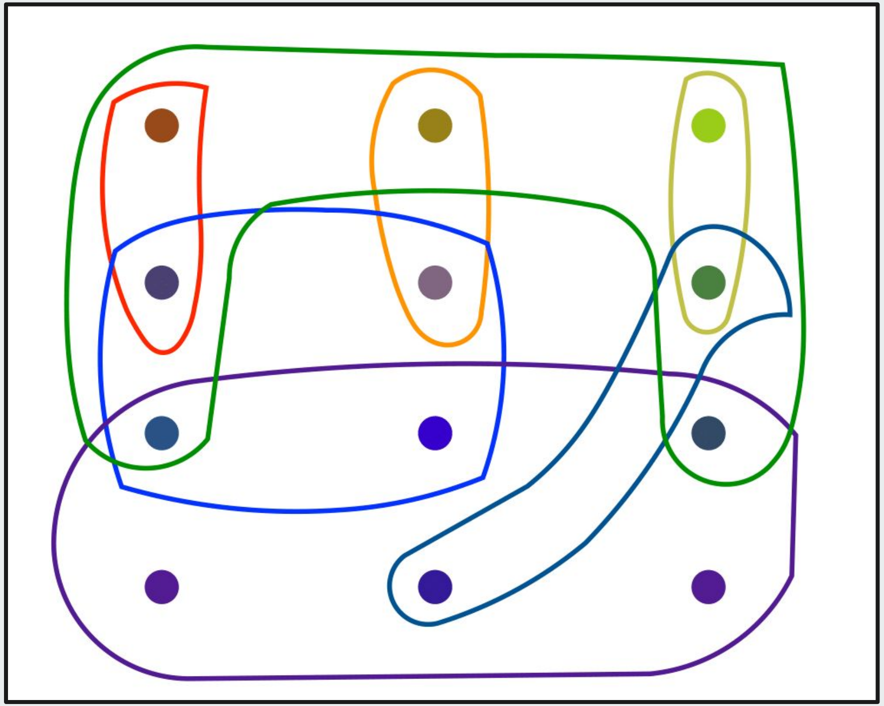
Cobertura de Conjuntos



Apresentação

Disciplina: Complexidade de Algoritmos

Período letivo: 2020/2 ERE

Prof: Mariana Luderitz Kolberg

Alunos: Roger Luis Sebastiany e Natanael da Silva Debona

Objetivo

Neste trabalho iremos utilizar conceitos de matemática discreta, teoria dos grafos e complexidade de algoritmos para caracterizar e analisar o Problema da Cobertura de Conjuntos e a sua NP-completude.

Considerações iniciais

Inicialmente, consideramos que as classes de complexidade P, NP e PSPACE são fechadas sob reduções em tempo polinomial.

Caracterização do problema

Dado um conjunto universo {1,2,...,n}, e um conjunto S com m subconjuntos {*s0*, *s1*,..., *sn*}, cuja união é igual ao conjunto universo, o **problema de cobertura de conjuntos** consiste em identificar o menor subconjunto de S cuja união é igual ao conjunto universo.

Problema de decisão

O problema de decisão possui como instância uma tupla (U, S, k), onde U e S são análogos à definição do slide anterior, e k é um número inteiro >= 0.

Instâncias onde o subconjunto $B \subseteq S$, tal que **B é cobertura de U** e **|B| <= k** serão instâncias positivas para o problema.

Instâncias onde o subconjunto B seja vazio, ou $B \subseteq S$, tal que b **não** é cobertura de U ou $B \subseteq S$, tal que b é cobertura de U e |B| **> k** serão instâncias negativas para o problema.

Cobertura de Conjuntos $\subseteq$ NP

Para demonstrar que o problema de cobertura de conjuntos pertence à classe de problemas NP, é necessário demonstrar um algoritmo de verificação com tempo polinomial.

Algoritmo usando método de força bruta

```
def brute_force(U, S):
    B = [0] * (len(S)+1) #O(S+1)
    C = set() #O(1)
    temp = len(U) #O(1)
    count = 0 #O(1)
    while C != U: #O(U)*O(S²)
        for s in S: #O(S)
            index = S.index(s)1 #O(1)
            if temp >= len(s.symmetric_difference(U))and B[index] == 0: #O(S²)
                temp =len(s.symmetric_difference(U))
        B[index] =1 #O(1)
        C = C.union(S[index])#O(S)
        S.pop(index) #O(1)
        count +=1 #O(1)
    return B
```

Análise de complexidade

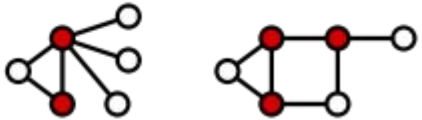
Seja |U| = n e |S| = m

$$O(U)*O(S^2) + O(S+1) => O(n*m^2)$$

Cobertura de Conjuntos ⊆ NP-Difícil

Para demonstrar que o problema de cobertura de conjuntos é NP-difícil, e consequentemente, NP-completo, iremos reduzir uma instância do problema de Cobertura de Vértices, sabidamente NP-Difícil.

O problema da cobertura mínima de vértices consiste em encontrar a menor coleção de vértices cujas arestas associadas cobrem um dado grafo G.



Redução

A redução ocorre em quatro etapas:

1) Receber uma instância válida para o problema de cobertura de vértices
2) Converter essa instância para o problema de cobertura de conjuntos
3) Rodar o algoritmo para obter uma saída
4) Converter a saída para uma instância do problema origem

Instânciamento

Uma instância do problema da Cobertura mínima de Vértices é um grafo G = (V,E) não orientado, onde V é o conjunto de vértices e E é o conjunto de arestas.

Para criar uma instância de grafo, definimos uma classe Graph que é responsável por criar um grafo, dados os vértices, e as arestas serão adicionadas pelo método addedge, que por sua vez recebe os dois vértices que compõem a aresta em questão.

Conversão da entrada

A partir de um grafo G, criamos o conjunto universo U e o conjunto de partes S.

```
V=set()
for u in dict(g.graph).keys():#O(u)*O(u*v) -> v*O(u²)
    for v in dict(g.graph)[u]:#O(u)
        V.add(u*10+v) #O(u*v)
E = []
max_u = g.V
def is_edge_of(v, u):
    return (str(u) in str(v))
for u in range(1, max_u+1):#O(u)*P(u+v)
    v_of_u = set(filter(lambda v: is_edge_of(v, u), V))#O(u+v)
    E.append(v_of_u)
```

Análise da complexidade

Seja u o número de vértices de G, e v o número de arestas,

$$v*O(u^2) + O(u^2) + O(u) \rightarrow v*O(u^2)$$

Interpretação da saída

Ao rodar o algoritmo set_cover.brute_force com as entradas U e S geradas a partir de um grafo G, a saída será o número mínimo de vértices necessários para cobrir o grafo G, e uma lista cujos índices marcados com 1 equivalem aos vértices utilizados para cobrir G.

Conclusão

Concluímos que o problema de cobertura de conjuntos pertence a NP-completo. Para análise, utilizamos um algoritmo de cobertura de conjuntos para solucionar uma instância do problema de cobertura de vértices, conhecidamente np-difícil, em tempo polinomial, através do método de redução.

Referências

[Set cover problem - Wikipedia]

[What is the set cover problem (mit.edu)]

[AM221_lecture21.pdf (harvard.edu)]

[fetch.php (ufrgs.br)]

[Algoritmos de aproximação - Problema de cobertura por conjuntos (usp.br)]

[Reduction (complexity) - Wikipedia]

https://wiki.python.org/moin/TimeComplexity