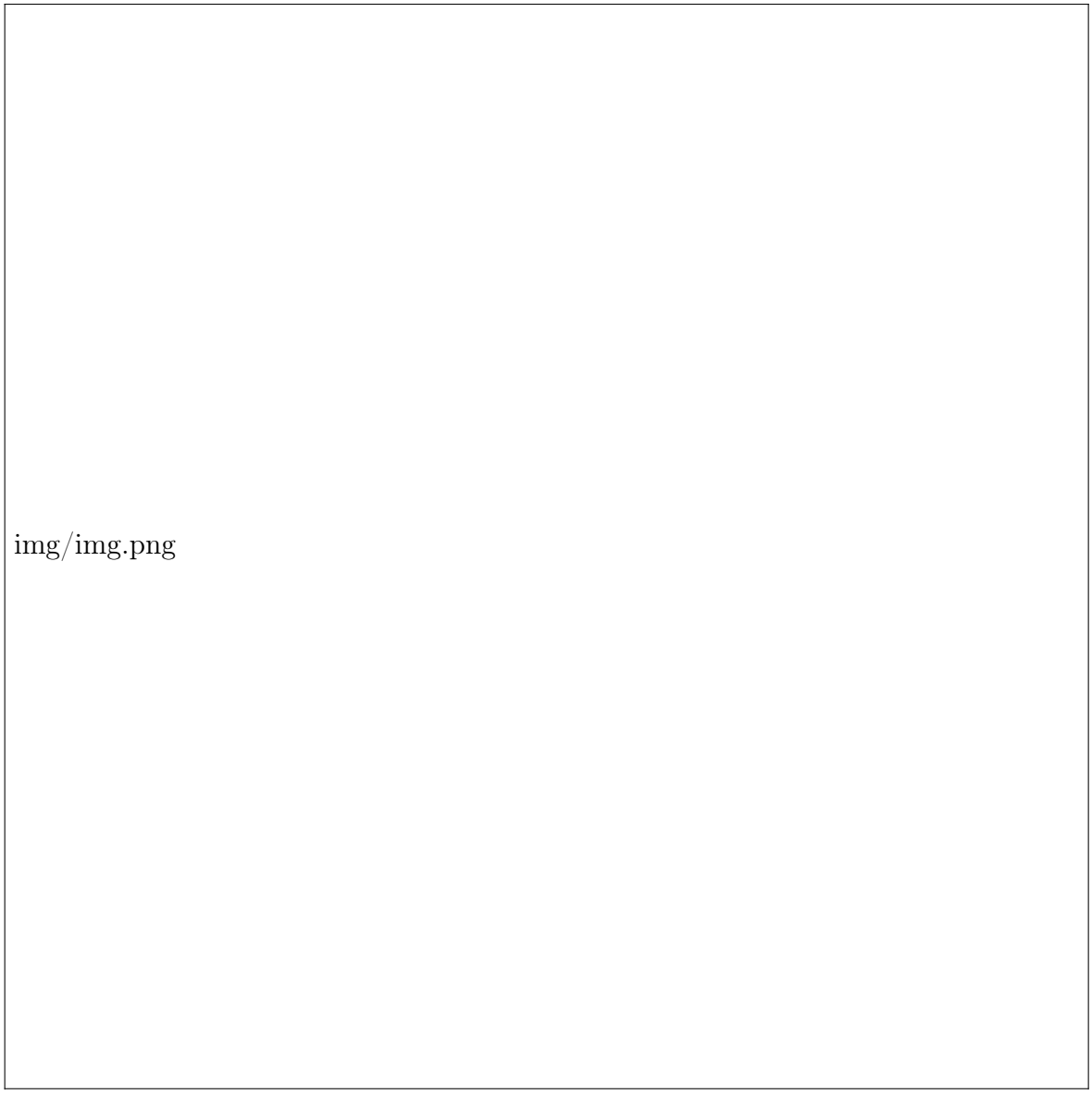
Instituto Federal do Norte de Minas Gerais
campus Montes Claros
Curso de Bacharelado em Ciência da Computação

HEURÍSTICA CONSTRUTIVA E PROBLEMA DE COLORAÇÃO DE GRAFOS

ÊNIO FILIPE E LORENA AVELINO
Orientador: Tadeu Knewitz Zubaran

Montes Claros
Novembro de 2024


img/img.png

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A coloração de grafos é uma das áreas mais estudadas na teoria dos grafos, devido à sua relevância teórica e ampla aplicabilidade prática. O problema consiste em atribuir cores aos vértices de um grafo de forma que vértices adjacentes possuam cores distintas, minimizando o número total de cores utilizadas. Esse número, conhecido como número cromático, é um parâmetro fundamental para a otimização em várias áreas (Brélaz [1979]).

A complexidade computacional do problema, classificado como NP-difícil, torna essencial o desenvolvimento de algoritmos eficientes e heurísticas inovadoras. Tais abordagens têm sido exploradas em diferentes contextos, como alocação de frequências em redes de telecomunicações, programação de horários, e design de redes.

Este artigo apresenta uma análise detalhada do problema de coloração de grafos, destacando a implementação de uma heurística baseada no algoritmo DSATUR. Além disso, discute-se o desempenho da abordagem proposta em comparação com outras técnicas como Tabu Search, GRASP e Simulated Annealing. A proposta visa avaliar tanto a qualidade das soluções quanto a eficiência computacional, contribuindo para o avanço de métodos aplicados à otimização de coloração de grafos.

1.1 Relevância

A relevância do problema de coloração de grafos se estende por diversas áreas, incluindo a alocação de recursos, planejamento de tarefas, design de redes e otimização de sistemas. Em particular, ele é amplamente aplicado em contextos como a programação de horários (onde cada vértice representa uma tarefa ou evento e as arestas indicam conflitos), a alocação de frequência em redes sem fio (com vértices representando transmissores e arestas indicando interferências) e a resolução de problemas de mapas (em que regiões adjacentes devem ser coloridas de forma diferente). Além disso, a comple-

xidade computacional associada ao problema – classificado como NP-difícil – o torna um tópico fundamental para estudos sobre algoritmos e limites computacionais.

1.2 Motivação

A motivação para a escolha deste problema reside tanto em sua profundidade teórica quanto em sua ampla aplicabilidade prática. Teoricamente, ele oferece uma rica base para explorar propriedades estruturais de grafos e desenvolver métodos algorítmicos eficientes. Na prática, sua resolução pode trazer benefícios diretos para questões cotidianas e industriais, como a gestão de recursos limitados em sistemas complexos. Além disso, a interdisciplinaridade do problema – interligando matemática, ciência da computação, e engenharia – reforça sua importância como um objeto de estudo abrangente e multifacetado. Assim, investigar o problema de coloração de grafos não apenas contribui para avanços acadêmicos, mas também para soluções inovadoras em desafios contemporâneos.

Capítulo 2

Revisão Bibliográfica

O problema de coloração de grafos tem sido amplamente estudado na literatura devido à sua relevância prática e à complexidade teórica associada. Este capítulo apresenta uma revisão detalhada dos principais trabalhos relacionados ao tema, com ênfase na importância de cada artigo para os estudos da coloração de grafos, destacando suas contribuições e limitações. O foco maior será dado ao artigo base Hertz & Werra [1987] por seu papel seminal na introdução da Busca Tabu como abordagem heurística para este problema.

O artigo de Hertz & Werra [1987] foi um marco na aplicação de metaheurísticas ao problema de coloração de grafos, apresentando a *Busca Tabu (Tabu Search, TS)* como uma alternativa eficiente para métodos clássicos como o recozimento simulado. A Busca Tabu foi proposta como uma técnica para evitar mínimos locais, utilizando uma *lista tabu* que impede o algoritmo de retornar a soluções já exploradas e critérios de aspiração para superar restrições temporárias quando movimentos promissores são identificados.

Os autores realizaram experimentos em grafos gerados aleatoriamente, com até 1000 vértices e alta densidade (≥ 0.5). O método mostrou eficiência superior ao recozimento simulado em termos de tempo computacional e qualidade das soluções, conseguindo colorações quase ótimas. A função objetivo utilizada foi a minimização de conflitos (arestas onde vértices adjacentes possuem a mesma cor), e a estratégia de vizinhança modificava a cor de vértices adjacentes.

Embora o artigo não utilize benchmarks conhecidos, como o DIMACS, ele estabelece uma base sólida para estudos posteriores. As ideias introduzidas, como a lista tabu e os critérios de aspiração, foram amplamente referenciadas e expandidas por outros autores, consolidando a TS como uma abordagem fundamental para problemas de otimização combinatória.

No trabalho de Galinier & Hao [1999], é apresentada uma abordagem híbrida que

combina a Busca Tabu com algoritmos evolutivos, resultando no *Hybrid Evolutionary Algorithm (HEA)*. Essa técnica se destaca por integrar operadores de cruzamento com estratégias de busca local, permitindo a exploração e intensificação do espaço de soluções. As instâncias utilizadas nos experimentos incluem benchmarks DIMACS, com grafos densos e heterogêneos.

O HEA mostrou resultados superiores a métodos puramente heurísticos, tanto em termos de qualidade quanto de eficiência computacional. A combinação de diversidade gerada pelos operadores de cruzamento com o refinamento das soluções pela TS foi um avanço significativo, especialmente para grafos de grande escala. Este trabalho referencia diretamente Hertz & Werra [1987], utilizando a TS como base para o refinamento local.

O artigo de Laguna & Martí [2001] introduz o *GRASP (Greedy Randomized Adaptive Search Procedure)* como uma abordagem adaptativa para coloração de grafos esparsos. Diferente de estudos anteriores que focaram em grafos densos, este trabalho avalia grafos de baixa densidade (≤ 0.2), utilizando instâncias conhecidas e geradas pelos autores.

As instâncias utilizadas incluem grafos da página de Michael Trick, como LEI (Leighton Graphs), MYC (Mycielski Transformation), REG (Register Allocation), e SGB (Stanford GraphBase). O GRASP foi eficaz na construção de soluções iniciais por meio de uma estratégia aleatorizada, refinando-as posteriormente com busca local. O método apresentou resultados superiores ao TS clássico em cenários de grafos esparsos.

O trabalho de Blöchliger & Zufferey [2008] propõe uma extensão da Busca Tabu denominada *FOO-PARTIALCOL*, que utiliza soluções parciais para resolver o problema de coloração. Esta abordagem é baseada nos fundamentos de Hertz & Werra [1987], mas introduz melhorias significativas, como a adaptação dinâmica dos parâmetros (*Tabu Search Reativo*) e a construção incremental de soluções completas a partir de parciais.

Os testes foram realizados em benchmarks DIMACS e instâncias personalizadas, abrangendo grafos de alta densidade e complexidade. O *FOO-PARTIALCOL* demonstrou ser mais eficiente que o *TABUCOL*, reduzindo custos computacionais e mantendo a qualidade das soluções. Este trabalho é uma evolução direta da TS original, destacando-se pelo tratamento de grafos grandes e desafiadores.

O artigo de Chams et al. [1987], explora a aplicação do *Simulated Annealing (SA)* ao problema de coloração de grafos. Este método é apresentado como uma alternativa baseada em heurísticas probabilísticas, que permite explorar o espaço de soluções ao escapar de mínimos locais por meio de transições controladas pela temperatura.

Os autores conduzem experimentos utilizando grafos gerados aleatoriamente, com até 1000 vértices e diferentes densidades. O algoritmo segue o paradigma do recozimento simulado, onde a probabilidade de aceitar soluções piores diminui gradualmente conforme a temperatura esfria. Os resultados indicaram que o método é eficaz para instâncias menores e médias, mas sua eficiência decresce em grafos maiores ou de alta densidade.

Embora o recozimento simulado não tenha superado a Busca Tabu em termos de eficiência computacional, ele apresentou uma abordagem inovadora para a época. Este artigo influenciou diretamente trabalhos posteriores, como os estudos de Hertz & Werra [1987], sobre TS, e foi amplamente referenciado como um ponto de partida para a integração de metaheurísticas no problema de coloração de grafos.

Tabela 2.1. Análise Comparativa Detalhada dos Artigos

Artigo	**Método**	**Instâncias Utilizadas**	**Resultados**
Hertz & Werra [1987]	Busca Tabu (TS)	Grafos aleatórios	Melhor desempenho que o SA; colorações quase ótimas em grafos densos grandes; base teórica para estudos futuros.
Galinier & Hao [1999]	HEA (Tabu + Cruzamento)	Benchmarks DIMACS	Soluções superiores às heurísticas puras; alto desempenho em grafos densos e heterogêneos.
Laguna & Martí [2001]	GRASP	LEI, MYC, REG, SGB	Melhor eficiência em grafos esparsos; bom desempenho em comparação ao TS clássico em baixa densidade.
Blöchliger & Zufferey [2008]	Reactive Tabu (FOO-PARTIALCOL)	Benchmarks DIMACS e personalizadas	Alta eficiência em grafos grandes e complexos; supera TABUCOL com custos computacionais reduzidos.
Chams et al. [1987]	Simulated Annealing (SA)	Grafos aleatórios	Eficiente em grafos menores e médios; superado pela TS em eficiência para grafos maiores ou mais densos.

O trabalho de Chams et al. [1987] introduziu o recozimento simulado como uma técnica promissora para coloração de grafos, destacando-se pela sua abordagem probabilística e capacidade de escapar de mínimos locais. Apesar de não ser tão eficiente quanto a Busca Tabu em grafos maiores, sua proposta fundamentou a evolução de metaheurísticas aplicadas ao problema, influenciando diretamente estudos subsequentes.

Ao compararmos todos os artigos, fica evidente que os métodos híbridos, como o *HEA* e a *Otimização Aritmética*, representam avanços significativos ao integrar técnicas

clássicas com heurísticas modernas. Trabalhos como o de Blöchliger & Zufferey [2008] e Hertz & Werra [1987] destacam-se pelo impacto direto na consolidação da Busca Tabu e suas extensões como ferramentas essenciais na coloração de grafos.

Tabela 2.2. Comparação Geral entre os Artigos

Aspecto	**Hertz & Werra [1987]**	**Galinier & Hao [1999]**	**Laguna & Martí [2001]**	**Blöchliger & Zufferey [2008]**	**Chams et al. [1987]**
Método	Busca Tabu	HEA (Tabu + Cruzamento)	GRASP	Reactive Tabu (FOO-PARTIALCOL)	Simulated Annealing
Instâncias	Grafos aleatórios	Benchmarks DIMACS	LEI, MYC, REG, SGB	Benchmarks DIMACS e personalizadas	Grafos aleatórios
Densidade	Alta (≥ 0.5)	Alta (≥ 0.5)	Baixa (≤ 0.2)	Alta (≥ 0.5)	Média a Alta
Desempenho	Superior ao SA	Superior às heurísticas puras	Superior ao TS clássico para grafos esparsos	Superior ao TABUCOL com custo reduzido	Eficiente para grafos menores .
Contribuição	Base da TS	Integração híbrida	Alternativa eficiente para grafos esparsos	Adaptação reativa da TS	Primeira metaheurística aplicada
Resultados	Melhor desempenho que o SA; colorações quase ótimas em grafos densos grandes; base teórica para estudos futuros.	Soluções superiores às heurísticas puras; alto desempenho em grafos densos e heterogêneos.	Melhor eficiência em grafos esparsos; bom desempenho em comparação ao TS clássico em baixa densidade.	Alta eficiência em grafos grandes e complexos; supera TABUCOL com custos computacionais reduzidos.	Eficiente em grafos menores e médios; superado pela TS em eficiência para grafos maiores ou mais densos.

Capítulo 3

Descrição da Heurística

3.1 Heurística de Coloração de Grafos

A heurística desenvolvida para este problema é inspirada no algoritmo DSatur(Brélaz [1979]), empregando saturação e grau dos vértices como critérios de escolha.

3.1.1 Metodologia

- **Inicialização das Estruturas:**
  - São criados vetores para armazenar:
    - As cores atribuídas a cada vértice (inicialmente, todos os vértices não coloridos têm valor -1).
    - A saturação de cada vértice, definida como o número de cores distintas presentes nos vizinhos.
    - O grau de cada vértice, calculado com base no número de vizinhos diretos.
- **Cálculo Inicial dos Graus:** O grau de cada vértice é computado iterativamente. Durante esta etapa, também é identificado o grau máximo do grafo, que será utilizado posteriormente para análise.
- **Processo Iterativo (N passos):**
  - O algoritmo realiza N iterações, onde N é o número total de vértices no grafo::
    - Em cada iteração, é selecionado um vértice candidato com base nos seguintes critérios, em ordem de prioridade:

- Vértices ainda não coloridos.
- Maior saturação.
- Maior grau (em caso de empate na saturação).
- Ordem de ocorrência no grafo (como último critério de desempate).

- Após a seleção, é atribuída ao vértice a menor cor disponível, ou seja, a menor cor que não esteja presente nos seus vizinhos adjacentes.
- Finalmente, os vértices vizinhos têm sua saturação incrementada.

- **Cálculo da Maior Cor Atribuída:** Após o término do processo iterativo, a maior cor atribuída a qualquer vértice é retornada como o número cromático aproximado do grafo.

3.1.2 Exemplo

Figura 3.1. Inicialização do algoritmo

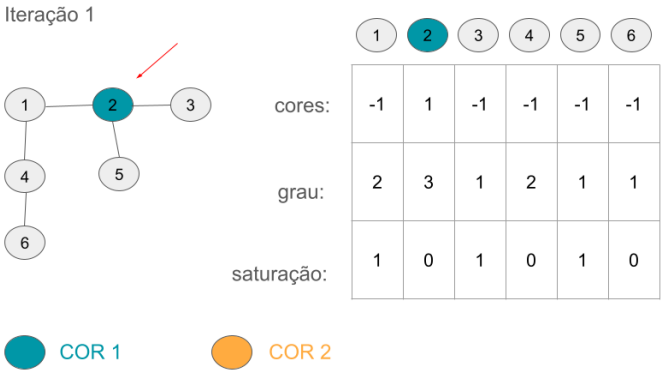


Figura 3.2. Iteracao 1

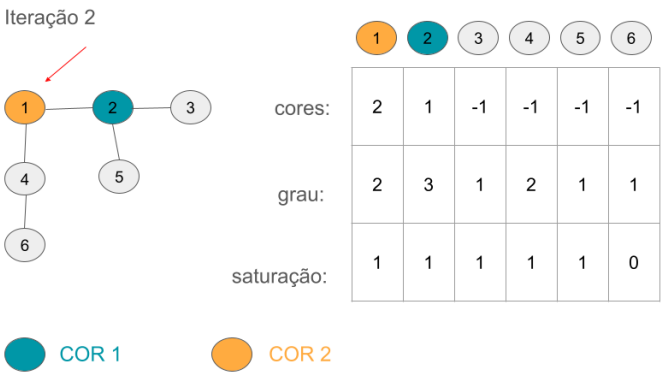


Figura 3.3. Iteracao 2

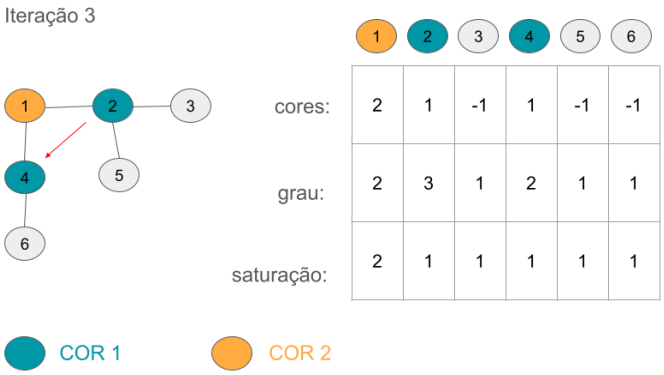


Figura 3.4. Iteracao 3

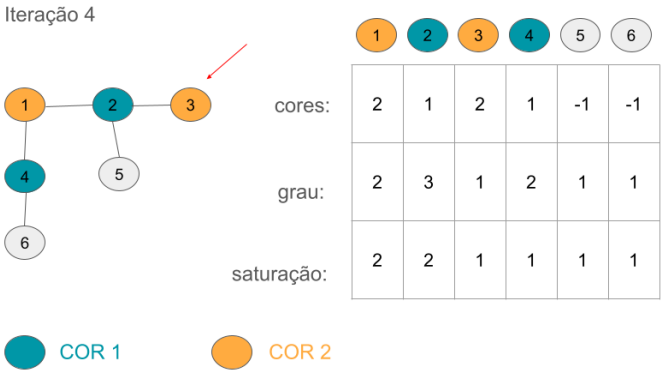


Figura 3.5. Iteracao 4

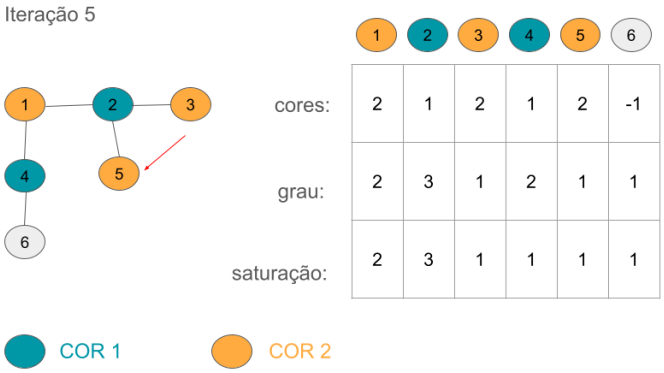


Figura 3.6. Iteracao 5

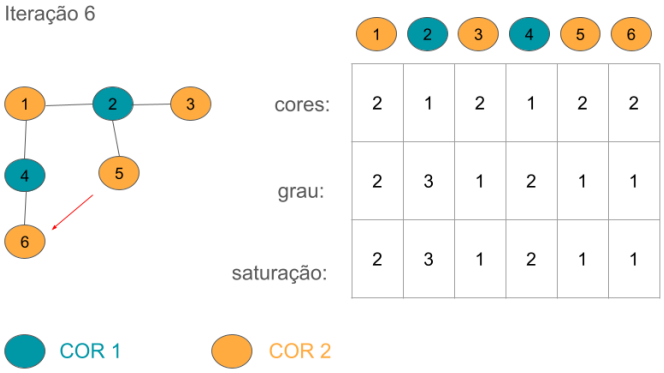


Figura 3.7. Iteracao 6

3.1.3 Código da Heurística

O código C++ abaixo descreve a implementação detalhada da heurística:

```
int getMaxColor(Graph &graph, int num_vertices)
{
    int grauMaximo = 0;
    vector<int> cores(num_vertices, -1);
    vector<int> saturacao(num_vertices, 0);
    vector<int> graus(num_vertices, 0);

    for (int i = 0; i < num_vertices; i++)
    {
        graus[i] = graph.get_neighbors(i).size();
        if (graus[i] > grauMaximo)
        {
            grauMaximo = graus[i];
        }
    }

    for (int passada = 0; passada < num_vertices; passada++)
    {
        int melhorVertice = -1;
        int maiorSaturacao = -1;
        int maiorGrau = -1;

        for (int vertice = 0; vertice < num_vertices; vertice++)
        {
            if (cores[vertice] == -1)
            {
                if (saturacao[vertice] > maiorSaturacao || (
                    saturacao[vertice] == maiorSaturacao && graus[
                    vertice] > maiorGrau))
                {
                    melhorVertice = vertice;
                    maiorSaturacao = saturacao[vertice];
                    maiorGrau = graus[vertice];
                }
            }
        }

```

```
36          int menorCor = 1;
37          set<int> coresIndisponiveis;
38          for (CoupleVertice vizinho : graph.get_neighbors(
                melhorVertice))
39          {
40              if (cores[vizinho.vertice2] != -1)
41              {
42                  coresIndisponiveis.insert(cores[vizinho.vertice2]);
43              }
44          }
45
46          while (coresIndisponiveis.count(menorCor) > 0)
47          {
48              menorCor++;
49          }
50
51          cores[melhorVertice] = menorCor;
52          for (CoupleVertice vizinho : graph.get_neighbors(
                melhorVertice))
53          {
54              saturacao[vizinho.vertice2]++;
55          }
56      }
57
58      int maxCor = 0;
59      for (int i = 0; i < num_vertices; i++)
60      {
61          if (cores[i] > maxCor)
62          {
63              maxCor = cores[i];
64          }
65      }
66
67      return maxCor;
68  }
```

Listing 3.1. Heurística Construtiva

3.1.4 Resultados Obtidos

O resultado final da heurística é uma coloração válida do grafo, aproximando o número cromático do grafo com eficiência e baixa complexidade computacional.

3.2 Algoritmo Guloso para Limite Inferior (Clique Aproximado)

Um clique de um grafo é um subconjunto de vértices no qual todos os pares de vértices estão conectados. A heurística gulosa apresentada aqui é utilizada para estimar um limite inferior do maior clique, priorizando a simplicidade e velocidade de execução.

3.2.1 Metodologia

- **Ordenação por Grau:** Os vértices do grafo são ordenados em ordem decrescente de grau (número de conexões diretas).
- **Construção do Clique:** Inicializa-se um conjunto vazio para armazenar os vértices do clique.
  - Cada vértice, começando pelos de maior grau, é avaliado:
    - Se o vértice é adjacente a todos os vértices já presentes no clique, ele é adicionado ao conjunto.
- **Retorno do Resultado:** Após a avaliação de todos os vértices, o conjunto resultante é retornado como o clique encontrado.

3.2.2 Código do Algoritmo

A implementação em C++ é dada como segue:

```
1 vector<int> Problem::calcClique(Graph &graph)
2 {
3     int n = graph.get_num_vertices();
4     vector<int> vertices(n);
5     for (int i = 0; i < n; ++i)
6         vertices[i] = i;
7 
8     sort(vertices.begin(), vertices.end(), [&](int a, int b)
9          { return graph.get_neighbors(a).size() > graph.
               get_neighbors(b).size(); });
```

```
10
11      vector<int> clique;
12      for (int v : vertices)
13      {
14          bool canAdd = true;
15          for (int u : clique)
16          {
17              if (!graph.ehVizinho(u, v))
18              {
19                  canAdd = false;
20                  break;
21              }
22          }
23          if (canAdd)
24              clique.push_back(v);
25      }
26      return clique;
27  }
```

Listing 3.2. Algoritmo Guloso

3.2.3 Limitações

Embora eficiente, este algoritmo não garante a obtenção do maior clique, pois utiliza critérios gulosos. No entanto, sua simplicidade o torna apropriado para gerar limites inferiores rápidos em aplicações práticas.

Capítulo 4

Resultados

Nesta seção, são analisados os resultados experimentais dos algoritmos apresentados, com foco na *Solução Heurística (SH)* e na sua capacidade de minimizar a quantidade de cores utilizadas na coloração dos grafos. A análise também considera a estabilidade das soluções, onde **maiores desvios indicam menor consistência**. Os algoritmos comparados incluem *Tabu Search (TS)*, *Simulated Annealing (SA)*, *GRASP*, *Hybrid Coloring Algorithm (HCA)* e métodos parciais, avaliados em diferentes instâncias de grafos. O objetivo principal é analisar como cada algoritmo reduz o número de cores, destacando o desempenho do SH.

4.1 Comparação entre TS, SA e SH

Tabela 4.1. Comparação entre Tabu Search & Simulated Annealing.

G	**I**	**N**	**MC**	**SA**		**TS**		**SH**	
				S	D	S	D	S	D
100 vértices	20	100	8	16	110.5	16	110.5	20	161.8
300 vértices	10	300	9	36	283.0	35	272.3	47	404.3
500 vértices	5	500	9	54	487.0	51	454.3	71	673.9
1000 vértices	1	1000	11	98	790.9	85	672.7	124	1027.3

Legenda: **G** = Grupo; **I** = Instâncias; **N** = Número de vértices; **MC** = Média Clique; **SA** = Simulated Annealing; **TS** = Tabu Search; **SH** = Solução Heurística; **S** = Solução; **D** = Desvio.

Em todas as instâncias, o SH utilizou **mais cores** que TS e SA. Por exemplo, para grafos com 1000 vértices, o SH utilizou **46% mais cores** que o TS (124 vs. 85). O desvio do SH foi significativamente maior, com **50% de variação a mais** em relação ao TS em instâncias maiores, indicando menor estabilidade.

4.2 Comparação entre GRASP e SH

Tabela 4.2. Comparação GRASP

G	I	N	M	MC	GRASP		SH	
					S	D	S	D
LEI	13	450	10787	1	16.5	1550.0	19.5	1846.2
MYC	5	73.4	688.4	1	6	500.0	4.2	320.0
REG	14	362.1	7608.1	1	37.4	3640.0	10.9	985.7
SGB	24	135.9	3745.25	9.7	22.7	134.8	19.5	101.3

Legenda: **G** = Grupo; **I** = Instâncias; **N** = Número de vértices; **M** = Arestas; **MC** = Média Clique; **GRASP** = Solução obtida pelo GRASP; **SH** = Solução Heurística; **S** = Solução; **D** = Desvio.

No grupo LEI, o SH utilizou **mais cores** que o GRASP, com um desvio **19% maior**, evidenciando sua menor consistência. No grupo MYC, o SH obteve uma solução com menos cores e **desvio 36% menor**, destacando bom desempenho para grafos esparsos. Para o grupo REG, o GRASP foi superior, utilizando menos cores com desvio menor.

4.3 Comparação entre HCA e SH

Tabela 4.3. Comparação HCA

G	I	N	M	MC	HCA		SH	
					S	D	S	D
DSJC250.5	1	250	15668	1	28	2700.0	38	3700.0
DSJC500.5	1	500	62624	1	48	4700.0	67	6600.0
DSJC1000.5	1	1000	249826	1	83	8200.0	118	11700.0
le450_15c	1	450	16680	1	15	1400.0	25	2400.0
le450_25c	1	450	17343	1	26	2500.0	29	2800.0
flat300_28_0.col	1	300	21695	1	31	3000.0	42	4100.0
flat1000_76_0.col	1	1000	246708	1	83	8200.0	119	11800.0
DSJC1000.1	1	1000	49629	1	20	1900.0	29	2800.0
DSJC1000.9	1	1000	449449	1	224	22300.0	276	27500.0

Legenda: **G** = Grupo; **I** = Instâncias; **N** = Número de vértices; **M** = Arestas; **MC** = Média Clique; **HCA** = Hybrid Coloring Algorithm; **SH** = Solução Heurística; **S** = Solução; **D** = Desvio.

Em todas as instâncias, o HCA utilizou **29% menos cores em média** do que o SH. O desvio do SH foi consistentemente maior, como em DSJC1000.5, onde o desvio do SH foi **42% superior** ao do HCA. Para problemas de minimização de cores, o HCA demonstrou ser a solução mais eficiente.

4.4 Comparação entre Partial, TS e SH

Tabela 4.4. Comparação entre Partial & Tabu.

G	**I**	**N**	**M**	**MC**	**PARTIAL**				**TABU**				**SH**	
					FOO		DYN		FOO		DYN			
					S	**D**	**S**	**D**	**S**	**D**	**S**	**D**	**S**	**D**
DSJC1000.1	1	1000	49629	1	21	2000.0	**20**	**1900**	21	2000	**20**	**1900**	29	2800.0
DSJC1000.5	1	1000	249826	1	89	8800.0	89	8800	89	8800	89	8800	118	11700.0
DSJC1000.9	1	1000	449449	1	228	22700.0	228	22700	227	22600	227	22600	276	27500.0
DSJC500.1	1	500	12458	1	12	1100.0	12	1100	12	1100	12	1100	17	1600.0
DSJC500.5	1	500	62624	1	50	4900.0	49	4800	49	4800	49	4800	67	6600.0
DSJC500.9	1	500	112437	1	128	12700.0	127	12600	127	12600	127	12600	150	14900.0
DSJR500.1c	1	500	121275	1	85	8400.0	85	8400	85	8400	85	8400	119	11800.0
DSJR500.5	1	500	58862	1	128	12700.0	126	12500	128	12700	126	12500	83	8200.0
R1000.1c	1	1000	485090	1	98	9700.0	99	9800	98	9700	98	9700	167	16600.0
R1000.5	1	1000	238267	1	252	25100.0	249	24800	254	25300	249	24800	160	15900.0
R250.1c	1	250	30227	1	64	6300.0	64	6300	64	6300	66	6500	65	6400.0
R250.5	1	250	14849	1	67	6600.0	66	6500	67	6600	67	6600	38	3700.0
flat1000_50_0	1	1000	245000	1	50	4900.0	50	4900	73	7200	50	4900	121	12000.0
flat1000_60_0	1	1000	245830	1	60	5900.0	60	5900	79	7800	60	5900	118	11700.0
flat1000_76_0	1	1000	246708	1	88	8700.0	88	8700	87	8600	88	8700	119	11800.0
flat300_28_0	1	300	21695	1	28	2700.0	28	2700	29	2800	31	3000	42	4100.0
le450_15c	1	450	16680	1	15	1400.0	15	1400	15	1400	16	1500	25	2400.0
le450_15d	1	450	16750	1	15	1400.0	15	1400	15	1400	15	1400	26	2500.0
le450_25c	1	450	17343	1	27	2600.0	27	2600	26	2500	26	2500	29	2800.0
le450_25d	1	450	17425	1	27	2600.0	27	2600	27	2600	27	2600	28	2700.0

Legenda: **G** = Grupo; **I** = Instâncias; **N** = Número de vértices; **M** = Arestas; **MC** = Média Clique; **PARTIAL** = Algoritmos Parciais; **FOO** = Fixed Order of Operations; **DYN** = Dynamic Operations; **TABU** = Busca Tabu; **SH** = Solução Heurística; **S** = Solução; **D** = Desvio.

O SH utilizou consistentemente mais cores que os métodos Partial e Tabu, com desvios consideravelmente maiores. Em DSJC500.5, o SH usou **37% mais cores** e teve desvio **37.5% maior** que os concorrentes. Na instância le450_15c, o SH foi significativamente pior, utilizando **56% mais cores** que o Partial.

4.5 Desempenho do SH

O SH mostrou limitações claras no contexto da minimização de cores, utilizando mais cores que os algoritmos comparados em todas as instâncias. Apesar de ser uma abordagem promissora para a exploração de grandes cliques (em outras análises), sua aplicação para coloração de grafos é ineficiente.

O SH apresentou altos desvios em todas as comparações, evidenciando resultados menos consistentes. Métodos como Tabu Search e HCA, com desvios significativamente menores, são mais confiáveis para o objetivo de minimizar cores.

Para problemas cujo foco principal é a minimização de cores, algoritmos como **Tabu Search**, **GRASP** e **HCA** superam o SH, oferecendo soluções de maior qualidade e estabilidade. O SH pode ser útil como uma abordagem híbrida, mas não se destaca como solução independente eficiente para esse objetivo.

Capítulo 5

Conclusão

Embora o *Solução Heurística (SH)* tenha demonstrado potencial em outras métricas, ele mostrou-se ineficiente para o objetivo de **minimizar cores**, apresentando resultados inferiores em qualidade e estabilidade. Métodos como *Tabu Search*, *GRASP* e *HCA* são preferíveis para coloração de grafos devido à sua consistência e eficiência, enquanto o SH pode ser explorado em cenários de alta variação ou como suporte em algoritmos híbridos.

Referências Bibliográficas

Blöchliger, I. & Zufferey, N. (2008). A graph coloring heuristic using partial solutions and a reactive tabu scheme. *Computers & Operations Research*, 35(3):960–975.

Brélaz, D. (1979). New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256.

Chams, M.; Hertz, A. & De Werra, D. (1987). Some experiments with simulated annealing for coloring graphs. *European Journal of Operational Research*, 32(2):260–266.

Galinier, P. & Hao, J.-K. (1999). Hybrid evolutionary algorithms for graph coloring. *Journal of combinatorial optimization*, 3:379–397.

Hertz, A. & Werra, D. d. (1987). Using tabu search techniques for graph coloring. *Computing*, 39(4):345–351.

Laguna, M. & Martí, R. (2001). A grasp for coloring sparse graphs. *Computational optimization and applications*, 19:165–178.