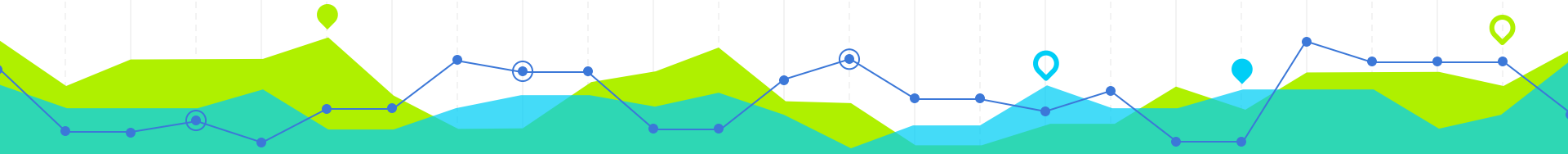


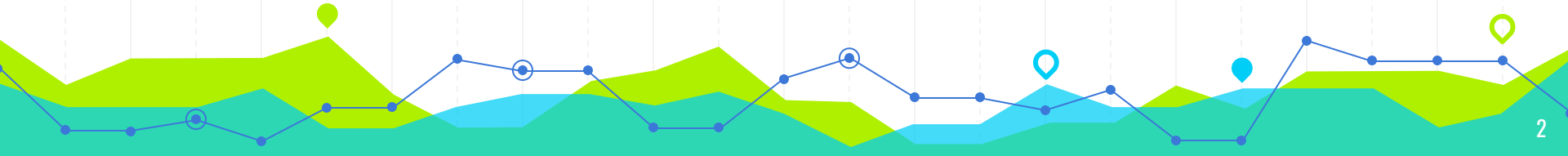
EMPARELHAMENTO DIVERSIFICADO SIMULATED ANNEALING

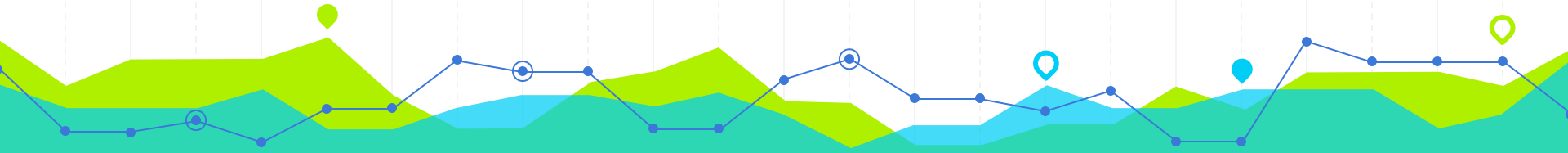
OLÁ!

Integrantes:

Guilherme Dytz dos Santos - 00244475

Lucas Spagnolo Bombana - 00314879



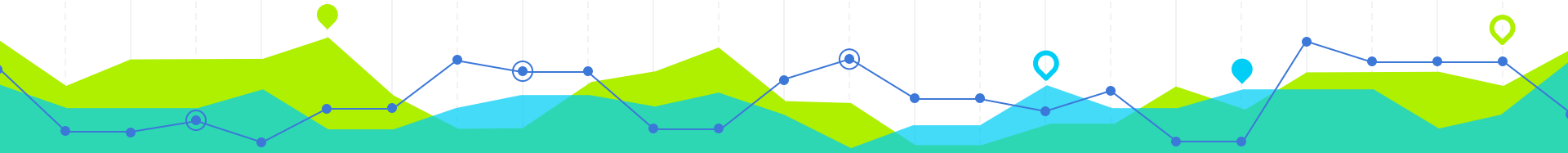


Definição do problema

1

DEFINIÇÃO DO PROBLEMA

Dado um grafo não-direcionado $G = (V, A)$, onde cada aresta $a \in A$ possui um tipo t_a, deseja-se encontrar um emparelhamento $M \subseteq A$ tal que todos $m \in M$ possuem tipos diferentes e que contenha o maior número de arestas possíveis.



Formulação do programa inteiro

2

FORMULAÇÃO DO PROGRAMA INTEIRO

Variáveis de Decisão:

$x_a \in \mathbb{B} \rightarrow$ 1 caso a aresta a seja selecionada, 0 caso contrário. $\forall a \in A$

FORMULAÇÃO DO PROGRAMA INTEIRO

Função Objetivo:

$$\textbf{maximiza} \sum_{a \in A} x_a$$

FORMULAÇÃO DO PROBLEMA INTEIRO

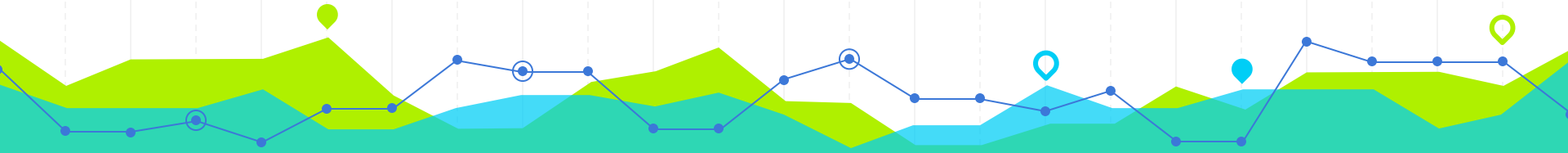
Restrições:

Para garantir o emparelhamento:

$$\sum_{a=(u,v)\in E_u} x_a \leq 1 \qquad \forall u \in V.$$

Para garantir os tipos diferentes de arestas:

$$\sum_{a\in E_t} x_a \leq 1 \qquad \forall t \in T.$$



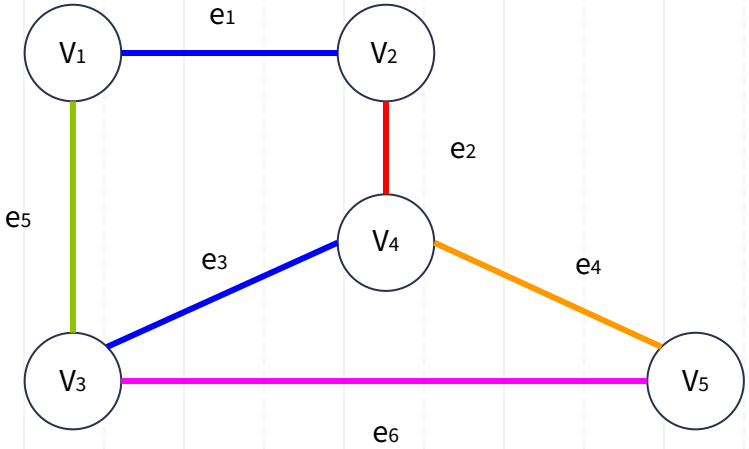
Solução inicial 3

SOLUÇÃO INICIAL

A solução inicial é construída através de um algoritmo guloso no menor grau médio dos vértices em que dada aresta incide. O desempate é feito de forma aleatória (depende da forma como o conjunto é ordenado).

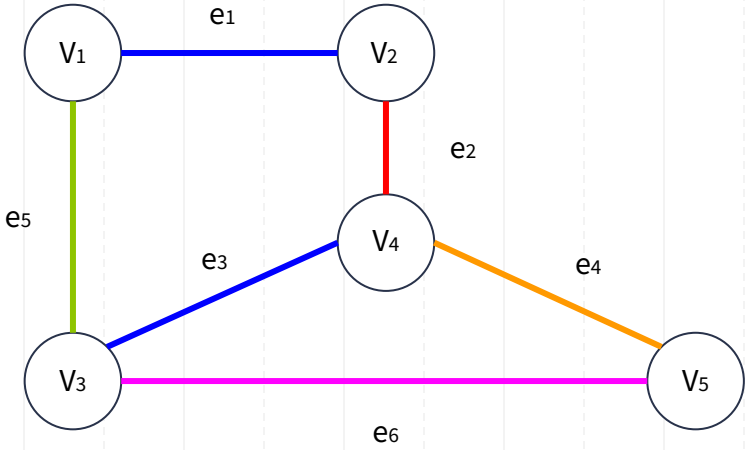
O exemplo dado a seguir é apenas uma forma visual de observar o algoritmo criado, ou seja, ele não reflete totalmente seu comportamento.

SOLUÇÃO INICIAL



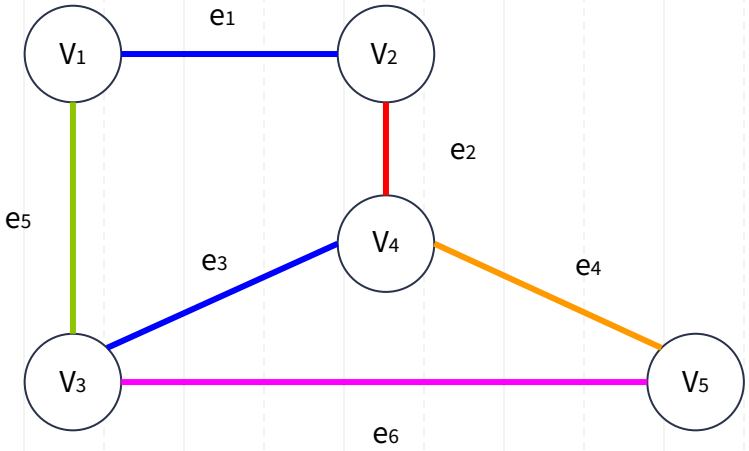
Vértice	Grau
V_1	2

SOLUÇÃO INICIAL



Vértice	Grau
V_1	2
V_2	2
V_3	3
V_4	3
V_5	2

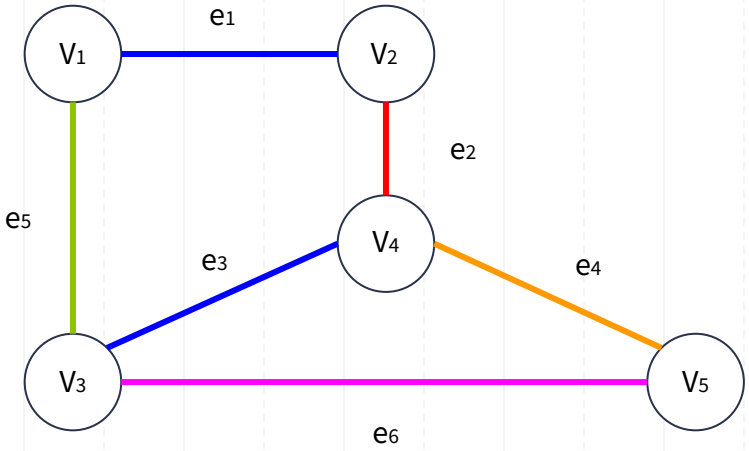
SOLUÇÃO INICIAL



Vértice	Grau
V_1	2
V_2	2
V_3	3
V_4	3
V_5	2

Aresta	Grau
e_1	2

SOLUÇÃO INICIAL



Vértice	Grau
V_1	2
V_2	2
V_3	3
V_4	3
V_5	2

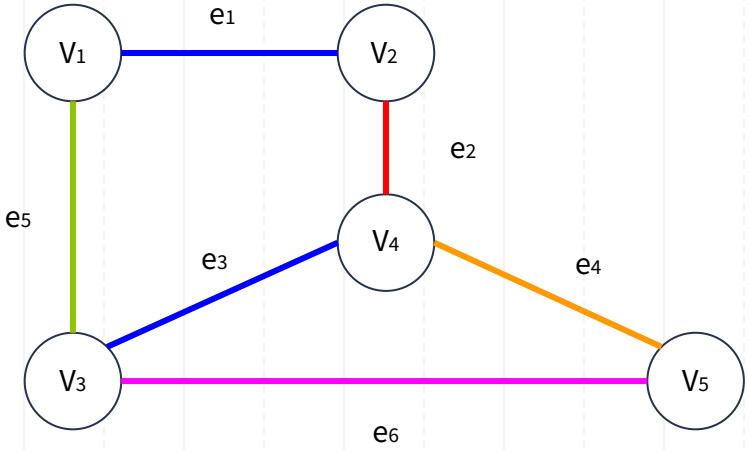
ORDENA

Aresta	Grau
e_1	2
e_2	2.5
e_3	3
e_4	2.5
e_5	2.5
e_6	2.5

Solução inicial = {}

Arestas ordenadas = { e_1, e_2, e_4, e_5, e_6, e_3 }

SOLUÇÃO INICIAL



Vértice	Grau
V1	2
V2	2
V3	3
V4	3
V5	2

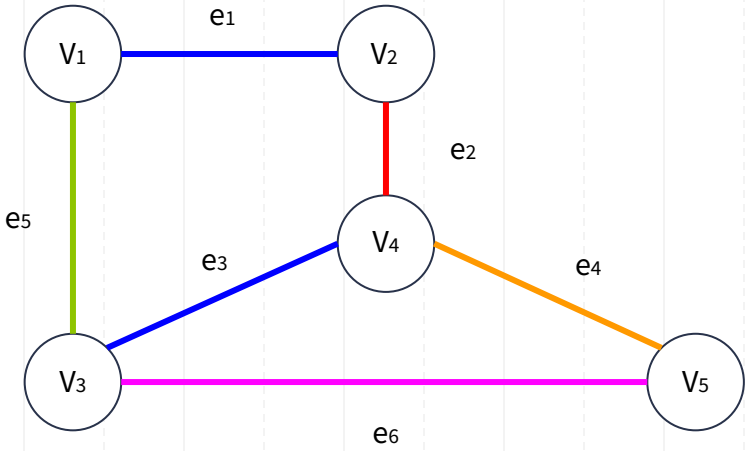
ORDENA

Aresta	Grau
e1	2
e2	2.5
e3	3
e4	2.5
e5	2.5
e6	2.5

Solução inicial = {}

Arestas ordenadas = { e1, e2, e4, e5, e6, e3 }

SOLUÇÃO INICIAL



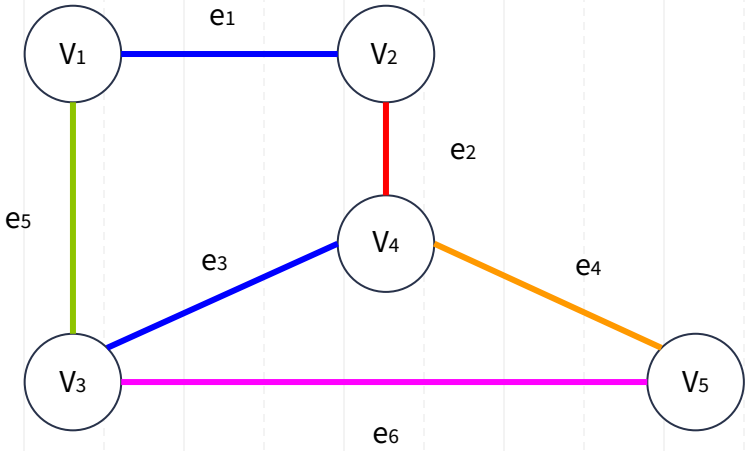
Vértice	Grau
V_1	2
V_2	2
V_3	3
V_4	3
V_5	2

Aresta	Grau
e_1	2
e_2	2.5
e_3	3
e_4	2.5
e_5	2.5
e_6	2.5

Solução inicial = { e_1 }

Arestas ordenadas = { e_2, e_4, e_5, e_6, e_3 }

SOLUÇÃO INICIAL



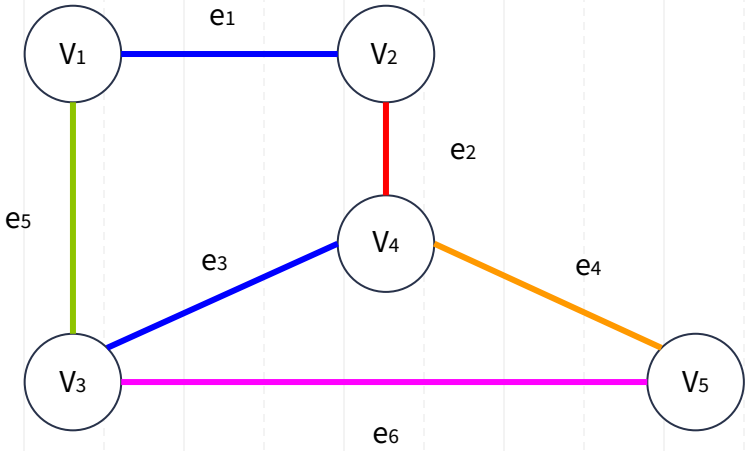
Vértice	Grau
V_1	2
V_2	2
V_3	3
V_4	3
V_5	2

Aresta	Grau
e_1	2
e_2	2.5
e_3	3
e_4	2.5
e_5	2.5
e_6	2.5

Solução inicial = { e_1 }

Arestas ordenadas = { e_4, e_5, e_6, e_3 }

SOLUÇÃO INICIAL

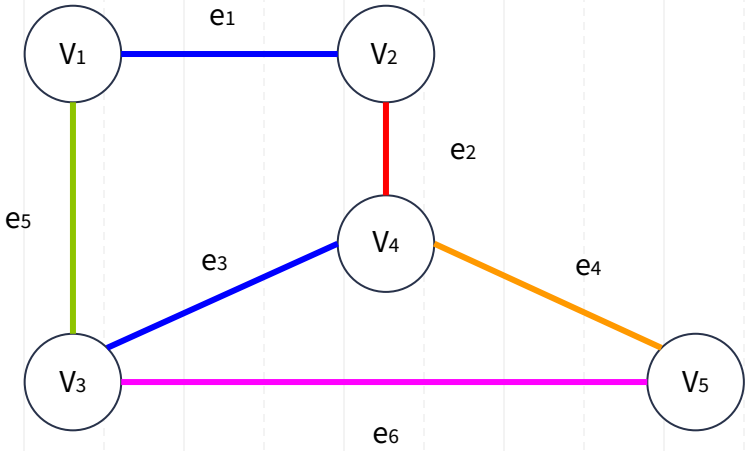


Solução inicial = { e_1 , e_4 }
Arestas ordenadas = { e_5, e_6, e_3 }

Vértice	Grau
V_1	2
V_2	2
V_3	3
V_4	3
V_5	2

Aresta	Grau
e_1	2
e_2	2.5
e_3	3
e_4	2.5
e_5	2.5
e_6	2.5

SOLUÇÃO INICIAL

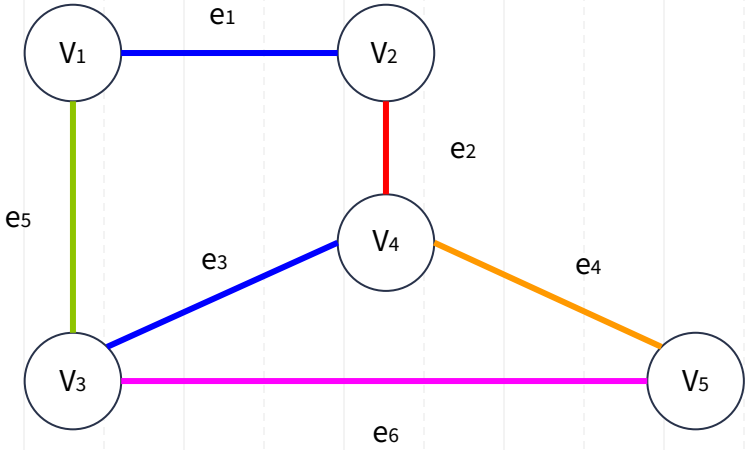


Solução inicial = { e_1 , e_4 }
Arestas ordenadas = { e_6, e_3 }

Vértice	Grau
V_1	2
V_2	2
V_3	3
V_4	3
V_5	2

Aresta	Grau
e_1	2
e_2	2.5
e_3	3
e_4	2.5
e_5	2.5
e_6	2.5

SOLUÇÃO INICIAL

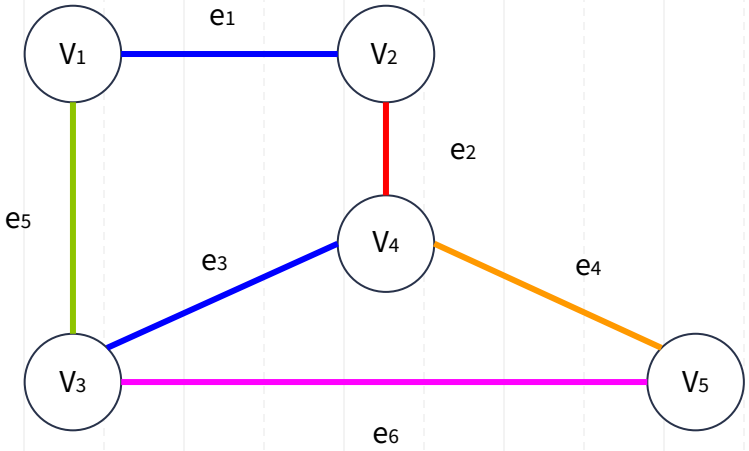


Vértice	Grau
V_1	2
V_2	2
V_3	3
V_4	3
V_5	2

Aresta	Grau
e_1	2
e_2	2.5
e_3	3
e_4	2.5
e_5	2.5
e_6	2.5

Solução inicial = { e_1 , e_4 }
Arestas ordenadas = { e_3 }

SOLUÇÃO INICIAL

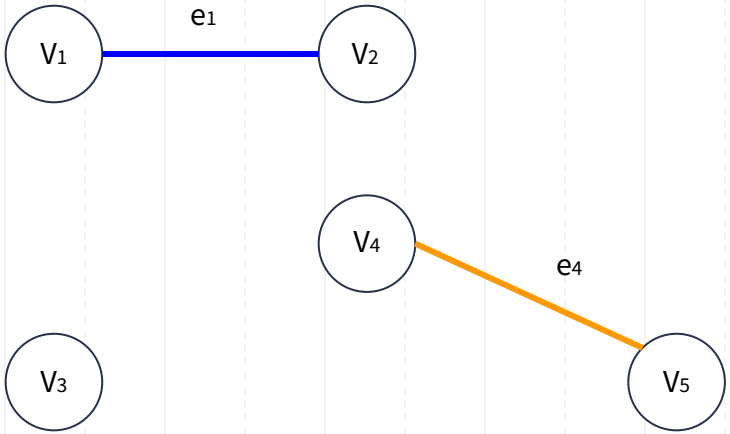


Solução inicial = { e1 , e4 }
Arestas ordenadas = { }

Vértice	Grau
V1	2
V2	2
V3	3
V4	3
V5	2

Aresta	Grau
e1	2
e2	2.5
e3	3
e4	2.5
e5	2.5
e6	2.5

SOLUÇÃO INICIAL



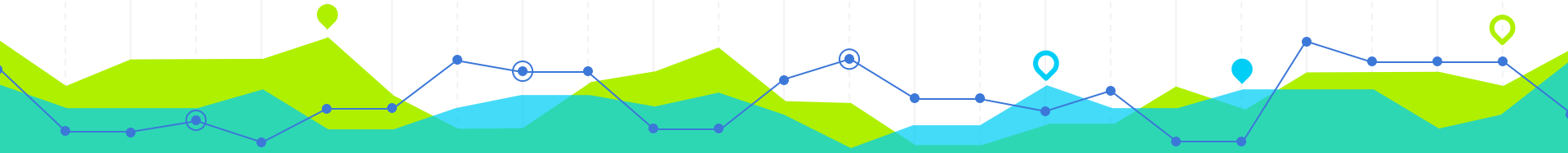
Vértice	Grau
V_1	2
V_2	2
V_3	3
V_4	3
V_5	2

Aresta	Grau
e_1	2
e_2	2.5
e_3	3
e_4	2.5
e_5	2.5
e_6	2.5

Solução inicial = { e_1 , e_4 }
Arestas ordenadas = { }

ALGORITMO

```
1: function GREEDY_INITIAL_SOLUTION(edges, deggre_counter)
2:     for e in edges do
3:         e.deggre ← (deggre_counter[e.vertex_u] + deggre_counter[e.vertex_v])/2
4:     end for
5:     sorted_edges ← sort_by_deggre(edges)
6:     edges' ← {}
7:     if |sorted_edges| > 0 then
8:         edges' ← edges' + {sorted_edges[0]}
9:         for edge in sorted_edges do
10:            for e in edges' do
11:                can_increase ← share_attributes(edge, e)
12:                if ¬can_increase then
13:                    edges' ← edges' + {edge}
14:                end if
15:            end for
16:        end for
17:    end if
18:    return edges'
19: end function
```



Vizinhança 4

VIZINHANÇA

Seja ***S*** um conjunto de arestas que configura uma solução para o problema. Sorteia-se uma aresta $a \in A$.

1. Caso $a \in S$.

2. Caso $a \notin S$.

VIZINHANÇA

Caso $a \in S$:

$$S' = S - \{a\}$$

Retorna S'

VIZINHANÇA

Caso $a \notin S$:

Caso a satisfaça as restrições do problema:

$$S' = S + \{a\}$$

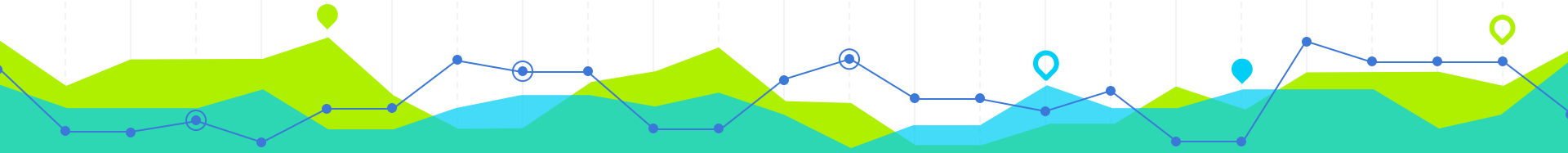
Retorna S'

Caso a não satisfaça as restrições do problema:

Repete-se o processo de sorteio.

VIZINHANÇA

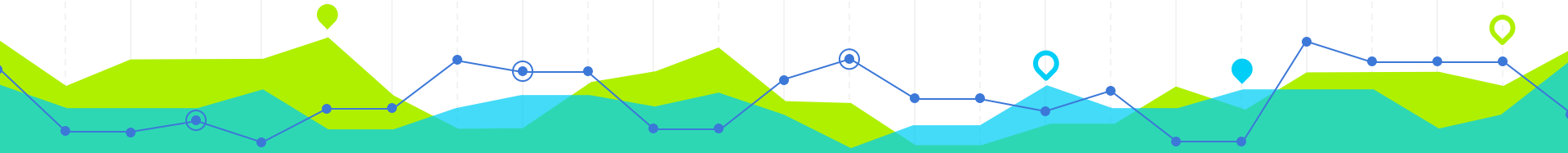
```
1:  function GET_NEIGHBOR(S, E)
2:      S' ← S                                          ▷ Caso a solução não aumente ou diminua
3:      for i = 0; i < |E|; i ← i + 1 do
4:          e ← random_edge(E)
5:          if e ∈ S then
6:              S' ← S − {e}
7:              break
8:          else
9:              can_increase ← true
10:             for e' in S do
11:                 can_increase ← ¬share_attributes(e', e)
12:                 if ¬can_increase then
13:                     break
14:                 end if
15:             end for
16:             if can_increase then
17:                 S' ← S + {e}
18:                 break
19:             end if
20:         end if
21:     end for
22:     return S'
23: end function
```



Parâmetros 5

Parâmetros do Algoritmo

- Iterações do Metrópolis
- Temperatura inicial
- Temperatura final
- Fator de desconto



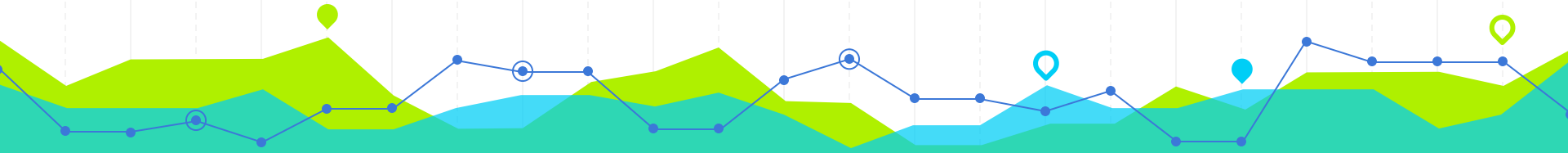
Implementação 6

Implementação

- **Otimizador:** GLPK utilizando Julia
- **Linguagem para o Simulated Annealing:** Python 3.8
- **Arestas:** Classe Edge
- **Grafo:** Lista de *edges*
- **Solução:** Python set

Implementação

- **Limite para o otimizador:** 30 minutos
- **Iterações do Metrópolis:** 500
- **Temperatura inicial:** 2
- **Temperatura final:** 0.01
- **Fator de desconto:** 0.99
- **Repetições do Simulated Annealing:** 10



Resultados

7

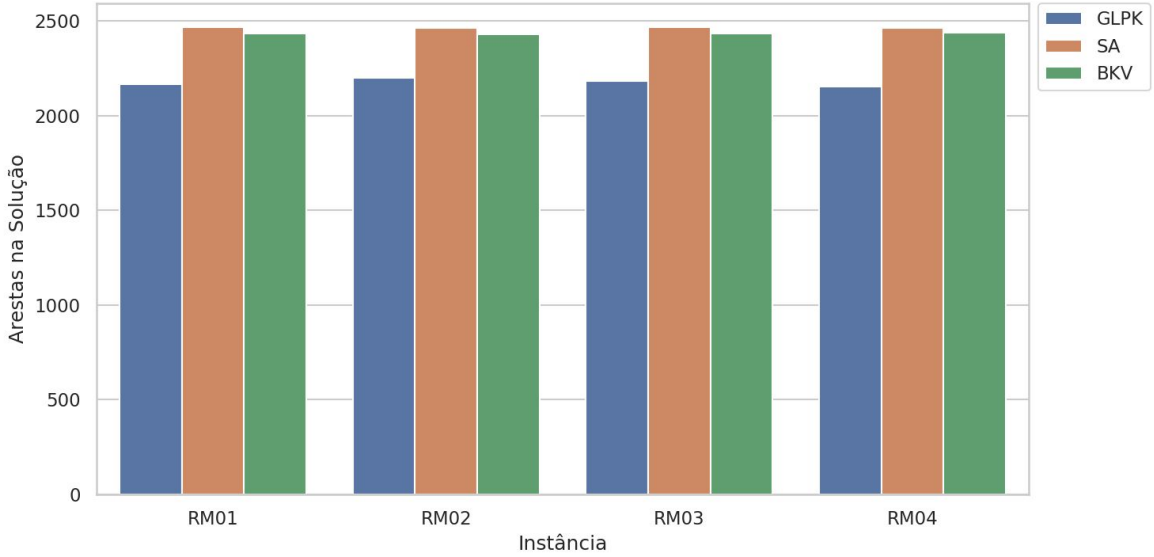
Resultados

Instância	GLPK	Simulated Annealing	BKV
RM01	2165	2467	2436
RM02	2202	2463	2432
RM03	2183	2470	2436
RM04	2155	2464	2439
RM05	2188	2465	2436
RM06	2153	2463	2431

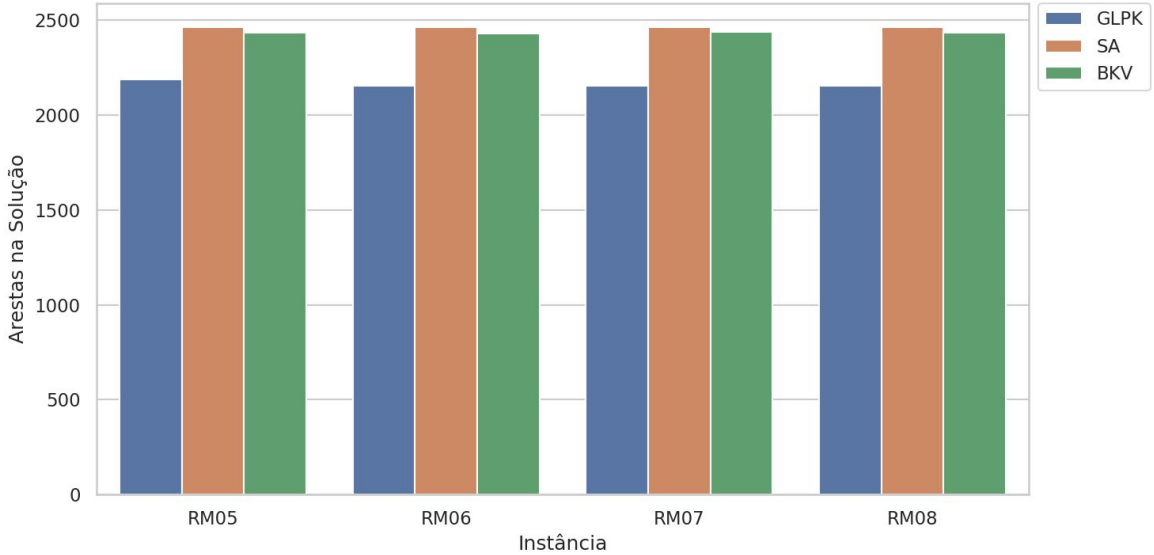
Resultados

Instância	GLPK	Simulated Annealing	BKV
RM07	2157	2465	2437
RM08	2156	2466	2433
RM09	2197	2466	2436
RM10	2184	2464	2429
RM11	2134	2465	2434
RM12	2188	2468	2440

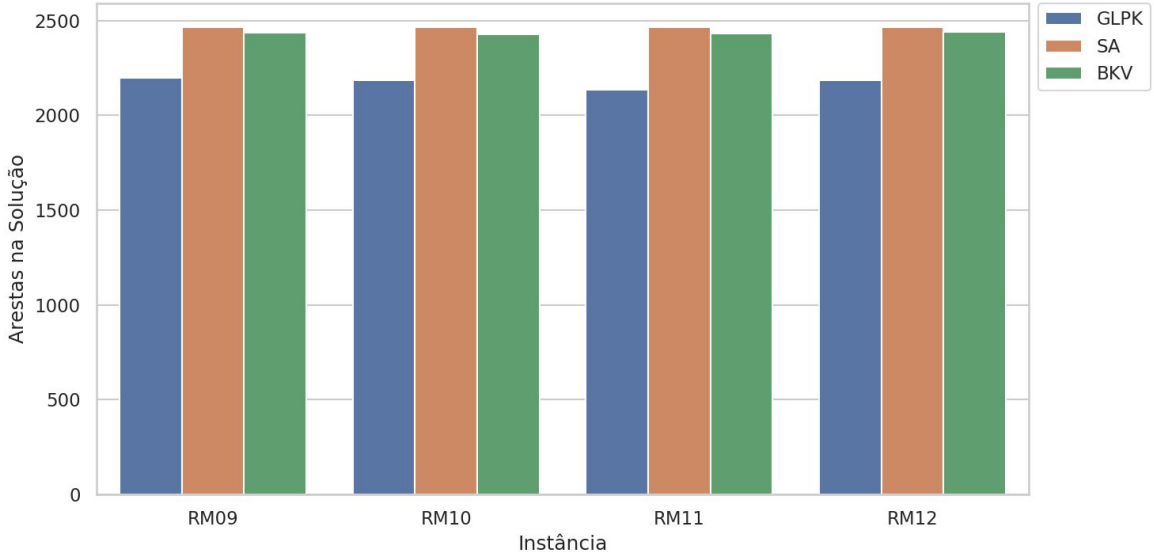
Resultados


2500
2000
1500
1000
500
0
Arestas na Solução
RM01
RM02
RM03
RM04
Instância
GLPK
SA
BKV

Resultados

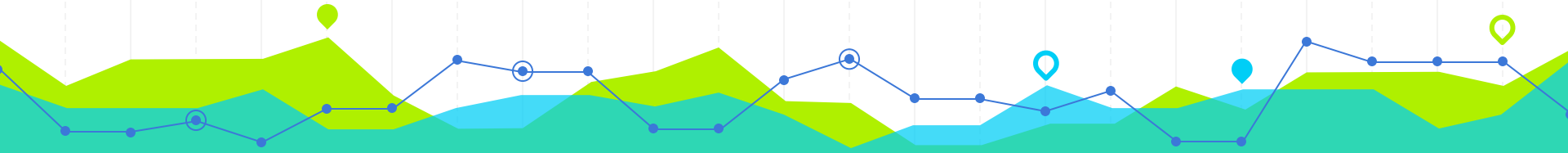

2500
2000
1500
1000
500
0
Arestas na Solução
RM05
RM06
RM07
RM08
Instância
GLPK
SA
BKV

Resultados


2500
2000
1500
1000
500
0
Arestas na Solução
RM09
RM10
RM11
RM12
Instância
GLPK
SA
BKV

VIZINHANÇA

```
1:  function GET_NEIGHBOR(S, E)
2:      S' ← S                                   ▷ Caso a solução não aumente ou diminua
3:      for i = 0; i < |E|; i ← i + 1 do
4:          e ← random_edge(E)
5:          if e ∈ S then
6:              S' ← S − {e}
7:              break
8:          else
9:              can_increase ← true
10:             for e' in S do
11:                 can_increase ← ¬share_attributes(e', e)
12:                 if ¬can_increase then
13:                     break
14:                 end if
15:             end for
16:             if can_increase then
17:                 S' ← S + {e}
18:                 break
19:             end if
20:         end if
21:     end for
22:     return S'
23: end function
```



Critério de parada

5

CRITÉRIO DE PARADA

YOU CAN ALSO SPLIT YOUR CONTENT

White

Is the color of milk and fresh snow, the color produced by the combination of all the colors of the visible spectrum.

Black

Is the color of ebony and of outer space. It has been the symbolic color of elegance, solemnity and authority.

IN TWO OR THREE COLUMNS

Yellow

Is the color of gold, butter and ripe lemons. In the spectrum of visible light, yellow is found between green and orange.

Blue

Is the colour of the clear sky and the deep sea. It is located between violet and green on the optical spectrum.

Red

Is the color of blood, and because of this it has historically been associated with sacrifice, danger and courage.

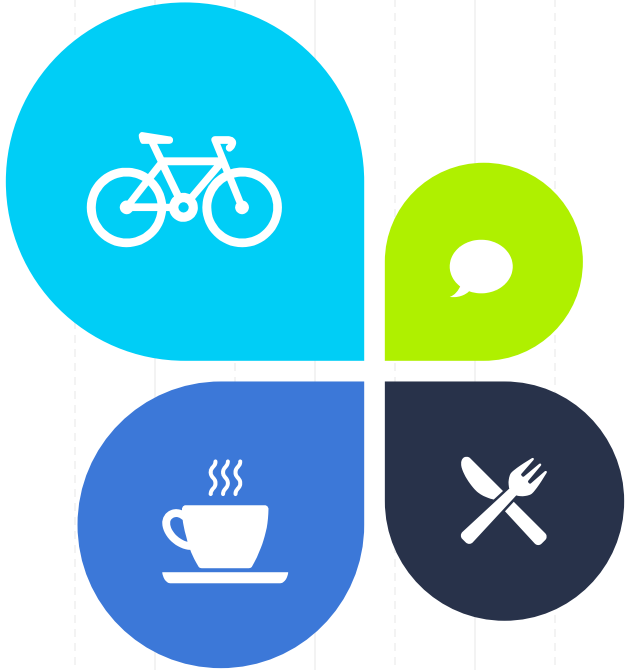
A PICTURE IS WORTH A THOUSAND WORDS

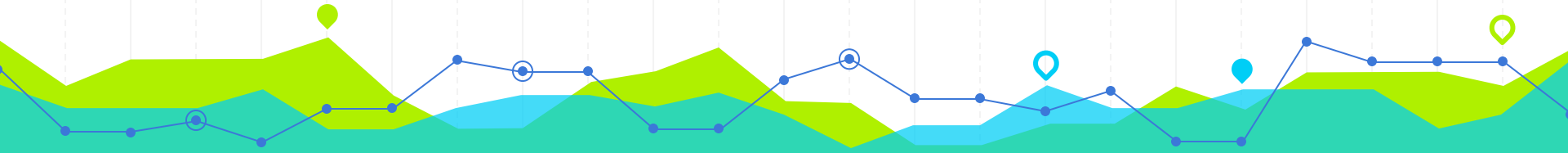
A complex idea can be conveyed with just a single still image.

Namely making it possible to absorb large amounts of data quickly.


WANT BIG IMPACT?
USE BIG IMAGE.

USE CHARTS TO EXPLAIN YOUR IDEAS





89,526,124

Whoa! That’s a big number, aren’t you proud?

89,526,124$

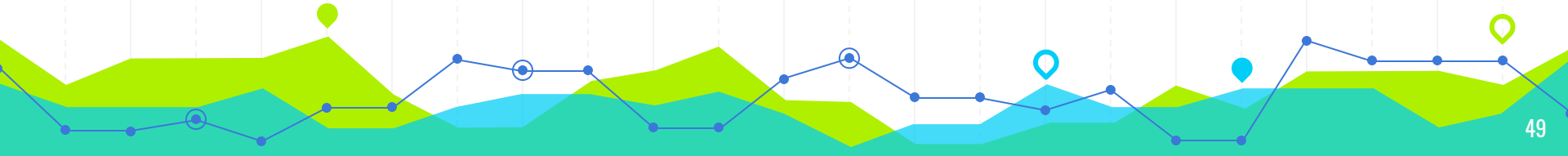
That’s a lot of money

185,244 users

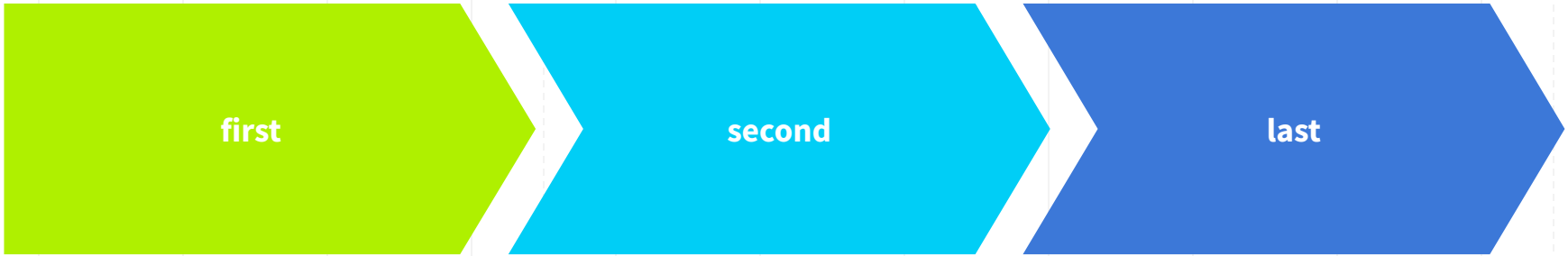
And a lot of users

100%

Total success!



OUR PROCESS IS EASY



LET'S REVIEW SOME CONCEPTS

Yellow

Is the color of gold, butter and ripe lemons. In the spectrum of visible light, yellow is found between green and orange.

Blue

Is the colour of the clear sky and the deep sea. It is located between violet and green on the optical spectrum.

Red

Is the color of blood, and because of this it has historically been associated with sacrifice, danger and courage.

Yellow

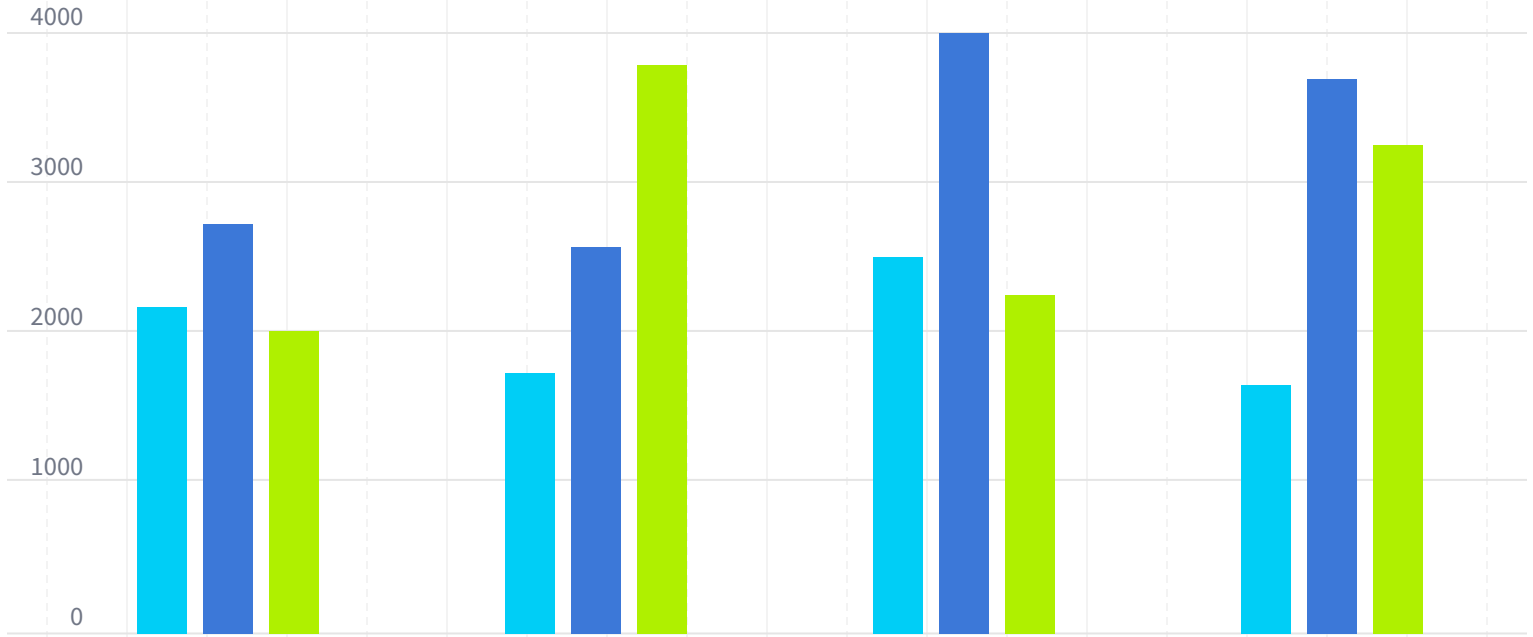
Is the color of gold, butter and ripe lemons. In the spectrum of visible light, yellow is found between green and orange.

Blue

Is the colour of the clear sky and the deep sea. It is located between violet and green on the optical spectrum.

Red

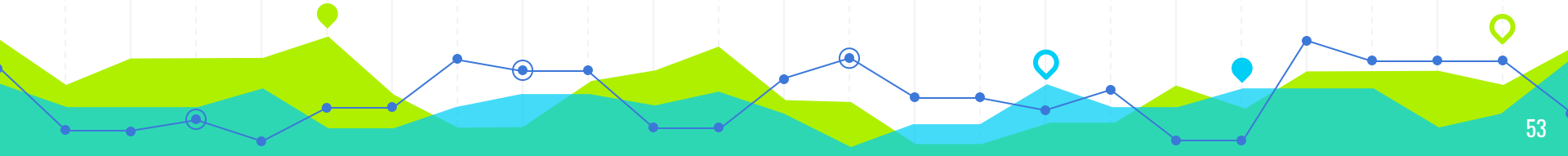
Is the color of blood, and because of this it has historically been associated with sacrifice, danger and courage.



You can insert graphs from Excel or Google Sheets

OBRIGADO!

Alguma dúvida?



SlidesCarnival icons are editable shapes.

This means that you can:

- Resize them without losing quality.
- Change fill color and opacity.
- Change line color, width and style.

Isn't that nice? :)

Examples: