Prova da NP-Completude 3SAT

Gabriel Manzoni Moreira e Rafael Thomazi Gonzalez

1. Introdução
2. Caracterização do Problema
3. Prova da NP-Completude
    - 3.1 Algoritmo de Verificação
    - 3.2 Redução de SAT para 3SAT
        - 3.2.1 Equivalência entre SAT e 3SAT
4. Conclusão

Introdução

- Problemas de decisão podem ser divididos em classes conforme suas características.
- Classe P – problemas que podem ser resolvidos em uma MTD em tempo polinomial
- Classe NP – problemas resolvidos em tempo polinomial por uma MTND
- Classe NP-Completo - possuem verificação e redução em tempo polinomial
- Classe NP-Difícil – possuem redução em tempo polinomial

Caracterização do Problema

- Problema da satisfazibilidade booleana
- 3SAT é um conjunto de conjunções de i cláusulas C na 3CNF
- Cada cláusula C é um conjunto de disjunção de exatamente 3 literais
- Exemplo de uma fórmula 3SAT:

(a V b V c) ^ (a V !b V c) ^ (a V b V !c) ^ (a V !b V !c)

Caracterização do Problema

- Problema da verificação
- Dada uma fórmula 3SAT e os valores verdades dos termos (certificado), saber se esses valores satisfazem a expressão.

Prova da NP-Completude

- Para provar que um problema é NP-Completo devemos:

1. Provar que ele pertence à classe dos problemas NP, ou seja, existe um algoritmo de verificação que excuta em tempo polinomial.
2. Provar que existe algum problema NP-Completo que pode ser reduzido ao problema, no caso o 3SAT

Algoritmo de Verificação

```
function verificaSatisf( f3SAT[n], certif )
{
    for( i = 0; i < n; i++) {
        if( certif[f3SAT[i].v1] == false &&
    certif[f3SAT[i].v2] == false &&
certif[f3SAT[i].v3] == false )
        {
            return false;
        }
    }
    return true;
}
```

Complexidade O(n)

Redução de SAT para 3SAT

- Será usada o problema SAT que, pelo Teorema de Cook, foi provado pertencer à classe dos problemas NP-Completos.
- O problema SAT é o mesmo princípio do 3SAT, com a diferença que cada cláusula do SAT pode conter um número variável de termos.
- Exemplo de uma expressão SAT:

(a) ^ (a V b) ^ (a V b V c) ^ (!a V !b V d V e V ! f)

Redução de SAT para 3SAT

- Se *Ci* = X_1, então coloca-se a seguinte expressão em *f3SAT*:

Ci' = $(X_1 \vee Y \vee Y_1) \wedge (X_1 \vee !Y \vee Y_1) \wedge (X_1 \vee Y \vee !Y_1) \wedge (X_1 \vee !Y \vee !Y_1)$

- Se *Ci* = $(X_1 \vee X_2)$, então coloca-se a seguinte expressão em *f3SAT*:

Ci' = $(X_1 \vee X_2 \vee Y) \wedge (X_1 \vee X_2 \vee !Y)$

- Se *Ci* = $(X_1 \vee X_2 \vee X_3)$, então copia-se *Ci* para *f3SAT*;

- Se *Ci* = $(X_1 \vee X_2 \vee \ldots \vee X_k)$ e $k > 3$, então coloca-se a seguinte expressão em *f3SAT*:

Ci' = $(X_1 \vee X_2 \vee Y) \wedge (!Y \vee X_3 \vee Y_1) \wedge (!Y_1 \vee X_4 \vee Y_2) \wedge \ldots \wedge$
$(!Y_{(k-4)} \vee X_{(k-2)} \vee Y_{(k-3)}) \wedge (!Y_{(k-3)} \vee X_{(k-1)} \vee X_k)$

Equivalência entre SAT e 3SAT

- Deve-se provar que *f3SAT* é satisfazível se e somente se *fSAT* é satisfazível

- Para cláusulas com um termo temos que X_1 deve possuir o mesmo valor verdade que $(X_1 \vee Y \vee Y_1) \wedge (X_1 \vee !Y \vee Y_1) \wedge (X_1 \vee Y \vee !Y_1) \wedge (X_1 \vee !Y \vee !\,Y_1)$.
- Se X_1 for Verdade, teremos um termo verdadeiro em todas as cláusulas, o que basta para tornara a expressão verdadeira.
- Se X_1 for Falso, podemos notar que existem todas as combinações de Y e Y_1 nas cláusulas, logo para qualquer combinação de valores verdade de Y e Y_1, sempre existirá uma cláusula que será falsa, tornando a expressão falsa.

Equivalência entre SAT e 3SAT

- Para cláusulas com dois elementos, podemos provar usando propriedades da lógica básica. Partiremos de (X_1 v X_2) e chegaremos em (X_1 v X_2 v Y) ^ (X_1 v X_2 v !Y).

1. (X_1 v X_2)
2. (X_1 v X_2 v Y) : por introdução da disjunção da linha 1
3. (X_1 v X_2 v !Y) : por introdução da disjunção da linha 1
4. (X_1 v X_2 v Y) ^ (X_1 v X_2 v !Y) : por união da linha 2 com a linha 3.

- Como partimos de (X_1 v X_2) e chegaremos em (X_1 v X_2 v Y) ^ (X_1 v X_2 v !Y), é provado então que as duas expressões possuem o mesmo valor verdade.

Equivalência entre SAT e 3SAT

- Para cláusulas com três elementos, basta copiar a cláusula, pois está já é uma cláusula 3SAT.

- Para cláusulas com 4 ou mais elementos devemos provar que:
- *f3SAT* é satisfazível → *fSAT* é satisfazível
- *fSAT* é satisfazível → *f3SAT* é satisfazível
- Provas por absurdo.

Algoritmo de Redução

```
function convertSATto3SAT( fSAT[n] )
{
    f3SAT = null;
    ind = 1;
    for (i = 0; i < n; i++)  {
        if( fSAT[i].size() == 1 ) {
           v = “x” + ind++;
           v1 = “x” + ind++;
           f3SAT.Add( new Clausula(fSAT[i][0], v, v1) );
           f3SAT.Add( new Clausula(fSAT[i][0], “!”+v, v1) );
           f3SAT.Add( new Clausula(fSAT[i][0], v, “!”+v1) );
           f3SAT.Add( new Clausula(fSAT[i][0], “!”+v, “!”v1) );
        } else if( fSAT[i].size() == 2 ) {
           v = "x" + ind++;
           f3SAT.Add( new Clausula(fSAT[i][0], fSAT[i][1], v) );
           f3SAT.Add( new Clausula(fSAT[i][0], fSAT[i][1], “!”+v) );
        } else if( fSAT[i].size() == 3 ) {
           f3SAT.Add(new Clausula(fSAT[i][0], fSAT[i][1], fSAT[i][2]));
        }
```

Algoritmo de Redução

```
        else {
           v = "x" + ind++;
           f3SAT.Add( new Clausula(fSAT[i][0], fSAT[i][1], v) );
           v1 = "x" + ind++;
           for( j = 2; j < fSAT[i].size()-2; j++) {
              f3SAT.Add( new Clausula("!"+v, fSAT[i][j], v1) );
              v = v1;
              v1 = "x" + ind++;
           }
           f3SAT.Add(new Clausula("!"+v, fSAT[i][fSAT[i].size()-2],
               fSAT[i][fSAT[i].size()-1]));
           ind--;
        }
    }
    return f3SAT;
}
```

Complexidade O(n*m)

Conclusão

- A importância do problema 3SAT reside no fato de este ser uma instância mais simétrica do problema SAT.
- Tendo em vista que suas cláusulas têm no máximo três variáveis, sendo mais fácil por isso provar propriedades sobre ele e reduzi-lo a outros problemas NP-Completos.

Perguntas?

Fim