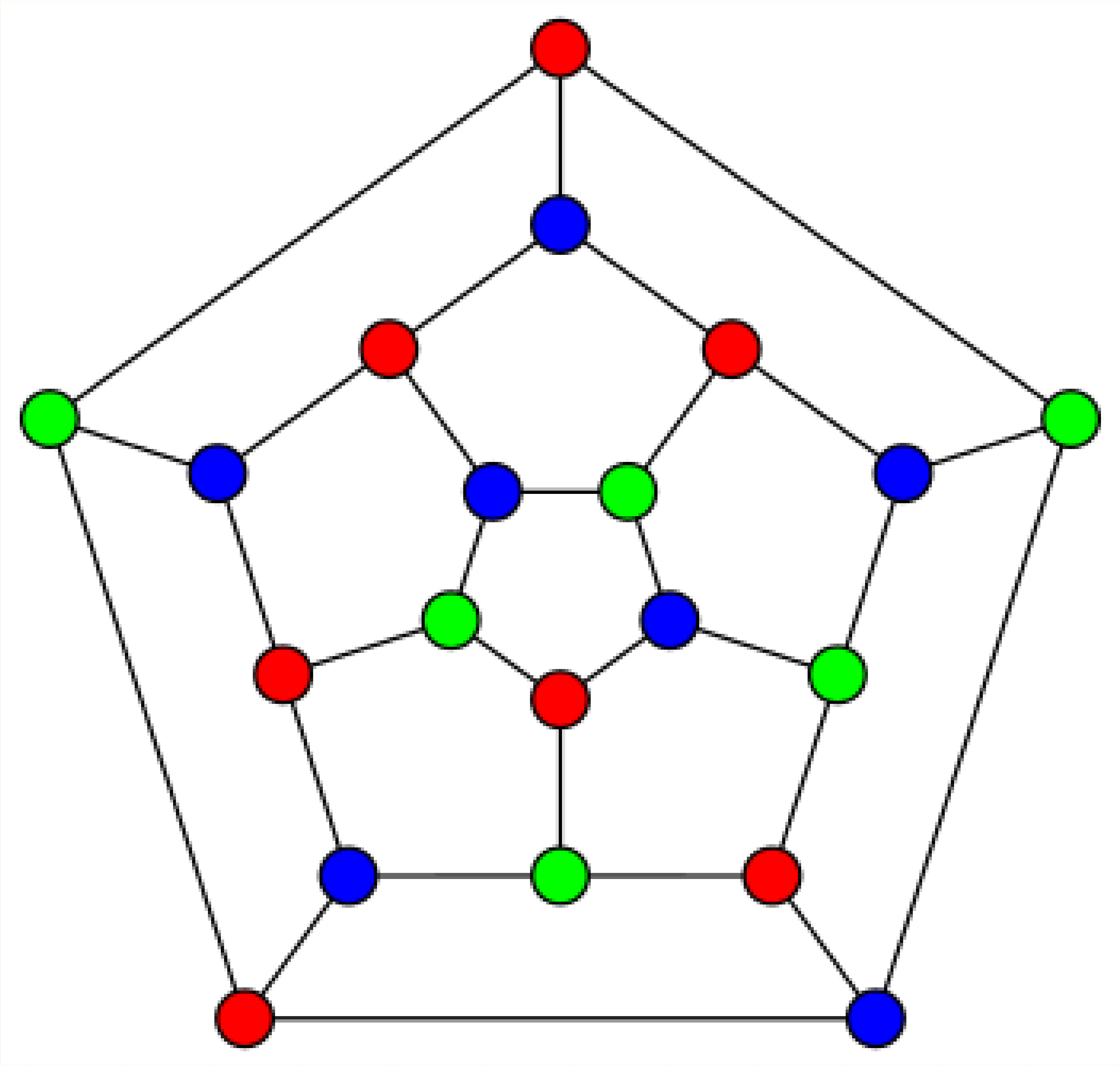
APPROXIMATE GRAPH COLORING



Problema de coloração de grafos

ALUNOS:
ANA JULIA VIEIRA PEREIRA ANDRADE DA COSTA
GABRIEL PEIXOTO MENEZES DA COSTA

PROFESSOR:
HERBERT OLIVEIRA ROCHA

CONTEXTO GERAL

Coloração de grafos é um problema clássico na Ciência da Computação;

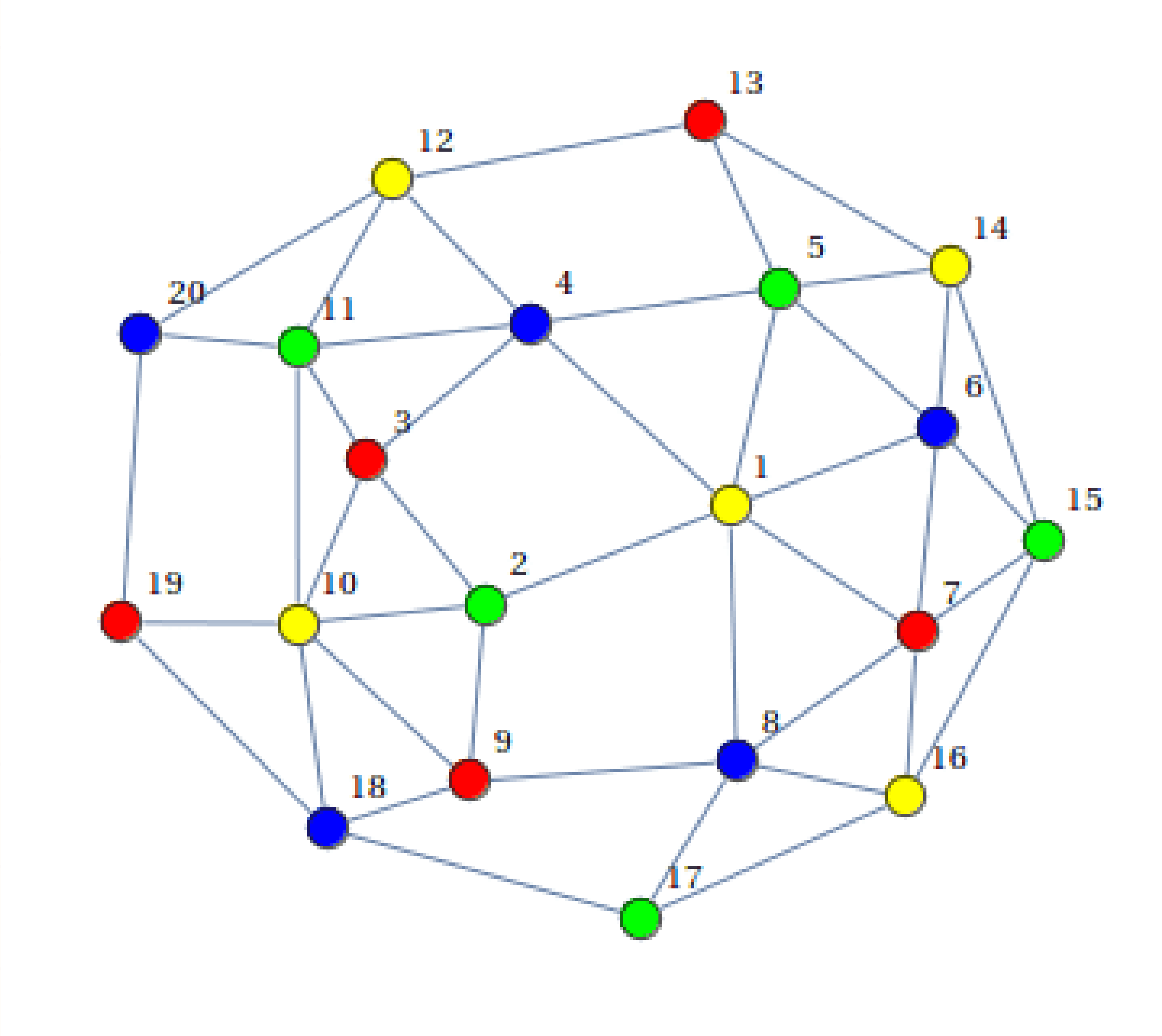
Escalonamento de tarefas, alocação de registradores e redes sem fio;

GRAFO G=(V,E)

- **V:** conjunto de vértices (elementos a serem coloridos);
- **E:** conjunto de arestas (conflitos entre vértices);

Coloração própria c: V→N tal que ∀{u,v} ∈ E, c(u) ≠ c(v);

Minimizar **χ(G)**, o número de cores distintas.



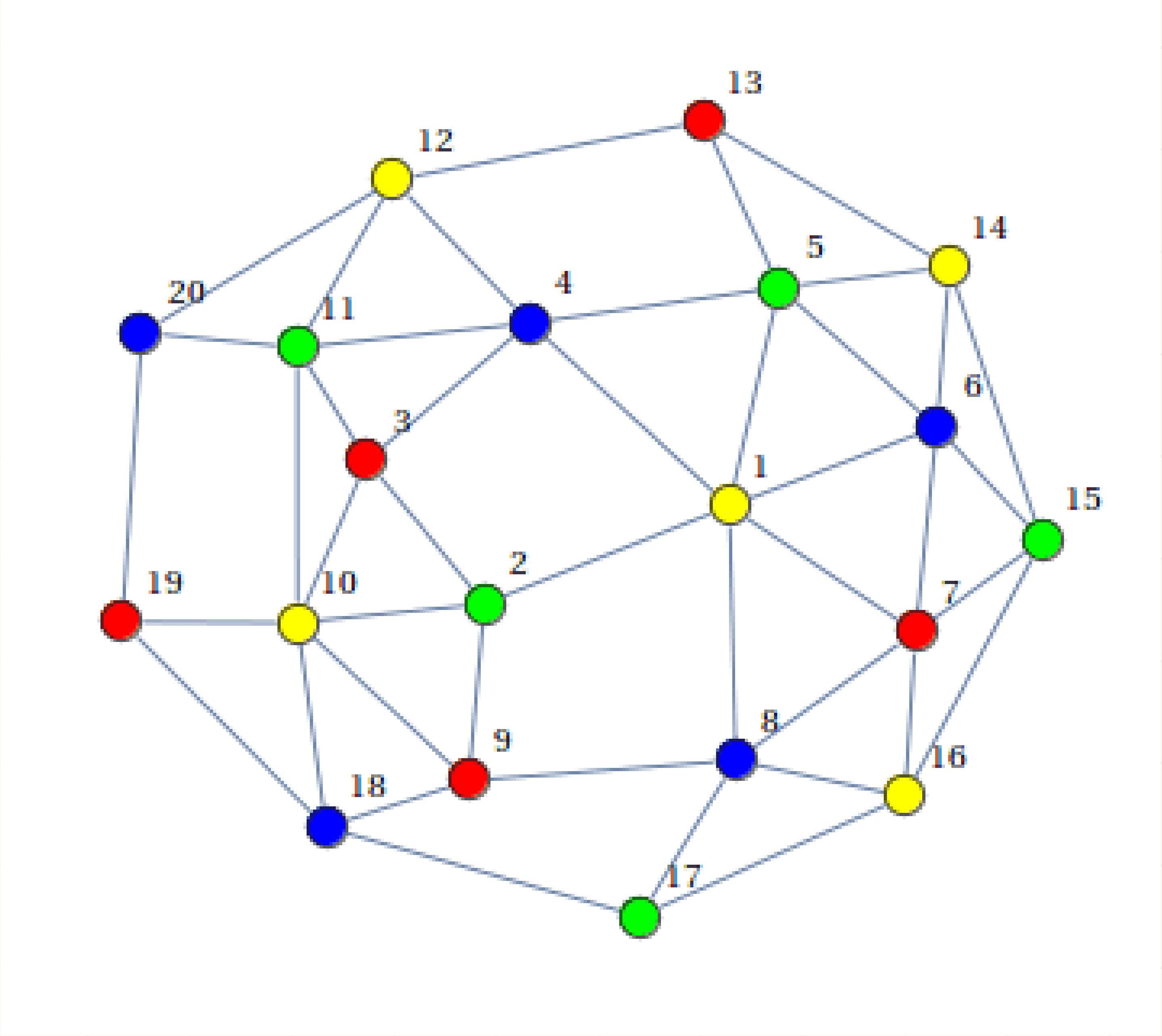
HEURÍSTICAS & COMPLEXIDADE

RLF (Recursive Largest First):

- Constrói iterativamente grandes conjuntos independentes;
- Complexidade: O(n^3);

DSATUR (Degree of Saturation):

- Prioriza vértices com maior “saturação” de cores nos vizinhos;
- Complexidade: O(n^2) (com estrutura simples de seleção).



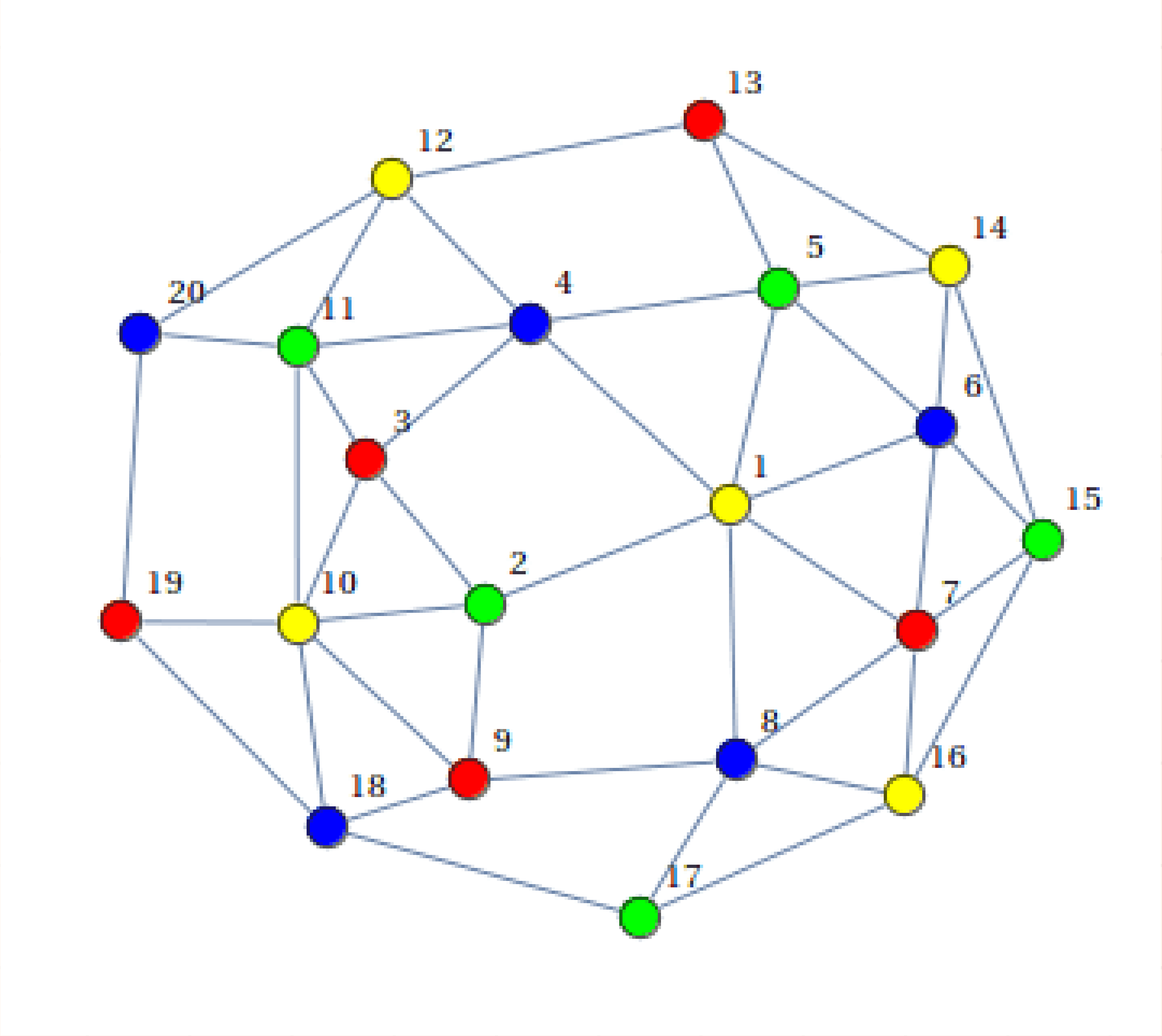
POR QUE FOI USADO OS RLF E DSATUR?

RLF:

- Produz poucos conjuntos de cores em grafos esparsos;
- Explora grandes independências locais;

DSATUR:

- Adapta-se bem a grafos densos e situações de alto grau;
- Foca em vértices mais “constrangidos”.



ALGORITMO
RLF

Passos principais:

- Escolher vértice não colorido de maior grau.
- Formar conjunto independente expandindo sem gerar conflitos.
- Colorir todo o conjunto de uma vez.
- Repetir até todos os vértices estarem coloridos.

Algorithm 1 Coloração RLF

Require: Grafo $G = (V, E)$
Ensure: Vetor de cores $colors[\,]$

$n \leftarrow |V|$
Inicialize $colors[v] \leftarrow -1,\ \forall v \in V$
Inicialize $colored[v] \leftarrow$ false, $\forall v \in V$
$color \leftarrow 0$
while existe $v \in V$ com $colored[v] =$ false **do**
  Escolha v não colorido com grau máximo no subgrafo de não coloridos
  $colors[v] \leftarrow color$
  $colored[v] \leftarrow$ true
  for all $u \in V$ com $colored[u] =$ false e $\{u, v\} \notin E$ **do**
    $canColor \leftarrow$ true
    for all $w \in V$ com $colored[w] =$ true e $colors[w] = color$ **do**
      if $\{u, w\} \in E$ **then**
        $canColor \leftarrow$ false
      end if
    end for
    if $canColor$ **then**
      $colors[u] \leftarrow color$
      $colored[u] \leftarrow$ true
    end if
  end for
  $color \leftarrow color + 1$
end while
return $colors$

ALGORITMO
DSATUR

Passos principais:

- Calcular grau e inicializar saturação = 0 para cada vértice.
- Selecionar vértice não colorido de maior saturação (desempate por grau).
- Atribuir menor cor disponível.
- Atualizar saturação dos vizinhos não coloridos.
- Repetir até colorir todo o grafo.

Algorithm 2 Coloração DSATUR

Require: Grafo $G = (V, E)$
Ensure: Vetor de cores $colors[\,]$
$n \leftarrow |V|$
Inicialize $colors[v] \leftarrow -1, \ \forall v \in V$
Para cada $v \in V$:
  $degree[v] \leftarrow |\{u : \{v, u\} \in E\}|$
  $saturation[v] \leftarrow 0$
$colored[v] \leftarrow$ false, $\forall v \in V$
for $i = 1$ **até** n **do**
  Ordene os vértices não coloridos por:
  1) maior $saturation$, 2) desempate por maior $degree$
  Selecione o primeiro vértice v não colorido
  Construa $available[0 \ldots n-1] \leftarrow$ false
  for all u vizinho de v com $colors[u] \neq -1$ **do**
    $available[\,colors[u]\,] \leftarrow$ true
  end for
  Escolha a menor cor c tal que $available[c] =$ false
  $colors[v] \leftarrow c$
  $colored[v] \leftarrow$ true
  for all u vizinho de v com $colored[u] =$ false **do**
    $saturation[u] \leftarrow saturation[u] + 1$
  end for
end for
return $colors$

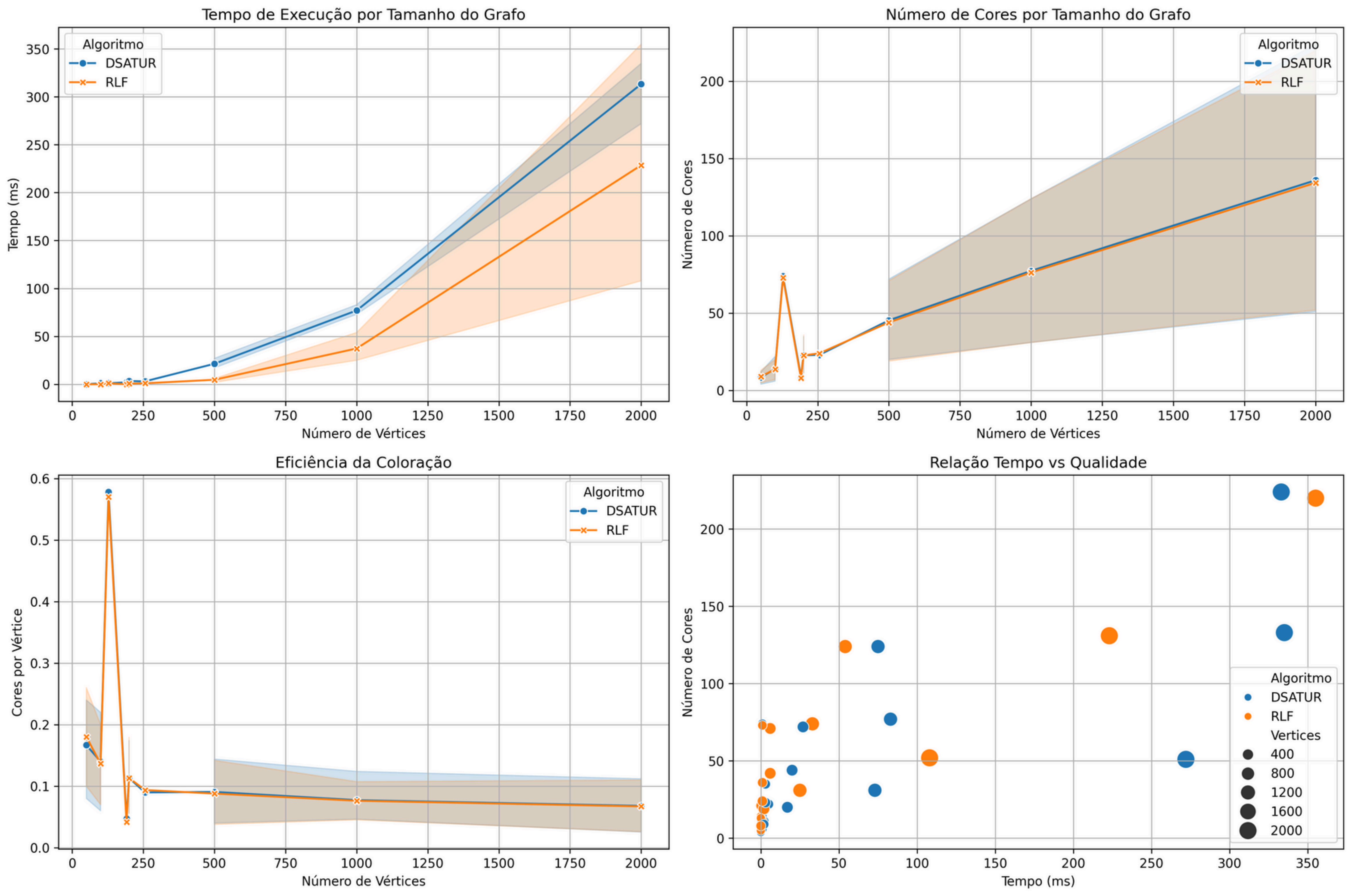
RESULTADOS EXPERIMENTAIS

BENCHMARKS CLÁSSICAS
Miles1500, myciel7, queen16_16

Instância	RLF (ms)	DSATUR (ms)
miles1500.txt	30.89	1.00
myciel7.txt	1.00	2.00
queen16_16.txt	1.00	3.00

GRAFOS ALEATÓRIOS
Grafos Aleatórios (50–2000 v., densidade 0,1–0,5)

Algoritmo	Tempo Médio (ms)
DSATUR	68.96
RLF	30.89



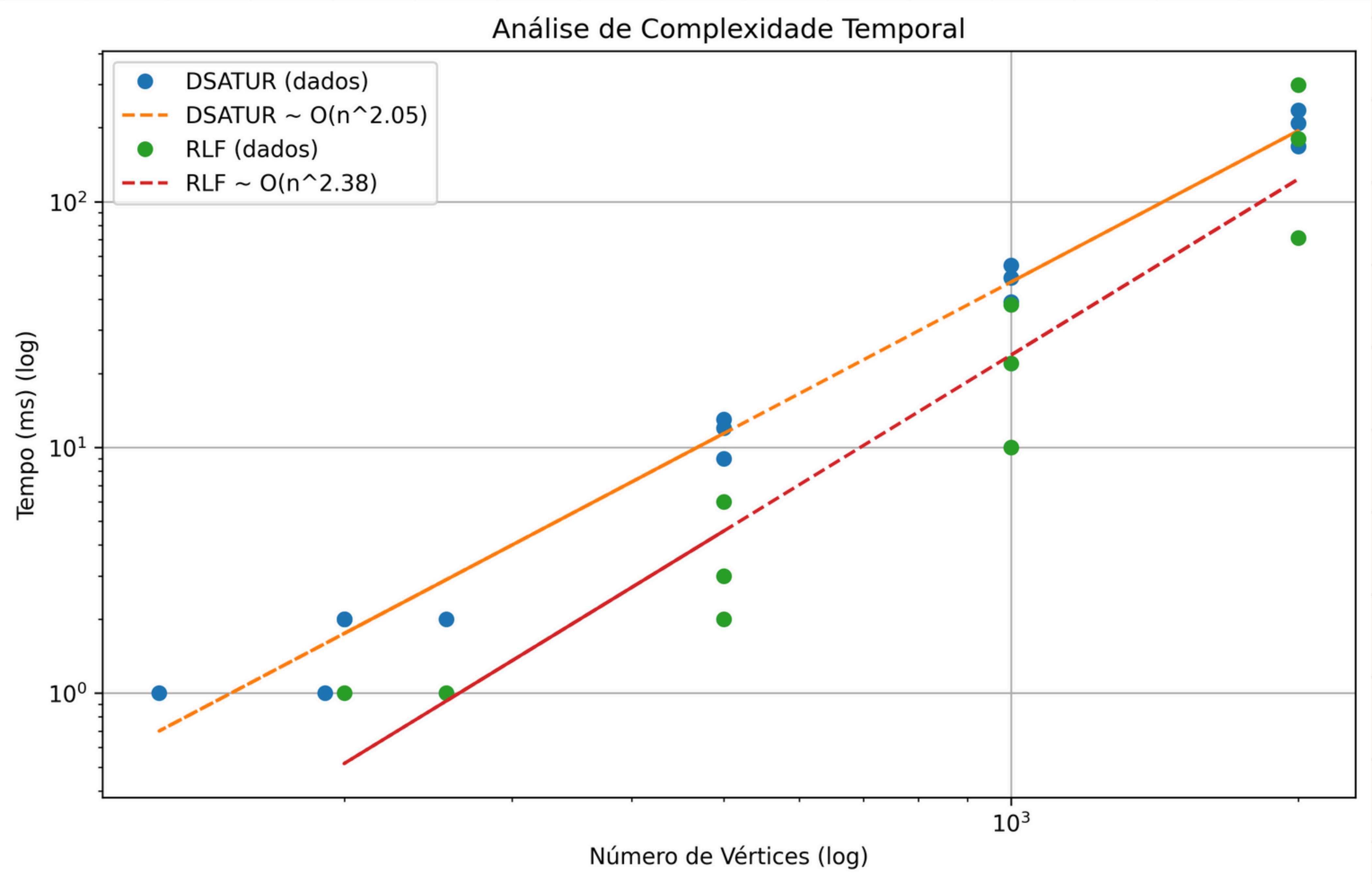
ANÁLISE DE COMPLEXIDADE

Gráfico log–log mostrando como o tempo de execução dos algoritmos cresce conforme aumenta o número de vértices no grafo.

Ajustes de curva:

- **DSATUR** segue aproximadamente $O(n^{1.99})$.
- **RLF** cresce como $O(n^{2.22})$.

Obs: Indica que o tempo de execução do DSATUR cresce aproximadamente de forma quadrática em relação ao número de vértices.
Obs2:Indica que o RLF é um pouco mais lento que o DSATUR, especialmente em grafos maiores.



CONCLUSÃO

RLF: ideal para grafos esparsos e até ~1 000 vértices (rapidez).

DSATUR: melhor para grafos densos e instâncias em larga escala (menos cores).

OBRIGADO PELA ATENÇÃO