# Demonstração da NP-Completude do problema da soma dos subconjuntos (Subset-Sum)

**Davi Azarias do Vale Cabral**[1], **João Antonio Lassister Melo**[1], **João Antonio Siqueira Pascuini**[1], **Renan Catini Amaral**[1], **Thallysson Luis Teixeira Carvalho**[1], **Vinicius Ribeiro da Silva do Carmo**[1]

[1]Departamento de Ciências da Computação – Universidade Federal de Alfenas
Avenida Jovino Fernandes de Sales 2600 – CEP 37133840 - Alfenas – MG - Brasil

davi.cabral@sou.unifal-mg.edu.br,

joao.lassister@sou.unifal-mg.edu.br,

joaoantonio.pascuini@sou.unifal-mg.edu.br,

renan.amaral@sou.unifal-mg.edu.br,

thallysson.carvalho@sou.unifal-mg.edu.br,

viniciusribeiro.carmo@sou.unifal-mg.edu.br

***Resumo.*** *Este trabalho tem como objetivo demonstrar que o problema da soma dos subconjuntos é NP-Completo. A metodologia adotada é demonstrar que esse problema pertence à classe NP e, posteriormente, apresentar um algoritmo de redução do problema da 3-satisfabilidade booleana, que sabe-se ser NP-Completo, ao problema da soma dos subconjuntos. Uma vez demonstradas essas duas proposições, concluímos que o problema da soma dos subconjuntos é NP-Completo.*

## 1. Introdução

A classe NP consiste nos problemas que são verificáveis em tempo polinomial, isto é, dada uma solução a um problema em NP, podemos verificar se esta solução é ou não correta em tempo polinomial ao tamanho da entrada para o problema. [Cormen et al. 2012]

Um problema em NP é dito ser NP-Completo caso ele seja no mínimo tão difícil quanto qualquer outro problema em NP.

O presente trabalho busca demonstrar que o problema da soma dos subconjuntos, definido a seguir, pertence a NP e, posteriormente, apresentar um algoritmo que transforme em tempo polinomial qualquer instância do problema da 3-satisfabilidade booleana, que sabemos ser NP-Completo, em uma instância do problema da soma dos subconjuntos. Deste modo, concluímos que este problema é no mínimo tão difícil quanto aquele, fazendo-o assim parte da classe dos problemas NP-Completos.

## 2. Metodologia

### 2.1. Problema 1

Sejam $K = \{k_1, k_2, ..., k_n\} \subseteq \mathbb{Z}$ e $t \in \mathbb{Z}$. O problema da soma dos subconjuntos, enunciado como um problema de decisão, é o seguinte:

Dada uma instância de entrada $< K, t >$, espera-se que um algoritmo $A$ que resolve o problema da soma dos subconjuntos responda $true$ caso exista um subconjunto $L \subseteq K$ tal que a soma de todos os elementos de $L$ seja igual a $t$, e $false$ caso contrário.

Exemplo 1: Sejam $K_1 = \{2, 3, 5, 1, -1\}$ e $t_1 = 8$. Para a instância de entrada $< K_1, t_1 >$, o algoritmo $A$ responderá $true$, pois $L_1 = \{3, 5\} \subseteq K_1$ e a soma dos elementos de $L_1$ é $3 + 5 = 8$. Note que há outros subconjuntos de $K$ que satisfazem à mesma propriedade, como $\{2, 5, 1\}$ e $\{3, 5, 1, -1\}$.

Exemplo 2: Sejam $K_2 = \{6, 7, 9, -2\}$ e $t_2 = 25$. Para a instância de entrada $< K_2, t_2 >$, $A$ responderá $false$, pois não existe nenhum subconjunto de $K$ cuja soma dos elementos seja igual a 25.

### 2.2. Problema 2

Uma expressão lógica $\phi$ é dita estar em 3CNF caso ela seja da forma $\phi = C_1 \wedge C_2 \wedge ... \wedge C_n$, para $i \in \{1, 2, ..., n\}$ e $C_i$ é uma disjunção de 3 ou menos literais. Definimos um literal como uma variável booleana ou sua negação, de forma que $x$ e $\neg x$ são ambos literais, caso $x \in \{true, false\}$.

A expressão $\phi$ é dita ser satisfatível caso ela possa ser avaliada como $true$ para alguma atribuição de valores a suas variáveis e não satisfatível caso contrário.

O problema da 3-satisfabilidade booleana pode ser definido como: dada uma expressão $\phi$ em 3CNF, um algoritmo $B$ diz se $\phi$ é satisfatível ou não.

Exemplo: Seja $\phi_1 = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$. Para a instância de entrada $\phi_1$, o algoritmo $B$ dirá que $\phi_1$ é satisfatível, pois $\phi_1 = true$ para $(x_1, x_2, x_3) = (true, false, false)$.

### 2.3. Prova de NP

Definimos o seguinte algoritmo como sendo o verificador de que o conjunto $L$ é um subconjunto de $K$ e a soma de seus elementos é igual a $t$:

```
algoritmo verifica(L,K,t)

    soma <- 0
    ehSubconjunto <- true

    para elem em L

        soma <- soma + elem
        ehSubconjunto <- ehSubconjunto && pertence(elem,K)

    fim para

    retorna soma == t && ehSubconjunto

fim algoritmo
```

O algoritmo acima verifica se a solução $L$ para a instância $< K, t >$ é verdadeira com uma complexidade de tempo $O(|L| * |K|)$, em que $|L|$ é o tamanho do conjunto $L$ e $|K|$ é o tamanho do conjunto $K$. Assim, demonstramos que o problema da soma dos subconjuntos pertence a NP.

### 2.4. Prova de NP-Completude

Seja $\phi$ uma expressão booleana $C_1 \wedge C_2 \wedge ... \wedge C_m$ em 3CNF, $m$ a quantidade de cláusulas de $\phi$ e $n$ a quantidade de variáveis em $\phi$.

Contruiremos uma instância $< K, t >$ do problema da soma de subconjuntos tal que, se a resposta do algoritmo $A$ para essa instância for $true$, então $\phi$ é satisfatível, e caso a resposta seja $false$, então $\phi$ não é satisfatível.

Criaremos dois números inteiros $y_i$ e $z_i$ para cada variável $x_i$ e dois números inteiros $g_j$ e $h_j$ para cada cláusula $C_j$ na expressão lógica para serem elementos do conjunto $K$. A seguinte tabela nos auxiliará a compreender como os números serão criados:

**Tabela 1. Redução de $\phi$ a $< K, t >$**

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | ... | $x_n$ | $C_1$ | $C_2$ | ... | $C_m$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $y_1$ | 1 | 0 | 0 | 0 | ... | 0 | | | ... | |
| $z_1$ | 1 | 0 | 0 | 0 | ... | 0 | | | ... | |
| $y_2$ | 0 | 1 | 0 | 0 | ... | 0 | | | ... | |
| $z_2$ | 0 | 1 | 0 | 0 | ... | 0 | | | ... | |
| $y_3$ | 0 | 0 | 1 | 0 | ... | 0 | | | ... | |
| $z_3$ | 0 | 0 | 1 | 0 | ... | 0 | | | ... | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| $y_n$ | 0 | 0 | 0 | 0 | ... | 1 | | | ... | |
| $z_n$ | 0 | 0 | 0 | 0 | ... | 1 | | | ... | |
| $g_1$ | 0 | 0 | 0 | 0 | ... | 0 | 1 | 0 | ... | 0 |
| $h_1$ | 0 | 0 | 0 | 0 | ... | 0 | 2 | 0 | ... | 0 |
| $g_2$ | 0 | 0 | 0 | 0 | ... | 0 | 0 | 1 | ... | 0 |
| $h_2$ | 0 | 0 | 0 | 0 | ... | 0 | 0 | 2 | ... | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| $g_m$ | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | ... | 1 |
| $h_m$ | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | ... | 2 |
| t | 1 | 1 | 1 | 1 | ... | 1 | 4 | 4 | ... | 4 |

**Primeiro quadrante:** As posições $[y_i, x_i]$ e $[z_i, x_i]$ da tabela são sempre 1, as demais são sempre 0.

**Segundo quadrante:** A posição $[y_i, C_j]$ da tabela será 1 caso a variável $x_i$ apareça sem negação na cláusula $C_j$ e 0 caso contrário. A posição $[z_i, C_j]$, por sua vez, será 1 caso a variável $x_i$ apareça com negação na cláusula $C_j$ e 0 caso contrário.

**Terceiro quadrante:** Todas as posições são 0.

**Quarto quadrante:** As posições $[g_j, C_j]$ da tabela são sempre 1, as posições $[h_j, C_j]$ são sempre 2, e as demais são sempre 0.

Quanto ao número $t$, este consistirá em uma sequência de $n$ 1's seguida por $m$ 4's.

Para efeitos de ilustração, iremos aplicar o algoritmo apresentado acima na expressão $\phi_1 = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$, transformando-a em uma instância do problema da soma de subconjuntos:

**Tabela 2. Redução de $\phi_1$ a $< K_1, t_1 >$**

| | $x_1$ | $x_2$ | $x_3$ | $C_1$ | $C_2$ | $C_3$ |
|---|---|---|---|---|---|---|
| $y_1$ | 1 | 0 | 0 | 1 | 0 | 1 |
| $z_1$ | 1 | 0 | 0 | 0 | 1 | 0 |
| $y_2$ | 0 | 1 | 0 | 0 | 0 | 1 |
| $z_2$ | 0 | 1 | 0 | 1 | 1 | 0 |
| $y_3$ | 0 | 0 | 1 | 0 | 1 | 1 |
| $z_3$ | 0 | 0 | 1 | 1 | 0 | 0 |
| $g_1$ | 0 | 0 | 0 | 1 | 0 | 0 |
| $h_1$ | 0 | 0 | 0 | 2 | 0 | 0 |
| $g_2$ | 0 | 0 | 0 | 0 | 1 | 0 |
| $h_2$ | 0 | 0 | 0 | 0 | 2 | 0 |
| $g_3$ | 0 | 0 | 0 | 0 | 0 | 1 |
| $h_3$ | 0 | 0 | 0 | 0 | 0 | 2 |
| $t_1$ | 1 | 1 | 1 | 4 | 4 | 4 |

A partir da tabela acima, somos capazes de construir a instância $< K_1, t_1 >$ do problema da soma de subconjuntos, tal que $K_1 = \{100101, 100010, 10001, 10110, 1011, 1100, 100, 200, 20, 10, 2, 1\}$ e $t_1 = 111444$.

Ao analisar essa instância, o algoritmo $A$ responde $true$, pois ele é capaz de encontrar, por exemplo, $L_1 = \{100101, 10110, 1100, 100, 20, 10, 2, 1\} \subseteq K_1$, cuja soma dos elementos é igual a $111444$.

Resta-nos, primeiramente, demonstrar que, se uma expressão $\phi$ qualquer for satisfatível, então a intância $< K, t >$ da soma de subconjuntos obtida a partir da redução de $\phi$ tem solução $L \subseteq K$:

**Passo 1:** Se $\phi$ é satisfatível, então existe um valor $true$ ou $false$ atribuído a cada variável $x_i$ de $\phi$ que torna a expressão verdadeira. Assim, para cada $x_i$:

- Caso $x_i = true$, inclua $y_i$ em $L$.
- Caso $x_i = false$, inclua $z_i$ em $L$.

**Passo 2:** Para cada cláusula $C_j$, como $\phi$ é satisfatível, então há pelo menos um literal em $C_j$ com valor $true$. Assim, para cada $C_j$:

- Caso haja um único literal $true$ em $C_j$, inclua $g_j$ e $h_j$ em $L$.
- Caso haja dois literais $true$ em $C_j$, inclua $h_j$ em $L$.
- Caso haja três literais $true$ em $C_j$, inclua $g_j$ em $L$.

**Passo 3:** Verifique que a soma dos elementos de $L$ é igual a $t$, pois:

- As primeiras $n$ posições do resultado da soma serão todas o valor 1, pois, como $x_i$ é $true$ ou $false$, exatamente um número $y_i$ ou $z_i$ foi incluído.
- As $m$ últimas posições do resultado serão todas 4, pois cada literal verdadeiro contribui com 1, enquanto os números nos quadrantes inferiores da tabela foram selecionados de acordo com a necessidade para alcançar o valor 4.

Por fim, demonstraremos que, se há solução $L$ para uma instância $< K, t >$ obtida pela redução de $\phi$, então $\phi$ é satisfatível.

Como a soma dos elementos de $L$ é igual a $t$, então as $n$ primeiras posições da soma dos elementos de $L$ são iguais a 1. Ora, isso quer dizer que, para cada $l_i \in L$, há exatamente um $k_i \in K$ tal que ou $k_i = y_i$, ou $k_i = z_i$. Pois se ambos $y_i$ e $z_i$ estivessem em $L$, teríamos o número 2 na posição $i$, e teríamos o número 0 caso nenhum dos dois estivesse presente. Isso define uma atribuição:

$$x_i = \begin{cases} true, & \text{se } y_i \in L \\ false, & \text{se } z_i \in L \end{cases} \tag{1}$$

Como as últimas $m$ posições da soma dos elementos de $L$ são iguais a 4, sabemos que, para cada cláusula $C_j$ em $\phi$, pelo menos um de seus literais está em $L$ e a satisfaz, caso contrário a posição correspondente a $C_j$ não alcançaria o valor 4, pois a soma dos números nos quadrantes inferiores não é o suficiente para chegar a esse valor.

Uma vez que atribuímos um valor para cada $x_i \in \{true, false\}$ em $\phi$ e esses valores são suficientes para satisfazer a todas as cláusulas da expressão, temos que $\phi$ é satisfatível.

## 3. Resultados

Durante este trabalho, conseguimos demonstrar que há um algoritmo determinístico que verifica uma solução para o problema da soma de subconjuntos em tempo polinomial, o que faz deste um problema em NP.

Além disso, demonstramos também que toda instância do problema da 3-satisfabilidade booleana, sabidamente NP-completo, pode ser reduzida a uma instância do problema da soma de subconjuntos, e, consequentemente, caso um algoritmo $A$ seja capaz de solucionar o problema da soma dos subconjuntos, por meio de uma redução ele também será capaz de solucionar qualquer instância do problema da 3-satisfabilidade booleana. Isso faz do problema da soma dos subconjuntos um problema NP-Difícil. Isto é, ele é no mínimo tão difícil quanto qualquer problema em NP.

## 4. Conclusões

A partir desses resultados, concluímos que o problema da soma dos subconjuntos (Subset-sum) é NP-Completo.

## Referências

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2012). *Algoritmos - Teoria e Prática*. GEN LTC, 3rd edition.