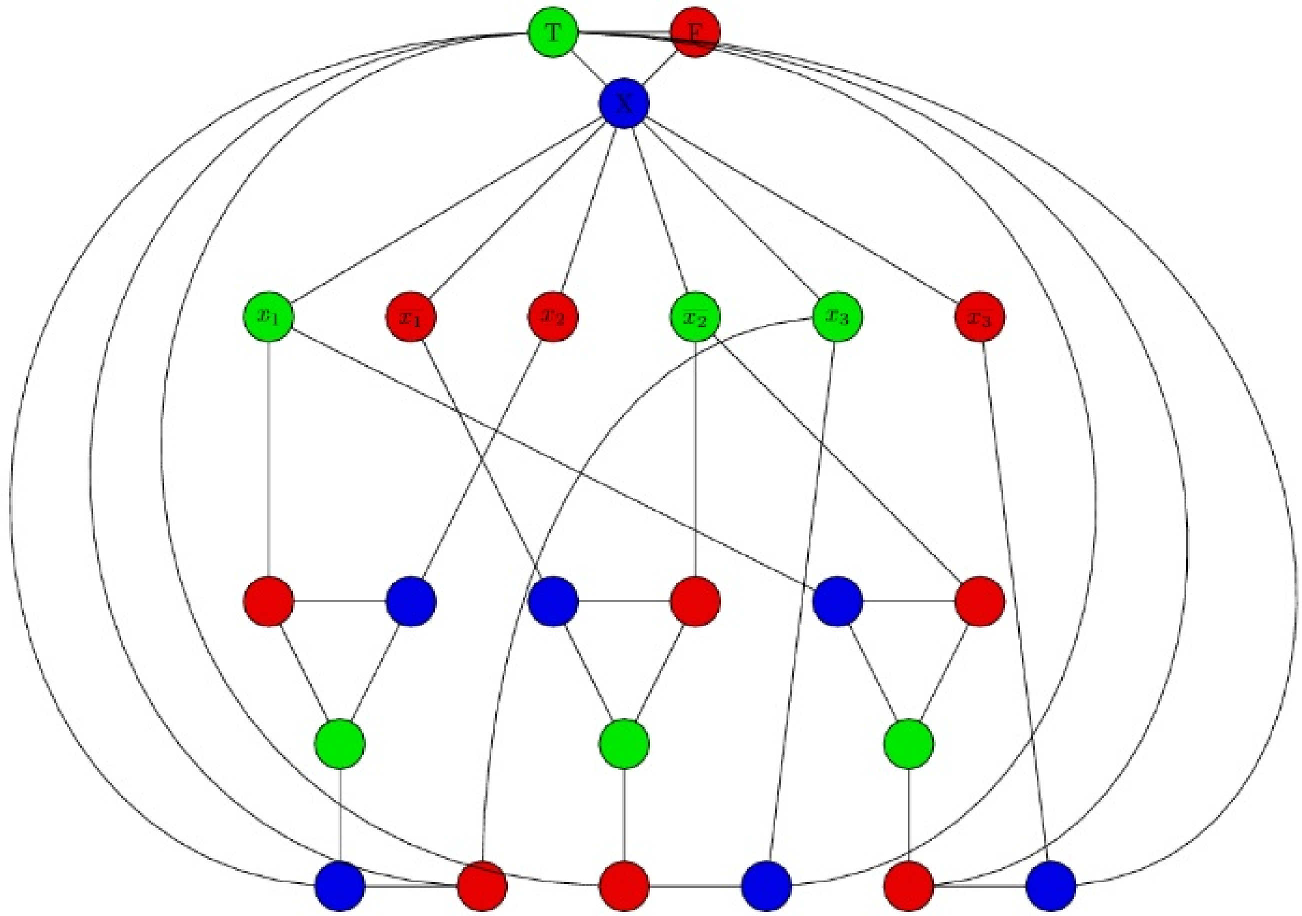
Se o grafo G é 3-Colorível implica que a fórmula 3-SAT é satisfazível

- Se vi é colorido com a cor de True então xi recebe True
- Considere qualquer cláusula Cj = (a v b v c). Não pode a, b e c serem False. Se forem False, a saída do OR-gadget para Cj tem que ser colorido com a cor de False, mas a saída é conectada ao nó Base e ao nó False, ou seja, isso não pode ocorrer!

Exemplos:



Grafo correspondente à fórmula $\Phi = (u \vee \neg v \vee w) \wedge (v \vee x \vee \neg y)$



¬x3)

A fórmula é satisfazível. (x1 = ~x2 = x3 = True) e (~x1 = x2 = ~x3 = False)

Observação: note que no exemplo acima foi feita uma simplificação, o terceiro nó da segunda OR-gate foi "colapsado" com o nó True do gadget principal. Também foram omitidas as arestas entre os pares x_i e $\sim x_i$.

Algoritmo de redução

Entrada:

- Uma instância de 3-SAT
- n, onde n é o número de variáveis
- m, onde m é o número de cláusulas

Saída:

- Um grafo G

Observação: o grafo G gerado na saída poder ser 3-Colorível se a instância de 3-SAT é satisfazível

```
1.  Criar gadget
2.  Para i:=1 até n faça
3.      Criar variáveis (xi e ~xi)
4.      Conectar (xi e ~xi)
5.      Conectar à B (xi e ~xi)
6.  Fim-Para
7.  Para i:=1 até m faça
8.      Criar cláusulas (i)
9.  Fim-Para
10. Retorne G
```

Cálculo da complexidade do algoritmo

desemp[todas linhas] = desemp[1] + desemp[2..6] + desemp[7..9]
desemp[2..6] $=\left(\sum_{i=1}^{n}(1+1+1)\right)$ *= O(3n) = O(n)*
desemp[7..9] $=\left(\sum_{i=1}^{m}(1)\right)$ *= O(m)*

desemp[todas linhas] = O(n + m)

A complexidade O(n + m), caracteriza nosso algoritmo não só como uma solução de tempo polinomial, mas de complexidade linear.

6. Conclusão

Nesse artigo mostramos que o problema 3-Colorível pertence à classe NP (possui algoritmo de verificação em tempo polinomial) e também demonstramos que ele pertence à classe NP-Difícil, através da redução do problema 3-SAT, que é NP-Completo, para 3-Colorível com um algoritmo em tempo polinomial. Esse problema atende as duas condições necessárias para ser NP-Completo (pertencer a NP e 3-SAT $\leq_p$ 3-COL), portanto o problema 3-Colorível é NP-Completo.

7. Bibliografia

TARDOS, Éva; KLEINBERG, John. Algorithm Design

CORMEN, Thomas H.; LEISERSON, Charles E.; RIVEST, Ronald L.; STEIN, Clifford. Algoritmos: teoria e prática. Tradução da 2ª edição (americana). Rio de Janeiro: Elsevier: Editora Campus, 2002 – 6ª Reimpressão.

TOSCANI, Laira Vieira; VELOSO, Paulo A. S. Complexidade de Algoritmos. 2ª Edição. Porto Alegre: Bookman: Instituto de Informática da UFRGS, 2008.

RITT, Marcus; BURIOL, Luciana S.; PRESTES, Édson. Algoritmos e Complexidade. Notas de aula, Instituto de Informática, UFRGS, Porto Alegre, RS: mar. 2010. Disponível em http://moodle.inf.ufrgs.br/mod/resource/view.php?id=26499.

MIYAZAWA, Flávio Keidi. Complexidade de Algoritmos. Notas de aula, UNICAMP, Campinas, SP. Disponível em www.ic.unicamp.br/~fkm/lectures/introcomp.pdf.

GRONER, Loiane. NP-Completude. Monografia, FAESA, Faculdades Integradas Espírito-Santenses, Vitória, ES: 2006. Disponível em http://www.loiane.com/2009/08/problemas-p-versus-np/.

CARVALHO, Marco Antonio Moreira de. Problemas NP-Completos. Slides, ITA. Disponível em http://www.comp.ita.br/~mamc/folhetos/6.pdf.

Discussão sobre como avaliar a dificuldade de colorir um grafo em um jogo:
http://stackoverflow.com/questions/5513805/how-could-i-evaluate-the-difficulty-of-a-graph-coloring-puzzle

Grafos Planares:
http://mathworld.wolfram.com/PlanarGraph.html

Grafo de Goldner-Harary:
http://en.wikipedia.org/wiki/Goldner%E2%80%93Harary_graph

Algoritmos para coloração de grafos:
http://scienceblogs.com/goodmath/2007/06/graph_coloring_algorithms_1.php

Slides sobre coloração de grafos:
http://www.cs.umd.edu/class/fall2009/cmsc451/lectures/Lec23-sat.pdf

Satisfabilidade da NP-Completude em lógica booleana:
http://en.wikipedia.org/wiki/Boolean_satisfiability_problem#NP-completeness

Notas em NP-Completude
http://valis.cs.uiuc.edu/~sariel/teach/courses/473/notes/02_npc_notes.pdf

ARTIGO - Polynomial 3-SAT Encoding for K-Colorability of Graph
http://research.ijcaonline.org/encc/number1/encc004.pdf

GRAPH COLORING
http://www.ic.uff.br/elavio/mini2.pdf

Data Reduction for Graph Coloring Problems
http://arxiv.org/abs/1104.4229

NP Completeness - 02 - Additional Problems
http://valis.cs.uiuc.edu/~sariel/teach/courses/473/notes/02_npc_notes.pdf