Trabalho 2 - SVP

17 de outubro de 2022

André Oliveira Barbosa - A91684 Francisco Antonio Borges Paulino - A91666

Caso de Estudo

O chamado problema do vetor curto (SVP) consiste no cálculo de um vetor de inteiros
$$e \in \{-1, 0, 1\}^m$$
não nulo que verifique a seguinte relação matricial
$$\forall\, i < n \cdot \sum_{j<m} e_j \times \mathsf{L}_{j,i} \;\equiv\; 0 \mod q$$

Para tal pertende-se resolver o SVP por programação inteira no qual teremos de respeitar certas condições.

Condições:

1. Os valores m, n, q são escolhidos com $,n > 6, |m| > 1 + |n|e|q| > |m|$.
2. O valor de q é primo e $q \geq 3$.
3. Os elementos $L_{j,i}$ são gerados aleatória e uniformemente no intervalo inteiro $\{-d \cdots d\}$ sendo $d \equiv (q-1)/2$.
4. Determinar em primeiro lugar, se existe um vetor e não nulo (pelo menos um dos e_j é diferente de zero).
5. Se existir e pretende-se calcular o vetor que minimiza o número de componentes não nulas.

Variáveis:

- $L_{j,i}$ - matriz que representa o reticulado
- M
- N
- q
- D
- E

Inicialização

Para a resolução deste exercício utilizamos a biblioteca [OR-Tools (https://developers.google.com/optimization)](https://developers.google.com/optimization) que criou uma interface para o SCIP. Esta biblioteca foi instalada com o commando pip install ortools.

In [88]:
```
!pip install ortools
```

```
Requirement already satisfied: ortools in d:\anaconda\envs\logica\lib\site-pack
ages (9.4.1874)
Requirement already satisfied: absl-py>=0.13 in d:\anaconda\envs\logica\lib\sit
e-packages (from ortools) (1.2.0)
Requirement already satisfied: numpy>=1.13.3 in d:\anaconda\envs\logica\lib\sit
e-packages (from ortools) (1.23.3)
Requirement already satisfied: protobuf>=3.19.4 in d:\anaconda\envs\logica\lib
\site-packages (from ortools) (4.21.6)
```

Implementação

Começamos por importar a biblioteca de programação linear do OR-Tools e criar uma instância do solver.

Depois inicializamos o solver ferramenta e definir os valores para as constantes m,n,q.

In [89]:
```
from ortools.linear_solver import pywraplp
from ortools.sat.python import cp_model
import random
import networkx as nx

solver = pywraplp.Solver.CreateSolver('SCIP')
```

Modelação das restrições e introdução do solver

Passamos agora à modelação das restrições e à sua introdução no solver. Para tal, iremos analizar as condições e subdividi-las de forma a facilitar a criação de uma expressão lógica, bem como a sua interpretação.

1. O valor de $n > 30$.

2. O valor de $|m| > 1 + |n|$.

Tendo em atenção as notas do exercicio, sabemos que para um dado $x \geq 0$, representa-se por $|x|$, o tamanho de x em bits: o menor l tal que $x < 2^{\ell}$

Após analisar a condição imposta, conseguimos perceber que o comprimento de m do número de bits será superiror ao comprimento de n em bits uma vez que $|m| > 1 + |n|$.

Para conseguir calcular o número de bits de m teremos de descobrir o número de bits de n primeiro. Assim, tendo em conta que $|m| > |n| + 1$, $m < 2^{n+2}$
Uma vez que m tem de ser inferior a 2^{n+2}, um possivel valor para m seria dado pela seguinte formula: $m = 2^{n+2} - 1$.

Para tal, será necessario a criação de uma função para calcular o valor de m.

In [97]:
```
def valorm(N):
    a = 0
    while N>0:
        a+=1
        N//=2
    m=2**(a+2)-1
    return m

#print(m)
```

3. O valor de $|q| > |m|$ e q é primo.

In [91]:
```
def primo(num):
    if num > 1:
        if num==2 or num==3:
            return True
        for i in range(2, num//2):
            if (num % i) == 0:
                return False
                break
            else:
                return True
    else:
        return False


#calcula q
def valorq(M):
    b = 0
    m= M

    #calcula |m|
    while m > 0:
        b += 1
        m//=2
    #como |q|> |m|, entao:
    b+=1
    #queremos encontrar um primo entre m e 2^b , para evitar casos em que o q sej
    #o intrevalo é entre m+1 e 2^b
    i = M+1
    T = 2**b
    while i < T:
        if primo(i) : return i
        else : i+=1
```

4. Os elementos $L_{j,i}$ são gerados aleatória e uniformemente no intervalo inteiro $\{-d \cdots d\}$ sendo $d \equiv (q-1)/2$

Definição da matriz de alocação

In [98]:
```
def lMatriz(matriz, N):

    M = valorm(N)
    Q = valorq(M)

    d = (Q-1)/2
    d = int(d)

    #criar a matriz
    for i in range(N):
        matriz[i]={}
        for j in range(M):
            matriz[i][j]={}

    #insere aleatoriamente elementos na matriz

    for x in range(N):
        for y in range(M):
            matriz[x][y] = random.randint(-d,d)

    #for n in range(N):
     #   print('\n')
      #  for m in range(M):
       #     print(matriz[n][m], end=' ')
```

Inputs e Outputs

In [99]:
```
N=3
matriz = {}

m = valorm(N)
q = valorq(m)

print("m=",m, "\nq=",q)

lMatriz(matriz, N)
```

```
m= 15
q= 17
```

Criação de um vetor e

Passamos a criação do vetor e de dimensões m e que será composto pelos valores $\{-1, 0, 1\}$

Para tar iremos recorrer ao uso da ferramento do $cp\ model$.

In [100]:
```
def vetorE(matriz,N,m,q):
    #print(N)
    #print(m)
    #print(q)
    model = cp_model.CpModel()
    #criação do vetor e no range de [-1,1]
    e={}
    for i in range(m):
        e[i] = model.NewIntVar(-1,1,f'e[{i}]')

    k={}
    for i in range(N):
        k[i] = model.NewIntVar(10000,-10000, f'k[{i}]')

    #verificar que e_i é diferente de 0
    model.Add(sum(e[i] for i in range(m)) >0)

    #verifica a segunda condiçao
    for i in range(N):
        model.Add(sum(e[j] * (matriz[i][j] ) for j in range(m)) == k[i]*q)

    #solver
    solver = cp_model.CpSolver()
    status = solver.Solve(model)
    if status == cp_model.OPTIMAL or status == cp_model.FEASIBLE:
        for i in range(m):
            print(solver.value(e[i]))
    else:
        print("Not found")
```

In [101]:
```
vetorE(matriz,N,m,q)
```

```
Not found
```

Verificação da condição inicial

Para finalizar o problema do vetor curto é necessario verificar a seguinte relação:

$$\forall\, i < n \cdot \sum_{j<m} e_j \times \mathsf{L}_{i,j} \;\equiv\; 0 \mod q$$

In [102]:
```
'''
for i in range(N):
    solver.Add(sum((e[j] * matriz[i][j]) for j in range(m)) == 0 % q)
'''
```

Out[102]:
```
'\nfor i in range(N):\n    solver.Add(sum((e[j] * matriz[i][j]) for j in range
(m)) == 0 % q)\n'
```