


# Trabalho Prático

-


# Metaheurísticas

Coloração dos vértices de um grafo





# Problema de Coloração de vértices de um Grafo



# Coloração de vértices

Uma coloração de vértices consiste em atribuir uma cor a cada vértice de um grafo de forma que não exista um par de vértices adjacentes com a mesma cor.

O problema da coloração de vértices consiste em encontrar o número mínimo de cores necessárias para a coloração de determinado grafo. Este número é chamado número cromático.

# Modelagem Matemática

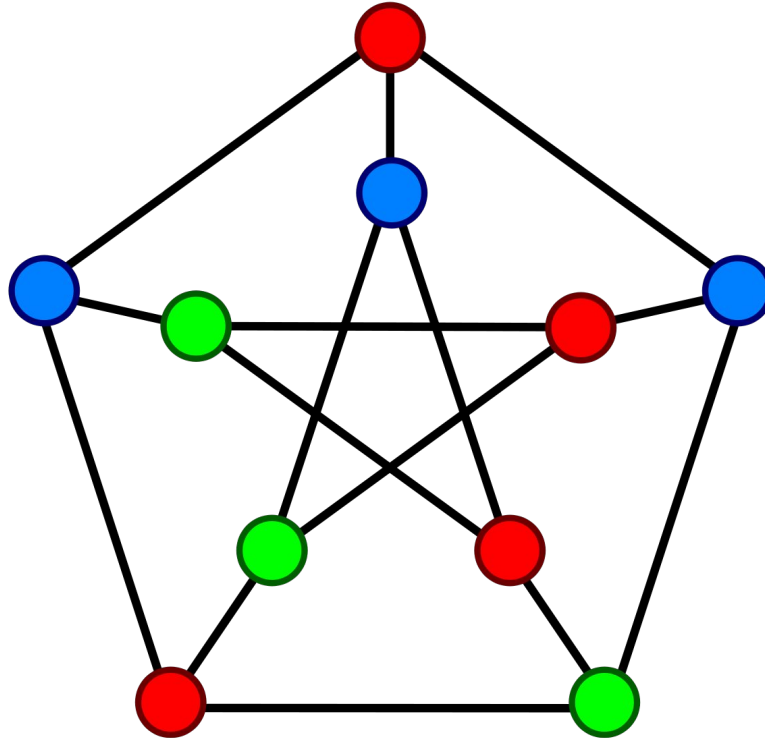
$$\text{(ASS-S) } \min \sum_{1 \leq i \leq H} w_i \quad (1)$$

$$\text{s.t. } \sum_{i=1}^H x_{v,i} = 1 \quad \forall v \in V \quad (2)$$

$$x_{u,i} + x_{v,i} \leq w_i \quad \forall (u,v) \in E, \quad i = 1, \dots, H \quad (3)$$

$$x_{v,i}, w_i \in \{0, 1\} \quad \forall v \in V, \quad i = 1, \dots, H \quad (4)$$

# Exemplo de Instância



# Metodologia: Instâncias

# Instâncias Utilizadas

| Instância      | Vértices | Arestas | Densidade(%) | Ótimo |
|----------------|----------|---------|--------------|-------|
| fpsol2.i.1.col | 496      | 11654   | 0.05         | 65    |
| inithx.i.1.col | 864      | 18707   | 2.51         | 54    |
| inithx.i.3.col | 621      | 13969   | 3.63         | 31    |
| le450_5a.col   | 450      | 5714    | 2.83         | 5     |
| le450_25d.col  | 450      | 17425   | 8.62         | 25    |
| miles250.col   | 128      | 774     | 4.76         | 8     |
| miles1500.col  | 128      | 10396   | 63.95        | 73    |
| myciel3.col    | 11       | 20      | 18.18        | 4     |
| qg.order60.col | 3600     | 212400  | 1.64         | 60    |
| queen5_5.col   | 25       | 320     | 53.33        | 5     |

# Grupo 1

- Lise Arantes
- Lucas Mateus
- Rafaela Martins



# Metodologia: Heurísticas

# Abordagens Heurísticas

**Heurística 1:** *Abduction*: Sugerida pela aluna Rafaela Martins, a heurística denominada *Abduction* começa com uma lista ordenada por grau, com todos os vértices presentes no grafo. Inicialmente se retira o vértice de maior grau entre todos os demais. Feito isso, adiciona-se uma flag de colorido e atribui-se a menor cor possível aquele vértice e o elimina da lista.

Feito isso, retira-se todos os seus adjacentes da lista ordenada e atribui-se esses adjacentes a uma outra lista chamada de “temporária”, além disso as flags de todos os adjacentes retirados mudam para visitados. Se a lista não estiver vazia, o algoritmo segue retirando e colorindo os vértices de maior grau. [Fonte Desenvolvida pelo próprio autor]

# Heurística 1: *Abduction* [Fonte: Próprio Autor]

```
# cria lista ordenada de vértices por grau (priority queue)
list = self.graph.vertex_list()
for v in self.graph.vertex_list():
    self.graph.calc_degree(v)
    entry = (self.graph.num_vertex - v.degree, next(my_count), v)
    heappush(ordered_vertices, entry)

# Solver
while len(ordered_vertices) > 0:
    color += 1
    while len(ordered_vertices) > 0:
        entry = heappop(ordered_vertices)
        v = entry[2]

        # aleatoriedade
        same_degree = [v]
        if len(ordered_vertices) > 0:
            entry = heappop(ordered_vertices)
            while entry[2].degree == v.degree:
                same_degree.append(entry[2])
                if len(ordered_vertices) > 0:
                    entry = heappop(ordered_vertices)
                else:
                    break
            if entry[2].degree != v.degree:
                heappush(ordered_vertices, entry)
        chosen = randint(0, len(same_degree)-1)
        v = same_degree[chosen]
        for i, w in enumerate(same_degree):
            if i != chosen:
                entry = (self.graph.num_vertex - w.degree, next(my_count), w)
                heappush(ordered_vertices, entry)
        same_degree = []

        if v.state == State.UNVISITED:
            v.set_color(color)
            for w in self.graph.adjacent_list(v):
                w.change_state(State.VISITED)
            v.change_state(State.COLORFUL)
        else:
            entry = (self.graph.num_vertex - v.degree, next(my_count), v)
            heappush(temp, entry)
        ordered_vertices = temp
        temp = []
        for _, v in ordered_vertices:
            v.change_state(State.UNVISITED)

# imprime resultado
print('Total de cores: {}'.format(color))
```

# Metodologia: Metaheurísticas

# Abordagens Metaheurísticas

Metaheurística 1: GRASP

Metaheurística 2: VND [Celso Ribeiro]

Metaheurística 3: Reduced VNS [Marconi]

# Metaheurística 1: GRASP [Fonte]

Algoritmo GRASP ( $\delta, N, s$ )

```
1       $f^* \leftarrow \infty$ ;    { valor da melhor solução obtida até então }
2      IterGRASP  $\leftarrow 0$ ; { Número de iterações GRASP }
3      enquanto (IterGRASP  $< N$ ) faça
4          IterGRASP  $\leftarrow$  IterGRASP + 1;
5           $s_0 \leftarrow$  ConstruaSolucao( $\delta$ );
6           $s \leftarrow$  BuscaLocal( $s_0$ );
7          se ( $f(s) < f^*$ )
8              então
9                   $s^* \leftarrow s$ ;
10                  $f^* \leftarrow f(s)$ ;
11             fim-se;
12     fim-enquanto;
13      $s \leftarrow s^*$ ;
14     Retorne  $s$ ;
fim GRASP;
```

# Metaheurística 1: Implementação

Parâmetros

*alfa*:0.7

Iterações: 100

Vizinhança: Buscou-se encontrar alguma vizinhança factível que possuísse um número cromático menor do que o menor encontrado até o momento.

Percorre-se todos os vértices.

Implementada por Lise Arantes

# Metaheurística 1: Implementação

Parâmetros

*alfa*:0.1

Iterações: 1000

Vizinhança: Buscou-se encontrar alguma vizinhança factível que possuísse um número cromático menor do que o menor encontrado até o momento. Vetor de índices aleatorizados é criado de tamanho  $\alpha * \text{número de vértices}$ .

Implementada por Lise Arantes



# Metaheurística 2: VND [Celso Ribeiro]

## **Procedimento VND( $s$ )**

$r$  = Número de procedimentos de refinamento;

$k \leftarrow 1$ ;

**enquanto**  $k \leq r$  **faça**

    Seja  $s'$  um ótimo local segundo o  $k$ -ésimo procedimento de refinamento;

**se**  $f(s') < f(s)$  **então**

$s \leftarrow s'$ ;

$k \leftarrow 1$ ;

**senão**

$k \leftarrow k + 1$ ;

**fim-enquanto**

    Retorne  $s$ ;

**fim VND**

# Metaheurística 2: Implementação

Implementada pelo integrante Lucas Mateus

Foi implementada a estrutura clássica do VND, utilizando-se de 3 estruturas de vizinhança, citadas abaixo:

- Estrutura de vizinhança básica
- 1ª estrutura de vizinhança sugerida pelo grupo
- 2ª estrutura de vizinhança sugerida pelo grupo

GitHub: <https://github.com/LiseAr/metaheuristic/blob/master/vnd.py>

# Metaheurística 3: Reduced VNS [Marconi]

## Procedimento RVNS

**Enquanto** o tempo máximo não for alcançado **faça**

$k \leftarrow 1$

**Enquanto**  $k = 1$  **faça**

Sortear aleatoriamente dois pontos distintos do vetor de demandas  $V$ ;

Execute movimentos de troca do tipo “2-opt” com as posições sorteadas;

Gerar solução gulosa conforme pseudocódigo da Figura 1;

Executar procedimento de melhoria conforme pseudocódigo da Figura 2;

**Se** houve melhora na FO **então**

Atualize a melhor solução obtida até o momento;

**Senão**  $k \leftarrow k + 1$ ;

**Fim enquanto**

**Enquanto**  $k = 2$  **faça**

Sortear aleatoriamente três pontos distintos do vetor de demandas  $V$ ;

Execute movimentos de troca do tipo “3-opt” com as posições sorteadas;

Gerar solução gulosa conforme pseudocódigo da Figura 1;

Executar procedimento de melhoria conforme pseudocódigo da Figura 2;

**Se** houve melhora na FO **então**

Atualize a melhor solução obtida até o momento;

$k \leftarrow 1$ ;

**Fim enquanto**

**Fim enquanto** {tempo limite do RVNS}

**Fim procedimento** RVNS

# Metaheurística 3: Implementação

Implementada pela integrante Rafaela Martins

Foram utilizadas as mesmas estruturas de vizinhança do VND, porém aplicando as diferenças que o Reduced VNS tem para o VND, que consistem em uma condição de parada a mais para o algoritmo e uma busca local realizada na vizinhança gerada pela estrutura.

GitHub: <https://github.com/LiseAr/metaheuristic/blob/master/vns.py>



# Experimentos Realizados



# Setup Experimental

Abaixo serão indicadas as tabelas que contém os dados dos experimentos realizados. Para todos os experimentos foi utilizado computador com as seguintes configurações:

- Processador Intel i5
- Memória RAM 8gb

Para cada uma das metaheurísticas foram realizadas 100 iterações, sendo registrados os dados e calculados mínimo, máximo, média, etc.

Resultados: **Heurística**

| Instância      | Min | Média  | Mediana | Max | SD    | VAR   | BEST | Desvio % |
|----------------|-----|--------|---------|-----|-------|-------|------|----------|
| fpsol2.i.1.col | 65  | 65     | 65      | 65  | 0     | 0     | 65   | 0        |
| inithx.i.1.col | 54  | 54     | 54      | 54  | 0     | 0     | 54   | 0        |
| inithx.i.3.col | 31  | 31     | 31      | 31  | 0     | 0     | 31   | 0        |
| le450_5a.col   | 11  | 11.683 | 12      | 12  | 0.465 | 0.216 | 5    | 120      |
| le450_25d.col  | 29  | 30.554 | 30      | 32  | 0.636 | 0.405 | 25   | 16       |
| miles250.col   | 8   | 8.247  | 8       | 9   | 0.431 | 0.186 | 8    | 0        |
| miles1500.col  | 73  | 73     | 73      | 73  | 0     | 0     | 73   | 0        |
| myciel3.col    | 4   | 4      | 4       | 4   | 0     | 0     | 4    | 0        |
| qg.order60.col | 64  | 64.386 | 64      | 66  | 0.525 | 0.276 | 60   | 6.66     |
| queen5_5.col   | 5   | 6.722  | 7       | 8   | 0.809 | 0.655 | 5    | 0        |



Resultados: **GRASP**

| Instância      | Min | Média  | Mediana | Max | SD    | VAR   | BEST | Desvio % |
|----------------|-----|--------|---------|-----|-------|-------|------|----------|
| fpsol2.i.1.col | 65  | 65.202 | 65      | 66  | 0.401 | 0.161 | 65   | 0        |
| inithx.i.1.col | 54  | 54.040 | 54      | 55  | 0.196 | 0.038 | 54   | 0        |
| inithx.i.3.col | 31  | 31.050 | 31      | 32  | 0.218 | 0.047 | 31   | 0        |
| le450_5a.col   | 12  | 13.272 | 13      | 15  | 0.616 | 0.380 | 5    | 140      |
| le450_25d.col  | 34  | 37.393 | 37      | 41  | 1.229 | 1.511 | 25   | 36       |
| miles250.col   | 8   | 9.959  | 10      | 11  | 0.777 | 0.604 | 8    | 0        |
| miles1500.col  | 73  | 76.111 | 76      | 80  | 1.476 | 2.179 | 73   | 0        |
| myciel3.col    | 4   | 4.050  | 4       | 5   | 0.218 | 0.047 | 4    | 0        |
| qg.order60.col | 63  | 64.393 | 64      | 66  | 0.565 | 0.319 | 60   | 5        |
| queen5_5.col   | 5   | 7.797  | 8       | 10  | 1.034 | 1.070 | 5    | 0        |

Resultados: **VND**

| <b>Instância</b> | <b>Min</b> | <b>Média</b> | <b>Mediana</b> | <b>Max</b> | <b>SD</b> | <b>VAR</b> | <b>BEST</b> | <b>Desvio %</b> |
|------------------|------------|--------------|----------------|------------|-----------|------------|-------------|-----------------|
| fpsol2.i.1.col   | 69         | 82.128       | 80             | 121        | 8.612     | 74.171     | 65          | 6.15            |
| inithx.i.1.col   | 59         | 73.079       | 73             | 99         | 8.239     | 67.894     | 54          | 9.25            |
| inithx.i.3.col   | 38         | 51.742       | 50             | 82         | 9.427     | 88.884     | 31          | 22.58           |
| le450_5a.col     | 18         | 30.653       | 28             | 59         | 8.205     | 67.335     | 5           | 260             |
| le450_25d.col    | 42         | 55.376       | 52.5           | 106        | 9.705     | 94.195     | 25          | 68              |
| miles250.col     | 13         | 26.475       | 25             | 51         | 8.522     | 72.625     | 8           | 62.5            |
| miles1500.col    | 76         | 83.059       | 83             | 92         | 2.834     | 8.036      | 73          | 4.10            |
| myciel3.col      | 4          | 5.772        | 6              | 9          | 1.241     | 1.542      | 4           | 0               |
| qg.order60.col   | 68         | 83.366       | 83             | 122        | 9.032     | 81.578     | 60          | 13.33           |
| queen5_5.col     | 9          | 13.742       | 14             | 19         | 2.173     | 4.725      | 5           | 80              |

Resultados: **VNS**

| <b>Instância</b> | <b>Min</b> | <b>Média</b> | <b>Mediana</b> | <b>Max</b> | <b>SD</b> | <b>VAR</b> | <b>BEST</b> | <b>Desvio %</b> |
|------------------|------------|--------------|----------------|------------|-----------|------------|-------------|-----------------|
| fpsol2.i.1.col   | 68         | 83.435       | 82.5           | 126        | 9.607     | 92.305     | 65          | 4.61            |
| inithx.i.1.col   | 60         | 72.009       | 71             | 106        | 7.718     | 59.574     | 54          | 11.11           |
| inithx.i.3.col   | 36         | 50.485       | 49             | 83         | 9.102     | 82.863     | 31          | 16.12           |
| le450_5a.col     | 16         | 30.603       | 29             | 55         | 8.012     | 64.199     | 5           | 220             |
| le450_25d.col    | 42         | 53.742       | 53             | 85         | 7.717     | 59.557     | 25          | 68              |
| miles250.col     | 13         | 24.613       | 23             | 45         | 6.050     | 36.613     | 8           | 62.5            |
| miles1500.col    | 73         | 82.059       | 82             | 88         | 2.687     | 7.224      | 73          | 0               |
| myciel3.col      | 4          | 4.920        | 5              | 7          | 0.684     | 0.468      | 4           | 0               |
| qg.order60.col   | 68         | 83.940       | 83             | 108        | 8.394     | 70.471     | 60          | 13.33           |
| queen5_5.col     | 9          | 12.207       | 12             | 16         | 1.366     | 1.867      | 5           | 80              |



# Resultados: Comparativos entre as Abordagens



Resultado Comparativo: **Mínimos**



| Instância  | H  | GRASP | VND | VNS |
|------------|----|-------|-----|-----|
| fpsol2.i.1 | 65 | 65    | 69  | 68  |
| inithx.i.1 | 54 | 54    | 59  | 60  |
| inithx.i.3 | 31 | 31    | 38  | 36  |
| le450_5a   | 11 | 12    | 18  | 16  |
| le450_25d  | 29 | 34    | 42  | 42  |
| miles250   | 8  | 8     | 13  | 13  |
| miles1500  | 73 | 73    | 76  | 73  |
| myciel3    | 4  | 4     | 4   | 4   |
| qg.order60 | 64 | 63    | 68  | 68  |
| queen5_5   | 5  | 5     | 9   | 9   |

Resultado Comparativo: **Medianas**

| Instância  | H  | GRASP | VND  | VNS  |
|------------|----|-------|------|------|
| fpsol2.i.1 | 65 | 65    | 80   | 82.5 |
| inithx.i.1 | 54 | 54    | 73   | 71   |
| inithx.i.3 | 31 | 31    | 50   | 49   |
| le450_5a   | 12 | 13    | 28   | 29   |
| le450_25d  | 30 | 37    | 52.5 | 53   |
| miles250   | 8  | 10    | 25   | 23   |
| miles1500  | 73 | 76    | 83   | 82   |
| myciel3    | 4  | 4     | 6    | 5    |
| qg.order60 | 64 | 64    | 83   | 83   |
| queen5_5   | 7  | 8     | 14   | 12   |

Resultado Comparativo: **Desvio % (Opt)**

| Instância  | H    | GRASP | VND   | VNS   |
|------------|------|-------|-------|-------|
| fpsol2.i.1 | 0    | 0     | 6.15  | 4.61  |
| inithx.i.1 | 0    | 0     | 9.25  | 11.11 |
| inithx.i.3 | 0    | 0     | 22.58 | 16.12 |
| le450_5a   | 120  | 140   | 260   | 220   |
| le450_25d  | 16   | 36    | 68    | 68    |
| miles250   | 0    | 0     | 62.5  | 62.5  |
| miles1500  | 0    | 0     | 4.1   | 0     |
| myciel3    | 0    | 0     | 0     | 0     |
| qg.order60 | 6.66 | 5     | 13.33 | 13.33 |
| queen5_5   | 0    | 0     | 80    | 80    |

# Grupo 2

- Eduardo Mezêncio
- Eduardo Miranda
- Augusto Amaral

# Metodologia: Heurísticas

# Abordagens Heurísticas

## **Heurística 1:** Guloso 1 (Proposto pelo grupo)

Selecionar vértices em ordem arbitrária e associar a primeira cor ainda não associada a nenhum de seus vizinhos

## **Heurística 2:** Guloso 2 (Proposto pelo grupo)

Selecionar vértices ordenados por grau, do maior para o menor, e associar a primeira cor ainda não associada a nenhum dos vizinhos do vértice selecionado

## **Heurística 3:** GRP (Proposto pelo grupo)

GRASP sem a busca local



# Heurística 1: Guloso 1 (Proposto pelo grupo)

---

## Algorithm 1 Guloso 1

---

**Ensure:** guloso1(grafo)

```
1: cores  $\leftarrow$  lista do tamanho do grafo
2: for all vértices  $\in$  grafo do
3:   cor  $\leftarrow$  0
4:   while  $\exists$  colisão de cores com vértices adjacentes do
5:     cor  $\leftarrow$  cor + 1
6:   end while
7:   cores[vértice]  $\leftarrow$  cor
8: end for
```

---

# Heurística 2: Guloso 2 (Proposto pelo grupo)

---

## Algorithm 2 Guloso 2

---

**Ensure:** guloso2(grafo)

- 1: cores  $\leftarrow$  lista do tamanho do grafo
  - 2: ordena grafo pela cardinalidade dos vértices
  - 3: **for all** vértices  $\in$  grafo **do**
  - 4:   cor  $\leftarrow$  0
  - 5:   **while**  $\exists$  colisão de cores com vértices adjacentes **do**
  - 6:     cor  $\leftarrow$  cor + 1
  - 7:   **end while**
  - 8:   cores[vértice]  $\leftarrow$  cor
  - 9: **end for**
-

# Heurística 3: GRP (Proposto pelo grupo)

---

## Algorithm 3 GRAP

---

**Ensure:** GRAP(MaxIter, Seed)

- 1: **for**  $k \leftarrow 1$  to MaxIter **do**
  - 2:     Solution  $\leftarrow$  GreedyRandomizedConstruction(Seed)
  - 3:     UpdateSolution(Solution, BestSolution)
  - 4: **end for**
  - 5: **return** BestSolution
-

# Metodologia: Metaheurísticas

# Metaheurística 1: GRASP+Hill Climb

## Algorithm 5 GRASP

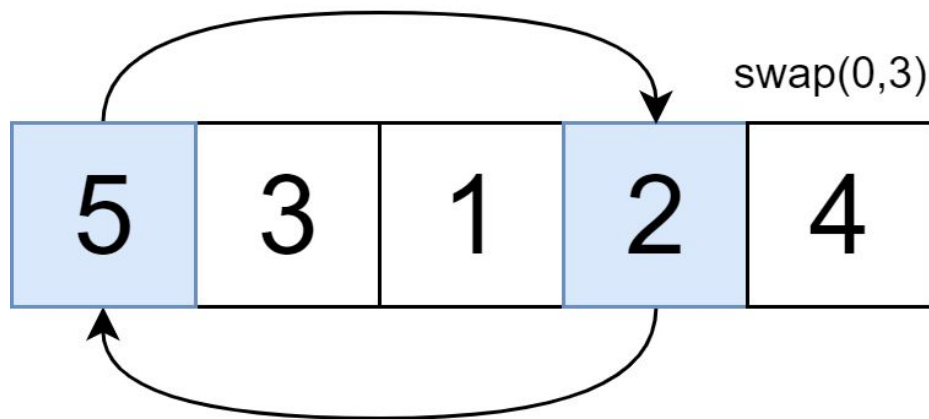
**Ensure:** GRASP(MaxIter, Seed)

- 1: **for**  $k \leftarrow 1$  to MaxIter **do**
- 2:   Solution  $\leftarrow$  GreedyRandomizedConstruction(Seed)
- 3:   Solution  $\leftarrow$  LocalSearch(Solution)
- 4:   UpdateSolution(Solution, BestSolution)
- 5: **end for**
- 6: **return** BestSolution

# GRASP+HC: Implementação

| GRASP                     |    |
|---------------------------|----|
| MAX ITERAÇÕES SEM MELHORA | 10 |
| TAMANHO RCL               | 5  |

Estrutura de vizinhança: SWAP



# Metaheurística 2: GRASP+Simulated Annealing

---

**Algoritmo 1** SIMULATED-ANNEALING( $f(\cdot)$ ,  $N(\cdot)$ ,  $\alpha$ ,  $SAMax$ ,  $T_0$ ,  $s$ )

---

```
1:  $s^* \leftarrow s$ ;  $Iter \leftarrow 0$ ;  $T \leftarrow T_0$ 
2: while ( $T > 0$ ) do
3:   while ( $Iter < SAMax$ ) do
4:      $Iter \leftarrow Iter + 1$ 
5:     Gere um vizinho qualquer  $s' \in N(s)$ 
6:      $\Delta \leftarrow f(s') - f(s)$ 
7:     if ( $\Delta < 0$ ) then
8:        $s \leftarrow s'$ 
9:       if ( $f(s') < f(s^*)$ ) then
10:         $s^* \leftarrow s'$ 
11:       end if
12:     else
13:       Tome  $x \in [0, 1]$ 
14:       if ( $x < e^{-\Delta/T}$ ) then
15:         $s \leftarrow s'$ 
16:       end if
17:     end if
18:   end while
19:    $T \leftarrow \alpha T$ ;  $Iter \leftarrow 0$ 
20: end while
21: return  $s^*$ 
```

SA [Van Laarhoven, P. J., & Aarts, E. H. (1987).]

---

# GRASP+SA: Implementação

| GRASP                     |    |
|---------------------------|----|
| MAX ITERAÇÕES SEM MELHORA | 10 |
| TAMANHO RCL               | 5  |

Estrutura de vizinhança: SWAP

Eduardo Miranda ([GitHub](#))

| SIMULATED ANNEALING  |     |
|----------------------|-----|
| SOLUÇÃO INICIAL      | -   |
| ALFA                 | 0.9 |
| QTD VIZINHOS GERADOS | 10  |
| TEMP INICIAL         | 100 |
| TEMP FINAL           | 10  |
| QTD REAQUECIMENTO    | 1   |



# Metaheurística 3: GRASP+SA+Path-Relinking

---

## Algorithm 4 Path Relinking

---

**Ensure:** PathRelinking( $x, y$ )

- 1: Calcule o conjunto  $\Delta(x, y)$
  - 2:  $tentativas \leftarrow \log len(\Delta)$
  - 3: **for**  $i \in \text{range}(tentativas)$  **do**
  - 4:    $order \leftarrow \text{shuffled}(\Delta)$
  - 5:   **for**  $movimento \in order$  **do**
  - 6:      $solucao \leftarrow \text{aplica movimento}$
  - 7:     **if**  $solucao > \text{melhor}$  **then**
  - 8:       Atualiza melhor
  - 9:     **end if**
  - 10:   **end for**
  - 11: **end for**
  - 12: **return** melhor
-

# GRASP+SA+PR: Implementação

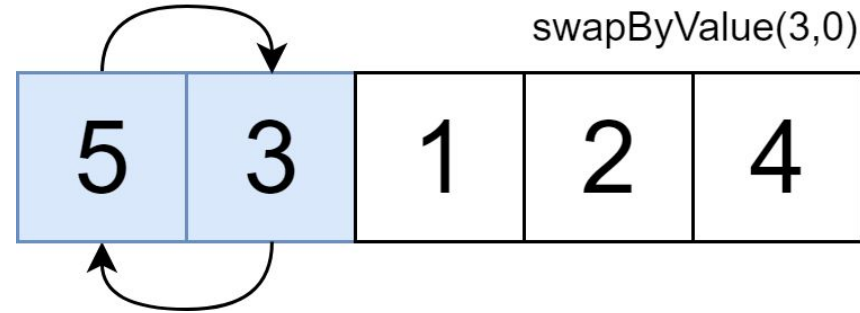
Parâmetros GRASP e SA: Mesmos da **GRASP+SA**

Estrutura de vizinhança PR: **SWAP by value**

## Path-relinking

TENTATIVAS

$$\max(\lfloor \ln(|S|) \rfloor, 1)$$





# Experimentos Realizados



| Experimento | Método utilizado | Parâmetros  |
|-------------|------------------|---|
| I           | Guloso 1         | -   |
| II          | Guloso 2         | -   |
| III         | GRP              | <b>local_search</b> = None; <b>max_iter</b> = 10;<br><b>rcl_size</b> = 5  |
| IV          | GRASP + HC       | <b>local_search</b> = hill_climbing;<br><b>neighbor_structure</b> =swap_by_position;<br><b>intial_solution</b> =graph   |
| V           | GRASP + SA       | local_search=simulated_annealing; alpha=0.9;<br>initial_temp=100; final_temp=10; reheat_times=1                         |
| VI          | GRASP + SA + PR  | local_search=simulated_annealing;<br>neighbor_structure=swap_by_value;<br>steps=<br>$\max(\lfloor \ln( S ) \rfloor, 1)$ |

# Setup Experimental

- Intel(R) Xeon(R) CPU ES-2430 v2 @ 2.50GHz
- 12 núcleos
- 48 GB de Memória RAM

# Resultados

## **GULOSO 1**

| Instância  | Min | Média  | Mediana | Max | SD    | VAR   | BEST | Desvio % |
|------------|-----|--------|---------|-----|-------|-------|------|----------|
| fpsol2.i.1 | 65  | 65     | 65      | 65  | 0     | 0     | 65   | 0        |
| inithx.i.1 | 54  | 54     | 54      | 54  | 0     | 0     | 54   | 0        |
| inithx.i.3 | 31  | 31.062 | 31.0    | 32  | 0.058 | 0.242 | 31   | 0        |
| le450_5a   | 12  | 13.218 | 13.0    | 14  | 0.295 | 0.543 | 5    | 140      |
| le450_25d  | 33  | 36     | 36      | 38  | 0.895 | 0.946 | 25   | 32       |
| miles250   | 8   | 9.322  | 9.0     | 11  | 0.364 | 0.603 | 8    | 0        |
| miles1500  | 73  | 74.187 | 74.0    | 77  | 0.839 | 0.916 | 73   | 0        |
| myciel3    | 4   | 4      | 4       | 4   | 0     | 0     | 4    | 0        |
| qg.order60 | 64  | 65.031 | 65      | 67  | 0.321 | 0.567 | 60   | 6,66     |
| queen5_5   | 6   | 7.489  | 7       | 9   | 0.666 | 0.816 | 5    | 0        |

# Resultados

## **GULOSO 2**



| Instância  | Min | Média  | Mediana | Max | SD    | VAR   | BEST | Desvio % |
|------------|-----|--------|---------|-----|-------|-------|------|----------|
| fpsol2.i.1 | 65  | 65     | 65      | 65  | 0     | 0     | 65   | 0        |
| inithx.i.1 | 54  | 54     | 54      | 54  | 0     | 0     | 54   | 0        |
| inithx.i.3 | 31  | 31     | 31      | 31  | 0     | 0     | 31   | 0        |
| le450_5a   | 11  | 11.666 | 12.0    | 12  | 0.222 | 0.471 | 5    | 120      |
| le450_25d  | 30  | 30.614 | 31.0    | 32  | 0.320 | 0.565 | 25   | 20       |
| miles250   | 8   | 8.229  | 8.0     | 9   | 0.176 | 0.420 | 8    | 0        |
| miles1500  | 73  | 73     | 73      | 73  | 0     | 0     | 73   | 0        |
| myciel3    | 4   | 4.0    | 4.0     | 4   | 0.0   | 0.0   | 4    | 0        |
| qg.order60 | 64  | 65.260 | 65      | 68  | 0.630 | 0.793 | 60   | 6,66     |
| queen5_5   | 5   | 6.552  | 7       | 9   | 0.768 | 0.876 | 5    | 0        |

# Resultados

## GRP

| Instância  | Min | Média  | Mediana | Max | SD    | VAR   | BEST | Desvio % |
|------------|-----|--------|---------|-----|-------|-------|------|----------|
| fpsol2.i.1 | 65  | 65     | 65      | 65  | 0     | 0     | 65   | 0        |
| inithx.i.1 | 54  | 54     | 54      | 54  | 0     | 0     | 54   | 0        |
| inithx.i.3 | 31  | 31     | 31      | 31  | 0     | 0     | 31   | 0        |
| le450_5a   | 10  | 10.979 | 11      | 11  | 0.020 | 0.142 | 5    | 100      |
| le450_25d  | 29  | 29.770 | 30      | 30  | 0.176 | 0.420 | 25   | 16       |
| miles250   | 8   | 8      | 8       | 8   | 0     | 0     | 8    | 0        |
| miles1500  | 73  | 73     | 73      | 73  | 0     | 0     | 73   | 0        |
| myciel3    | 4   | 4      | 4       | 4   | 0     | 0     | 4    | 0        |
| qg.order60 | 64  | 64.145 | 64      | 65  | 0.124 | 0.352 | 60   | 6,66     |
| queen5_5   | 5   | 5.25   | 5       | 7   | 0.208 | 0.456 | 5    | 0        |

# Resultados

## **GRASP+HC**

| Instância  | Min | Média  | Mediana | Max | SD       | VAR   | BEST | Desvio % |
|------------|-----|--------|---------|-----|----------|-------|------|----------|
| fpsol2.i.1 | 65  | 65     | 65      | 65  | 0        | 0     | 65   | 0        |
| inithx.i.1 | 54  | 54     | 54      | 54  | 0        | 0     | 54   | 0        |
| inithx.i.3 | 31  | 31     | 31      | 31  | 0        | 0     | 31   | 0        |
| le450_5a   | 10  | 10625  | 11      | 11  | 0.234375 | 0.484 | 5    | 100      |
| le450_25d  | 29  | 29     | 29      | 29  | 0        | 0     | 25   | 16       |
| miles250   | 8   | 8      | 8       | 8   | 0        | 0     | 8    | 0        |
| miles1500  | 73  | 73     | 73      | 73  | 0        | 0     | 73   | 0        |
| myciel3    | 4   | 4      | 4       | 4   | 0        | 0     | 4    | 0        |
| qg.order60 | 63  | 63.937 | 64      | 64  | 0.0585   | 0.242 | 60   | 5        |
| queen5_5   | 5   | 5      | 5       | 5   | 0        | 0     | 5    | 0        |

# Resultados

## **GRASP+SA**

| Instância  | Min | Média  | Mediana | Max | SD     | VAR   | BEST | Desvio % |
|------------|-----|--------|---------|-----|--------|-------|------|----------|
| fpsol2.i.1 | 65  | 65     | 65      | 65  | 0      | 0     | 65   | 0        |
| inithx.i.1 | 54  | 54     | 54      | 54  | 0      | 0     | 54   | 0        |
| inithx.i.3 | 31  | 31     | 31      | 31  | 0      | 0     | 31   | 0        |
| le450_5a   | 10  | 10.697 | 11      | 11  | 0.210  | 0.459 | 5    | 100      |
| le450_25d  | 28  | 28.979 | 29      | 29  | 0.020  | 0.142 | 25   | 12       |
| miles250   | 8   | 8      | 8       | 8   | 0      | 0     | 8    | 0        |
| miles1500  | 73  | 73     | 73      | 73  | 0      | 0     | 73   | 0        |
| myciel3    | 4   | 4      | 4       | 4   | 0      | 0     | 4    | 0        |
| qg.order60 | 63  | 63.927 | 64      | 64  | 0.0675 | 0.259 | 60   | 5        |
| queen5_5   | 5   | 5      | 5       | 5   | 0      | 0     | 5    | 0        |

# Resultados

## **GRASP+SA+PR**



| Instância  | Min | Média  | Mediana | Max | SD    | VAR   | BEST | Desvio % |
|------------|-----|--------|---------|-----|-------|-------|------|----------|
| fpsol2.i.1 | 65  | 65     | 65      | 65  | 0     | 0     | 65   | 0        |
| inithx.i.1 | 54  | 54     | 54      | 54  | 0     | 0     | 54   | 0        |
| inithx.i.3 | 31  | 31     | 31      | 31  | 0     | 0     | 31   | 0        |
| le450_5a   | 10  | 10.552 | 11      | 11  | 0.247 | 0.497 | 5    | 100      |
| le450_25d  | 29  | 29     | 29      | 29  | 0     | 0     | 25   | 16       |
| miles250   | 8   | 8      | 8       | 8   | 0     | 0     | 8    | 0        |
| miles1500  | 73  | 73     | 73      | 73  | 0     | 0     | 73   | 0        |
| myciel3    | 4   | 4      | 4       | 4   | 0     | 0     | 4    | 0        |
| qg.order60 | --  | --     | --      | --  | --    | --    | --   | --       |
| queen5_5   | 5   | 5      | 5       | 5   | 0     | 0     | 5    | 0        |



# Resultados: Comparativos entre as Abordagens



# Resultado Comparativo

## **Mínimos**

| Instância  | G1 | G2 | GRP | GRASP+HC | GRASP+SA | GRASP+SA+PR |
|------------|----|----|-----|----------|----------|-------------|
| fpsol2.i.1 | 65 | 65 | 65  | 65       | 65       | 65          |
| inithx.i.1 | 54 | 54 | 54  | 54       | 54       | 54          |
| inithx.i.3 | 31 | 31 | 31  | 31       | 31       | 31          |
| le450_5a   | 12 | 11 | 10  | 10       | 10       | 10          |
| le450_25d  | 33 | 30 | 29  | 29       | 28       | 29          |
| miles250   | 8  | 8  | 8   | 8        | 8        | 8           |
| miles1500  | 73 | 73 | 73  | 73       | 73       | 73          |
| myciel3    | 4  | 4  | 4   | 4        | 4        | 4           |
| qg.order60 | 64 | 64 | 64  | 63       | 63       | --          |
| queen5_5   | 6  | 5  | 5   | 5        | 5        | 5           |

# Resultado Comparativo

## **Medianas**

| Instância  | G1 | G2 | GRP | GRASP+HC | GRASP+SA | GRASP+SA+PR |
|------------|----|----|-----|----------|----------|-------------|
| fpsol2.i.1 | 65 | 65 | 65  | 65       | 65       | 65          |
| inithx.i.1 | 54 | 54 | 54  | 54       | 54       | 54          |
| inithx.i.3 | 31 | 31 | 31  | 31       | 31       | 31          |
| le450_5a   | 13 | 12 | 11  | 11       | 11       | 11          |
| le450_25d  | 36 | 31 | 30  | 29       | 29       | 29          |
| miles250   | 9  | 8  | 8   | 8        | 8        | 8           |
| miles1500  | 74 | 73 | 73  | 73       | 73       | 73          |
| myciel3    | 4  | 4  | 4   | 4        | 4        | 4           |
| qg.order60 | 65 | 65 | 64  | 64       | 64       | --          |
| queen5_5   | 7  | 7  | 5   | 5        | 5        | 5           |

# Resultado Comparativo

## **Desvio % (Opt)**

| Instância  | G1   | G2   | GRP  | GRASP+HC | GRASP+SA | GRASP+SA+PR |
|------------|------|------|------|----------|----------|-------------|
| fpsol2.i.1 | 0    | 0    | 0    | 0        | 0        | 0           |
| inithx.i.1 | 0    | 0    | 0    | 0        | 0        | 0           |
| inithx.i.3 | 0    | 0    | 0    | 0        | 0        | 0           |
| le450_5a   | 140  | 120  | 100  | 100      | 100      | 100         |
| le450_25d  | 32   | 20   | 16   | 16       | 12       | 16          |
| miles250   | 0    | 0    | 0    | 0        | 0        | 0           |
| miles1500  | 0    | 0    | 0    | 0        | 0        | 0           |
| myciel3    | 0    | 0    | 0    | 0        | 0        | 0           |
| qg.order60 | 6,66 | 6,66 | 6,66 | 5        | 5        | --          |
| queen5_5   | 0    | 0    | 0    | 0        | 0        | 0           |



# Grupo 3

- Jefferson Marques
- Vinícius Araújo
- Vinícius Moraes

# Metodologia: Heurísticas

# Abordagens Heurísticas

## Heurística 1: Coloração Sequencial

A ordem do vetor é de 0 até o número de vértices.

# Heurística 1: Coloração Sequencial [Paulo Feofiloff]

```
#define UGraph Graph
```

```
/* Esta função calcula uma coloração válida dos vértices do grafo não-dirigido G e devolve o número de cores usadas.
A coloração é armazenada no vetor color[]. */
```

```
int UGRAPHseqColoring( UGraph G, int *color)
{
    int k = 0;
    for (vertex v = 0; v < G->V; ++v) color[v] = -1;

    for (vertex v = 0; v < G->V; ++v) {
        // as cores são 0..k-1
        bool available[100];
        int c;
        for (c = 0; c < k; ++c) available[c] = true;
        for (link a = G->adj[v]; a != NULL; a = a->next) {
            if (color[a->w] != -1)
                available[color[a->w]] = false;
        } // A
        c = 0;
        while (c < k && !available[c]) ++c;
        if (c < k) color[v] = c;
        else color[v] = k++;
    }
    return k;
}
```

# Heurística 1: Coloração Sequencial [Paulo Feofiloff]

Na Heurística de Coloração Sequencial o grafo é colorido de acordo com a sequência que os vértices são lidos do arquivo de entrada e adicionados no grafo.

É testada uma cor inicial para um vértice e da mesma forma é testada com a mesma cor para o próximo vértice da sequência, se for válida utiliza a mesma cor, se não for válida troca de cor.

# Metodologia: Metaheurísticas

# Metaheurística 1: GRASP

Algorithm 1: runGRASP

```
bestSolution = solucaoInicial(graph) *Cada vertice uma cor*
for k <- to MaxIter do
  newGraph <- constructivePhase(graph)
  solutionColors = newGraph.checkColor() *Checa Quantas cores o grafo esta pintado*
  *Fase de busca local utilizando o SA desenvolvido*
  solution <- SA(None, tempEnt, tempMin,alpha, SAMax, False, newGraph, solutionColors)
  bestSolution <- updateSolution()
end for
return bestSolution
```

# Metaheurística 1: GRASP

```
✓ Algorithm 2: constructivePhase
  candidatos, rcl <- []
  vizDescoloridos = {}
  ✓ for v <- graph.getAmountV do * Copia os vertices nao coloridos para serem os candidatos *
  ✓   if(verticeDescolorico) do
  |     candidatos = vertice
  |     vizDescoloridos[v] = -1
  |   end if
  end for
  ✓ if(candidatos == []) do
  |   return graph
  end if
  ✓ for v <- len(candidatos) do
  |   vizDescoloridos[v] = qntdVizinhosDescoloridos
  end for
  maior = -1
  ✓ for v <- len(candidatos) do
  ✓   if (vizDescoloridos[v] > maior) do
  |     maior = vizDescoloridos[v]
  |     rcl <- []
  |     rcl = vertice[v]
  |   end if
  |   else do
  |     rcl = vertice[[v]]
  |   end else
  ✓ end for
```



# Metaheurística 1: GRASP

Algorithm 2(Continuação): constructivePhase

```
while rcl not vazia do
  escolhido = random(rcl)
  for n <- escolhido.vizinhos do
    if (graph.vizinho.getColor()[n] == colorido) do
      listColors[n] = True *cor indisponivel*
    end if
  end for
  for c <- len(listColors) do
    Pega cor disponivel *igual a false*
    break
  end for
  index = pegaIndice(escolhido)
  graph.vertice[index].setColor(c)
  removeVizinhoRcl(rcl, escolhido)
  rcl.remove(escolhido)
  listColors[c] = true
end while
return constructivePhase(graph)
```

# Metaheurística 1: Implementação

O Grasp inicialmente gera uma solução inicial com todos os vértices coloridos de cores diferentes, que é definida como a melhor até então. Logo após em um total de 10 iterações é rodada a fase construtiva e de busca local com o Simulated Annealing. No qual a solução encontrada é comparada com a melhor até então, e se a encontrada for melhor é feita uma atualização.

Vinícius Moraes ([GitHub](#))

# Metaheurística 2: Simulated Annealing

## [Adap. [Lopes et.al]]

---

**Algoritmo 2** SIMULATED-ANNEALING( $f(\cdot)$ ,  $N(\cdot)$ ,  $\alpha$ ,  $SAMax$ ,  $T_0$ ,  $s$ )

---

```
1:  $s^* \leftarrow s$ ;  $Iter \leftarrow 0$ ;  $T \leftarrow T_0$ 
2: while ( $T > 0$ ) do
3:   while ( $Iter < SAMax$ ) do
4:      $Iter \leftarrow Iter + 1$ 
5:     Gere um vizinho qualquer  $s' \in N(s)$ 
6:      $\Delta \leftarrow f(s') - f(s)$ 
7:     if ( $\Delta < 0$ ) then
8:        $s \leftarrow s'$ 
9:       if ( $f(s') < f(s^*)$ ) then
10:         $s^* \leftarrow s'$ 
11:       end if
12:     else
13:       Tome  $x \in [0, 1]$ 
14:       if ( $x < e^{-\Delta/\tau}$ ) then
15:         $s \leftarrow s'$ 
16:       end if
17:     end if
18:   end while
19:    $T \leftarrow \alpha T$ ;  $Iter \leftarrow 0$ 
20: end while
21: return  $s^*$ 
```

# Simulated Annealing: Implementação

A solução inicial do SA é a quantidade de cores ser a mesma quantidade de nós do grafo, ou seja, todos estão coloridos com cores diferentes, este seria o pior caso.

**Estrutura de vizinha:** Laço de repetição até 5 iterações, randomizando um nó para sua cor ser modificada por outra cor válida que não seja a mesma dos vizinhos nem a dele mesmo. Caso nesse laço não ache um nó que há cores válidas para ele ser modificado, então faz um laço até a quantidade de vértices no grafo procurando por um nó que possa ser modificado por uma cor válida.

**TempInicial:** 10000 **TempFinal:** 20 **Alpha:** 0.95 **Iterações:** 1000

Jefferson Marques ([GitHub](#))

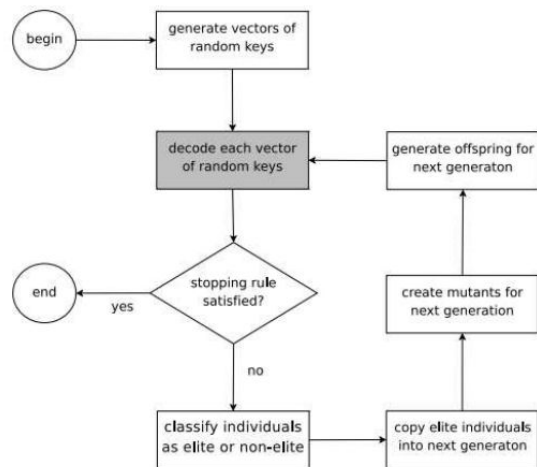
# Metaheurística 3: BRKGA [Fonte]

SILVA, Diego Mello da

RESENDE, Mauricio Elavio

## Biased Random Key Genetic Algorithm

### ■ Framework do RKGA



## Specifying a biased random-key GA

- Encoding is always done the same way, i.e. with a vector of  $N$  random-keys (parameter  $N$  must be specified)
- Decoder that takes as input a vector of  $N$  random-keys and outputs the corresponding solution of the combinatorial optimization problem and its cost (this is usually a heuristic)
- Parameters:
  - Size of population: a function of  $N$ , say  $N$  or  $2N$
  - Size of elite partition: 15-25% of population
  - Size of mutant set: 5-15% of population
  - Child inheritance probability:  $> 0.5$ , say 0.7
  - Stopping criterion: e.g. time, # generations, solution quality, # generations without improvement

# BRKGA: Implementação

---

**Algorithm 1:** runBRKGA

---

```
KeysSorting = no intervalo [0,1] * TAMPOP
Population = decoder(KeysSorting)
while ( $g \leq QTDGE$ ) do
    Population.sort()
    PopulationG = Population[0...TAMELI]
    for  $i \leftarrow TAMMUT$  do
        (parentBest, parentRandom) = getParents(KeysSorting)
        child = crossover(parentBest, parentRandom)
        PopulationG = child
    end
    for  $i \leftarrow (TAMPOP - (TAMELI + TAMMUT))$  do
        (parentBest, parentRandom) = getParents(KeysSorting)
        child = crossover(parentBest, parentRandom)
        PopulationG = child
    end
    PopulationG = decoder(KeysSorting)
    Population  $\leftarrow$  PopulationG
     $g = g + 1$ 
end
```

---

**TAMPOP:** Tamanho da população.

**QTDGE:** Quantidade de gerações.

**TAMELI:** Corresponde aos 20% da elite.

**TAMMUT:** Corresponde aos 15% dos mutantes.

# BRKGA: Implementação

---

## Algorithm 2: decoder

---

```
for  $m \leftarrow TAMPOP$  do
  collided = False
  usedColors = [0]
  KeysSorting[m].sort()
  for  $c \leftarrow KeysSorting[m]$  do
    index = KeysSorting[m].index(c)
    color = usedColors[len(usedColors)-1]
    if graph.vertex[index].color == incolor then
      | graph.vertex[index].color = color
    end
    colorNeighbor = getcolorNeighbor(c)
    if graph.vertex[index].color in colorNeighbor then
      | *Tenta reutilizar uma das cores da paleta de cores.
      | Se não der, cria uma nova cor.
      | Atribui a cor selecionada.*
      | graph.vertex[index].color = color
    end
    for  $i \leftarrow graph.AmountVertex$  do
      | *Para todos não adjacentes ao vertice atual, recebe a mesma
      | cor*
    end
  end
end
end
```

---

**TAMPOP:** Tamanho da população.

**QTDGE:** Quantidade de gerações.

**TAMELI:** Corresponde aos 20% da elite.

**TAMMUT:** Corresponde aos 15% dos mutantes.

Vinícius Araújo ([GitHub](#))



# Experimentos Realizados





# Setup Experimental

- Core i5 - 5<sup>a</sup> geração
- 4 núcleos
- 8 GB de Memória RAM

# Resultados: Heurística - Coloração Sequencial

| Instância      | Min | Média | Mediana | Max | SD | VAR | Opt? | Desvio % |
|----------------|-----|-------|---------|-----|----|-----|------|----------|
| fpsol2.i.1.col | 66  | 66    | 66      | 66  | 0  | 0   | 65   | 1.5      |
| inithx.i.1.col | 55  | 55    | 55      | 55  | 0  | 0   | 54   | 1.8      |
| inithx.i.3.col | 31  | 31    | 31      | 31  | 0  | 0   | 31   | 0        |
| le450_5a.col   | 15  | 15    | 15      | 15  | 0  | 0   | 5    | 200      |
| le450_25d.col  | 34  | 34    | 34      | 34  | 0  | 0   | 25   | 36       |
| miles250.col   | 10  | 10    | 10      | 10  | 0  | 0   | 8    | 25       |
| miles1500.col  | 74  | 74    | 74      | 74  | 0  | 0   | 73   | 1.3      |
| myciel3.col    | 4   | 4     | 4       | 4   | 0  | 0   | 4    | 0        |
| qg.order60.col | 66  | 66    | 66      | 66  | 0  | 0   | 60   | 10       |
| queen5_5.col   | 8   | 8     | 8       | 8   | 0  | 0   | 5    | 60       |

# Resultados: GRASP + Simulated Annealing

| Instância      | Min | Média | Mediana | Max | SD       | VAR      | Melhor | Desvio % |
|----------------|-----|-------|---------|-----|----------|----------|--------|----------|
| fpsol2.i.1.col | 65  | 65.38 | 65      | 66  | 0.487831 | 0.237979 | 65     | 0        |
| inithx.i.1.col | 54  | 54.9  | 55      | 55  | 0.316227 | 0.1      | 54     | 0        |
| inithx.i.3.col | 31  | 32.48 | 32      | 34  | 1.11446  | 1.24202  | 31     | 0        |
| le450_5a.col   | 13  | 16.11 | 16      | 18  | 0.897752 | 0.805959 | 5      | 160      |
| le450_25d.col  | 38  | 43.68 | 44      | 48  | 2.974233 | 8.846061 | 25     | 52       |
| miles250.col   | 8   | 8.02  | 8       | 9   | 0.140705 | 0.019797 | 8      | 0        |
| miles1500.col  | 73  | 73    | 73      | 73  | 0        | 0        | 73     | 0        |
| myciel3.col    | 4   | 4     | 4       | 4   | 0        | 0        | 4      | 0        |
| qg.order60.col | 64  | 64.2  | 64      | 65  | 0.421637 | 0.177777 | 60     | 6        |
| queen5_5.col   | 5   | 5     | 5       | 5   | 0        | 0        | 5      | 0        |

# Resultados: Simulated Annealing

| Instância      | Min | Média | Mediana | Max | SD       | VAR      | Melhor | Desvio % |
|----------------|-----|-------|---------|-----|----------|----------|--------|----------|
| fpsol2.i.1.col | 65  | 65    | 65      | 65  | 0        | 0        | 65     | 0        |
| inithx.i.1.col | 55  | 55    | 55      | 55  | 0        | 0        | 54     | 1        |
| inithx.i.3.col | 32  | 32.28 | 32      | 33  | 0.451260 | 0.203636 | 31     | 3        |
| le450_5a.col   | 16  | 16.98 | 17      | 17  | 0.140705 | 0.019797 | 5      | 220      |
| le450_25d.col  | 44  | 44.9  | 45      | 46  | 0.522233 | 0.272727 | 25     | 76       |
| miles250.col   | 8   | 8.65  | 9       | 9   | 0.479372 | 0.229798 | 8      | 0        |
| miles1500.col  | 73  | 73    | 73      | 73  | 0        | 0        | 73     | 0        |
| myciel3.col    | 4   | 4     | 4       | 4   | 0        | 0        | 4      | 0        |
| qg.order60.col | 76  | 76    | 76      | 76  | 0        | 0        | 60     | 26       |
| queen5_5.col   | 5   | 5     | 5       | 5   | 0        | 0        | 5      | 0        |

# Resultados: BRKGA

| Instância      | Min | Média | Mediana | Max | SD       | VAR      | Melhor | Desvio % |
|----------------|-----|-------|---------|-----|----------|----------|--------|----------|
| fpsol2.i.1.col | 65  | 65    | 65      | 65  | 0        | 0        | 65     | 0        |
| inithx.i.1.col | 54  | 54    | 54      | 54  | 0        | 0        | 54     | 0        |
| inithx.i.3.col | 31  | 31    | 31      | 31  | 0        | 0        | 31     | 0        |
| le450_5a.col   | 12  | 12.52 | 13      | 13  | 0.502116 | 0.252121 | 5      | 140      |
| le450_25d.col  | 25  | 26.10 | 26      | 28  | 0.758786 | 0.575757 | 25     | 0        |
| miles250.col   | 8   | 8     | 8       | 8   | 0        | 0        | 8      | 0        |
| miles1500.col  | 73  | 73    | 73      | 73  | 0        | 0        | 73     | 0        |
| myciel3.col    | 4   | 4     | 4       | 4   | 0        | 0        | 4      | 0        |
| qg.order60.col | -   | -     | -       | -   | -        | -        | 60     | -        |
| queen5_5.col   | 5   | 5.76  | 6       | 6   | 0.429234 | 0.184242 | 5      | 0        |



# Resultados: Comparativos entre as Abordagens



# Resultado Comparativo: Mínimos

| Instância      | H1 | MH1 | MH2 | MH3 |
|----------------|----|-----|-----|-----|
| fpsol2.i.1.col | 66 | 65  | 65  | 65  |
| inithx.i.1.col | 55 | 54  | 55  | 54  |
| inithx.i.3.col | 31 | 31  | 32  | 31  |
| le450_5a.col   | 15 | 13  | 16  | 12  |
| le450_25d.col  | 34 | 38  | 44  | 25  |
| miles250.col   | 10 | 8   | 8   | 8   |
| miles1500.col  | 74 | 73  | 73  | 73  |
| myciel3.col    | 4  | 4   | 4   | 4   |
| qg.order60.col | 66 | 64  | 76  | -   |
| queen5_5.col   | 8  | 5   | 5   | 5   |

# Resultado Comparativo: Medianas

| Instância      | H1 | MH1 | MH2 | MH3 |
|----------------|----|-----|-----|-----|
| fpsol2.i.1.col | 66 | 65  | 65  | 65  |
| inithx.i.1.col | 55 | 55  | 55  | 54  |
| inithx.i.3.col | 31 | 32  | 32  | 31  |
| le450_5a.col   | 15 | 16  | 17  | 13  |
| le450_25d.col  | 34 | 44  | 45  | 26  |
| miles250.col   | 10 | 8   | 9   | 8   |
| miles1500.col  | 74 | 73  | 73  | 73  |
| myciel3.col    | 4  | 4   | 4   | 4   |
| qg.order60.col | 66 | 64  | 76  | -   |
| queen5_5.col   | 8  | 5   | 5   | 6   |



# Resultado Comparativo: Desvio % (Opt)

| Instância      | H1  | MH1 | MH2 | MH3 |
|----------------|-----|-----|-----|-----|
| fpsol2.i.1.col | 1.5 | 0   | 0   | 0   |
| inithx.i.1.col | 1.8 | 0   | 1   | 0   |
| inithx.i.3.col | 0   | 0   | 3   | 0   |
| le450_5a.col   | 200 | 160 | 220 | 140 |
| le450_25d.col  | 36  | 52  | 76  | 0   |
| miles250.col   | 25  | 0   | 0   | 0   |
| miles1500.col  | 1.3 | 0   | 0   | 0   |
| myciel3.col    | 0   | 0   | 0   | 0   |
| qg.order60.col | 10  | 6   | 26  | -   |
| queen5_5.col   | 60  | 0   | 0   | 0   |



# Referências Consultadas



# Referências Consultadas

- Tião e Zé, Um algoritmo fodástico para resolver NP Completude do CV, Lugar onde foi publicado, 2050.
- Zé et al, Yet Another Algorithm to Solve Traveler Salesman Problem Using Tomatoes Farm Techniques, Journal where it was published, 2099.
- FEOFILOFF, Paulo. Coloração de Vértices. Disponível em: <[https://www.ime.usp.br/~pf/algoritmos\\_para\\_grafos/aulas/vertex-coloring.html](https://www.ime.usp.br/~pf/algoritmos_para_grafos/aulas/vertex-coloring.html)>. Acessado em: 20 de Setembro de 2019.

# Apêndices

# Apêndice A - Gráficos que fiz e não couberam lá