TP1-2

October 6, 2025

1 TP1 - Grupo 7

André Filipe Dourado Pinheiro - A108473

Tiago Silva Costa - A108657

1.1 Problema 3 - Shortest Vector Problem

Na criptografia pós-quântica os reticulados inteiros ("hard lattices") e os problemas a eles associados são uma componente essencial. Um reticulado inteiro pode ser definido por uma matriz $\mathsf{H} \in \mathbb{Z}^{m \times n}$ (com $m \geq 2\,n$) e por um inteiro primo $q \gg m$. Para além dos parâmetros n, m, q , o "input" do problema é definido pelos inteiros $\mathsf{H}_{j,i}$ gerados de forma aleatória, independente e uniforme num intervalo $\{0, ..., q-1\}$.

O chamado problema do vetor mais curto (SVP) consiste no cálculo de um vetor de inteiros $e \in \{0,1\}^m$ não nulo que verifique as seguintes relações

$$\exists\, j < m \;.\; e_j \neq 0 \qquad ,$$

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

Pretende-se determinar, em primeiro lugar, se existe uma solução e . Se e existir pretende-se determinar e que minimiza o número de componentes não nulas.

1.2 Modelação

Para resolver o problema, faremos uso da biblioteca de programação linear do **OR-Tools**, o **pywraplp**. Assim, começamos por importar a biblioteca e criar uma instância do `solver`, a que daremos o nome de *solver*.

[1]:
```
from ortools.linear_solver import pywraplp
solver = pywraplp.Solver.CreateSolver('SCIP')
import numpy
```

1.2.1 Variáveis

Começemos por inicializar as variáveis n, m, q arbitráriamente a título de exemplo.

Além disso, vamos definir as variáveis de decisão:

Sejam

1. $e_{j,t}$ variáveis binárias, organizadas em uma matriz $e \in \{0,1\}^{m\times 2}$, que representa uma matriz de alocação, em que

$$e_{j,t} = \begin{cases} 1, & t = 1, \\ 0, & t = 0. \end{cases}$$

   Com finalidade de representar os valores de cada componente do vetor e.

2. p_i variáveis inteiras reunidas num vetor $p \in \mathbb{Z}^n$. Esta variável irá permitir verificar se $\sum_{j<m} e_j \times \mathsf{H}_{j,i}$ é multiplo de q.

[2]:
```
n, m, q = 3,8,17
e, p = {}, {}
H = numpy.random.randint(0, q, (m, n))

for j in range(m):
    e[j,0] = solver.BoolVar('e[%i][0]' % j)
    e[j,1] = solver.BoolVar('e[%i][1]' % j)

for i in range(n):
    p[i] = solver.IntVar(0.0, solver.infinity(), 'p[%i]' % i)
```

1.2.2 Restrições

Formalizemos as restrições do problema

1. Existe $j < m$ tal que $e_j \neq 0$, isto é,

$$\sum_{j<m} e_{j,0} \leq m - 1$$

pois se exister pelo menos um $e_j = 1$, a soma dos $e_{j,0}$ nunca pode ser m pela restrição 2.

[3]:
```
for j in range(m):
    solver.Add(sum(e[j,0] for j in range(m)) <= m-1)
```

2. Unicidade de cada componente do vetor e, i.e,

$$\forall j < n, \quad e_{j,0} + e_{j,1} = 1$$

pois cada componente ou é 1, ou é 0.

[4]:
```
for j in range(m):
    solver.Add(e[j,0]+e[j,1]==1)
```

3. O produto do vetor e com H resulta num vetor cuja cada componente é divisível por q, ou seja

$$\forall\, i < n, \quad \sum_{j<m} e_{j,1} \times \mathsf{H}_{j,i} = q \cdot p_i$$

[5]:
```
for i in range(n):
    solver.Add(sum(e[j,1]*H[j][i] for j in range(m)) == p[i]*q)
```

1.3 Optimização

Caso a solução exista, devemos maximizar o número de componentes não nulas de e, ou seja,

$$\max \sum_{j<m} e_{j,1}$$

[6]:
```
solver.Maximize(sum(e[j,1] for j in range(m)))
```

1.4 Ilustração

Podemos calcular e representar a solução da seguinte forma:

[7]:
```
status = solver.Solve()
if status == pywraplp.Solver.OPTIMAL:
    print("Solution found!")
    print(H)

    print("   ", end=" ")
    for i in range(m):
        print(i,end="  ")
    print(" ")
    print("e | ", end="")
    for j in range(m):
        if e[j,1].solution_value():
            print(1, end="  ")
        else:
            print(0, end="  ")
    print(" ")
    print("p | ", end="")
    for i in range(n):
        print(int(p[i].solution_value()), end = "  ")


else:
    print(H)
    print("Not solvable")
```

```
[[13  0  3]
 [ 0  0 11]
 [ 1 12 15]
 [ 6 13 10]
 [ 6  8 14]
 [11 13  0]
 [14  1  7]
 [ 4 15 16]]
Not solvable
```

1.5 Modelação final

Podemos assim criar uma função chamada *solverSVP* que resolve o problema para dados n, m, q.

[8]:
```
def solverSVP(n,m,q):
    solver = pywraplp.Solver.CreateSolver('SCIP')
    # Definição das variáveis
    H = numpy.random.randint(0, q, (m, n))
    e, p = {}, {}

    for j in range(m):
        e[j,0] = solver.BoolVar('e[%i][0]' % j)
        e[j,1] = solver.BoolVar('e[%i][1]' % j)

    for i in range(n):
        p[i] = solver.IntVar(-solver.infinity(), solver.infinity(), 'p[%i]' % i)

    # Restrição 1:
    for j in range(m):
        solver.Add(sum(e[j,0] for j in range(m)) <= m-1)

    # Restrição 2:
    for j in range(m):
        solver.Add(e[j,0]+e[j,1]==1)

    # Restrição 3:
    for i in range(n):
        solver.Add(sum(e[j,1]*H[j][i] for j in range(m)) == p[i]*q)

    # Objetivo:
    solver.Maximize(sum(e[j,1] for j in range(m)))

    # Ilustração do resultado
    status = solver.Solve()
    if status == pywraplp.Solver.OPTIMAL:
        print("Solution found!")
        print(H)

        print("   ", end=" ")
        for i in range(m):
            print(i,end="  ")
        print(" ")
        print("e | ", end="")
        for j in range(m):
            if e[j,1].solution_value():
                print(1, end="  ")
            else:
                print(0, end="  ")
```

```
        print(" ")
        print("p | ", end="")
        for i in range(n):
            print(int(p[i].solution_value()), end = "  ")


    else:
        print(H)
        print("Not solvable")
```

1.6 Testes

Segue-se alguns testes para certos valores de m, n, q.

[9]:
```
# Teste 1
solverSVP(1, 7, 5)
```

```
Solution found!
[[0]
 [3]
 [4]
 [3]
 [2]
 [0]
 [3]]
    0  1  2  3  4  5  6
e | 1  1  1  1  1  1  1
p | 3
```

[181]:
```
# Teste 2
solverSVP(3, 8, 17)
```

```
Solution found!
[[10  7 11]
 [ 1  1  4]
 [16 10  4]
 [14 12 15]
 [11  1  8]
 [16  7  3]
 [ 5 15 12]
 [12  8 14]]
    0  1  2  3  4  5  6  7
e | 0  1  1  0  0  0  1  1
p | 2  2  2
```

[188]:
```
# Teste 3
solverSVP(8, 17, 29)
```

```
[[ 5 10  0 21 19 27  1  7]
 [17 23  9 23  2 13  1  8]
 [ 7 11 16 24 25 28 24 27]
 [ 9 11 14  9 10  5  0 26]
 [21 13 14  0 24 21 22 16]
 [ 1 27 23 10 20  9 15 28]
 [17 25  9 19 16 20 28 12]
 [26  7 22 19 17 12  8  6]
 [ 4 19 23 11 27 22 23 17]
 [ 5  5 20 22 19  1 10 17]
 [13 27 13 18 21 12 24  9]
 [13  8  1  7  0  8  1 18]
 [28 11 15 27 25 25 10 22]
 [ 8  0  7 28 10 24  2 27]
 [ 6 21 22 24  3 23 21  8]
 [ 1 20 25  9  2 23  0  1]
 [28 21  1 17  5  8 21  0]]
Not solvable
```

[12]:
```
# Teste 4
solverSVP(10, 20, 67)
```

```
[[52  4 43  0 34 20 61 24 17 41]
 [53 61  3 49 51 48 62 34 14  5]
 [39 45  5  2 51 31 52 64 64 64]
 [56 31 33 64 42 57 29 39 51 32]
 [31  8 48  9 10 29 54 57  3  5]
 [35 62  8  5 12 42 33 26 31 24]
 [ 0 28 60 51 66 49 44 47  9 66]
 [10  0  5 11 53 61  6 24  7 13]
 [31 32 31 20 59 10 19 64 32 32]
 [40 49 10 43 31 19 57 27 18 37]
 [26 48 16 35 43 25 63 37 10  8]
 [53 54 36 41 42 24 10 26 43 50]
 [36 44 43 41 22 21 13 56 22 13]
 [50 21 10 44  6 31 47 15 59 28]
 [61 62 13 66 43 23 17 25  7 34]
 [21 61 39 29 60 14 39 31 46 39]
 [53 24 55 32 62 17 60 50 33 17]
 [38 37 41 18 27 13 36 10 53 16]
 [53 28 53 28 43 27 61 43 33 11]
 [66 16 30  1 25 33 48  6 18 35]]
Not solvable
```