Universidade Federal do Rio Grande do Norte – UFRN
Centro de Ensino Superior do Seridó – CERES
Departamento de Computação e Tecnologia – DCT
Bacharelado em Sistemas de Informação – BSI

Meta-Heurísticas para o Problema de Coloração de Grafos

Mosiah Adam Maria de Araújo

Orientador: João Paulo de Souza Medeiros

Trabalho de Disciplina apresentado ao Curso de Bacharelado em Sistemas de Informação como parte dos requisitos para obtenção da aprovação no componente curricular BSI2603 (Algoritmos Experimentais).

Laboratório de Elementos do Processamento da Informação – LabEPI
Caicó, RN, 18 de dezembro de 2025

Resumo

Este trabalho apresenta...

Palavras-chave: Primeira palavra chave; Segunda palavra chave; Terceira palavra chave.

Abstract

This document presents...

Keywords: First keyword; Second keyword; Third keyword.

Sumário

Lista de Algoritmos

Lista de Definições

Lista de Figuras

Glossário

Acrônimos

BFS ..... *Breadth-First Search*
RLF ..... *Recursive-Largest-First*
ABC ..... *Artificial Bee Colony*

1. Introdução

"If knowledge can create problems,
it is not through ignorance that we can solve them."
Isaac Asimov

O problema de coloração de grafos já é um desafio conhecido há bastante tempo. Tal desafio trata-se de descobrir quantas cores são necessárias para colorir um grafo, de forma que vértices ligados por uma mesma aresta não compartilhem as mesmas cores. A princípio, pode parecer uma questão simples e sem relevância prática. Todavia, diversos problemas do mundo real podem ser modelados como grafos e resolvidos através da coloração dessas estruturas, como a alocação de registradores e conflitos de agenda.

Dentre essas aplicações, a alocação de registradores é um caso de interesse particular na área de Ciência da Computação. Refere-se à dificuldade de definir em quais locais os valores usados pelo programa vão ser armazenados. Esse impasse ocorre pois as possibilidades de armazenamento ficam entre a memória e os registradores; estes últimos possuem uma grande velocidade, mas são mais limitados em capacidade e número TAVARES (2011). Como o acesso a memória principal é muito mais custoso em tempo do que aos registradores, torna-se imprescindível alocar com sabedoria os valores do programa.

Para resolver esse impasse, é possível transformar tal problema em um grafo de interferência Chaitin et al. (1981). Os vértices do grafo podem representar variáveis de um programa e suas arestas seriam a interferência entre esses nós, simbolizando que ambas variáveis estão "vivas" ao mesmo tempo. Com isso, o grafo pode ser colorido para achar um número k de registradores que satisfaça a condição da coloração. O menor k possível é conhecido como número cromático Švarcmajer et al. (2025).

Entretanto, colorir um grafo com $k \geq 3$ ou encontrar o seu número cromático é uma tarefa que não pode ser resolvida em tempo polinomial, sendo classificada como *NP-complete* Irving (1983). Isso significa dizer que a solução determinística desse problema não pode ser dada em tempo hábil à medida que a ordem do grafo aumenta. Num contexto de alocação de registradores, com vários valores, obter uma solução exata torna-se inviável.

Tendo em vista essa inviabilidade, torna-se necessário recorrer a soluções aproximadas. Assim, o presente trabalho tem como primeiro objetivo estudar soluções aproximativas de trabalhos relacionados. Já como principal tarefa, o trabalho mostrará a implementação e análise de um algoritmo genético e um algoritmo de colônia de abelhas para o problema de coloração de grafos.

1.1 Motivação

A motivação do trabalho nasce da atividade de alocar registradores no processo de compilação de um programa. Minimizar o acesso a memória principal otimiza o programa

ao reduzir consideravelmente o seu tempo de execução. Como esse processo é feito a partir da coloração de um grafo de interferência, surge a necessida de estudar e implementar algoritmos e heurísticas capazes de lidar com a natureza NP da coloração em tempo hábil. Como as soluções dadas por esses algoritmos são aproximativas, é preciso avaliar quais algoritmos tendem a ter um desempenho mais próximo do ideal.

1.2 Objetivos

Geral

O objetivo geral do trabalho é desenvolver e comparar duas meta-heurísticas para o problema de coloração de grafo: algoritmo genético e algoritmo de colônia de abelhas. Ao implementar soluções para a coloração de grafo, possíveis novas otimizações podem ser aplicadas a alocação de registrador.

Específicos

1. Estudar algoritmos e trabalhos relacionados na temática de coloração de grafos;
2. Desenvolver algoritmo genético para o problema;
3. Desenvolver algoritmo de colônia de abelhas para o problema;
4. Comparar algoritmos sob tempo de execução e qualidade das respostas.

1.3 Trabalhos relacionados

Essa seção está dividida por algoritmos. Cada parágrafo trata de um trabalho diferente que aborda uma solução específica para quando um grafo não pode ser colorido com menos de três cores.

O trabalho de Klotz (2002) aborda alguns algoritmos determinísticos para o problema. Dentre eles, existem os algoritmos de contração. Essas soluções se baseiam na ação de contrair as arestas de um grafo, ou seja: remover uma aresta, de forma que os dois vértices que formavam ela se tornem um novo vértice, unindo as adjacências de cada vértice. O algoritmo dessa natureza com melhor resultado mostrado por Klotz (2002) é o RLF (*Recursive-Largest-First*), que possui complexidade $O(|V|^3)$.

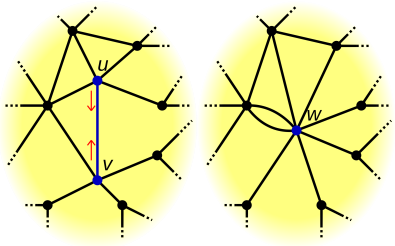

Figura 1.1: Contração da aresta de um grafo

A meta-heurística de algoritmo genético surge como uma alternativa ao uso de soluções determinísticas. Esse tipo heurística simula o processo de seleção natural, cruzando soluções do problema para gerar soluções melhores. Malhotra et al. (2024) investiga o

uso dessa abordagem, focando em gerar soluções válidas e que minimizem o número de cores necessárias para colorir o grafo.

Os estudos de Chen e Kanoh (2016) e de Fister Jr et al. (2012) aplicam o algoritmo de *Artificial Bee Colony* (ABC), conhecido como Colônia Artificial de Abelhas ou somente Colônia de Abelhas. Essa meta-heurística foi desenvolvida com base no comportamento de busca por alimento das abelhas melíferas para problemas de otimização Karaboga (2010). Fister Jr et al. (2012) usam uma abordagem híbrida para esse algoritmo ao mesclá-lo com uma heurística de busca local de passeio aleatório. Já Chen e Kanoh (2016) propõe um algoritmo não híbrido para problemas de natureza discreta.

1.4 Organização do trabalho

O trabalho está organizado nos seguintes capítulos: Introdução, Levantamento Bibliográfico, Desenvolvimento e Conclusões.

A introdução serve para contextualizar o problema e mostrar a importância da coloração de grafos, em especial na área de compiladores. Também define a motivação do trabalho e seus objetivos, gerais e específicos. Comtempla ainda uma revisão rápida por trabalhos relacionados, focando que algoritmos e heurísticas para soluções aproximativas.

O capítulo de levantamento bibilográfico se desbruça nos fundamentos da pesquisa. Aqui, o principal foco é fornecer o arcabouço para se trabalhar com grafos, coloração de grafos, algoritmos genéticos e algoritmos ABC. Além disso, é fundamentado melhor o contexto de alocação de registradores para evidenciar a importância desse estudo. Esse capítulo se divide em: introdução, metodologia e cronograma.

O desenvolvimento é a parte prática do estudo. Aqui são mostrados os algoritmos desenvolvidos para o problema e a experimentação realizada, comparando a qualidade das respostas de cada solução. A organização ocorre através de: meta-heurísticas, experimentos e considerações.

Por fim, o estudo é encerrado com o capítulo de conclusões. Essa parte discute os resultados da experimentação e elucida o leitor acerca de trabalhos futuros na área.

2. Levantamento bibliográfico

"We can only see a short distance ahead,
but we can see plenty there that needs to be done."
Alan Mathison Turing

2.1 Introdução

Definição 2.1 (Coloração de grafo)**.** (Irving, 1983) A coloração dos vértices de um grafo $G = (V, E)$ é uma função $F : V \mapsto \mathbb{N}$ em que vértices adjacentes possuem cores distintas em N; dado duas arestas $uv \in E$, significa dizer que $F(u) \neq F(v)$. O número cromático $\chi(G)$ é o menor número de cores para colorir G. O grafo será *k-cromático* se $\chi(G) = k$ e k-colorível se $\chi(G) \leq k$. Uma classe de cores de uma coloração F possui todos os vértices da mesma cor.

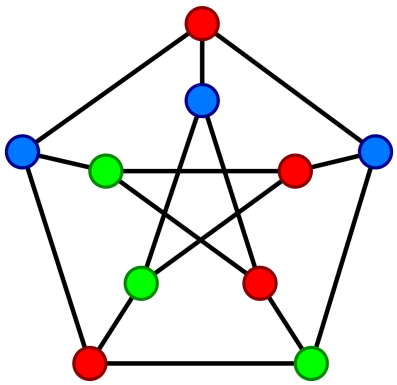

Figura 2.1: Coloração de um grafo

□

Definição 2.2 (Alocação de registrador)**.** (Chaitin, 1982) A alocação de registrador é uma etapa que entre a fase de otimização e a de geração do código final em *assembly* no processo de compilação de um programa. Na fase de otimização, se assume que a máquina alvo do programa possui um número ilimitado de registradores, visando manter os dados nos registradores e eliminar acessos a memória. Quando o processo de alocação de registradores se inicia, os registradores hipotéticos assumidos na fase anterior serão mapeados para os registradores reais da CPU. Caso os registradores da CPU não sejam o suficiente para o programa, será preciso realizar *spill code*, o processo de mover e carregar os dados da memória. Esse processo é modelado como a coloração de um grafo de interferência,

onde as cores são registradores, os vértices valores e as arestas simbolizam o tempo de vida.

□

Definição 2.3 (Algoritmo Genético)**.** (Goldbarg e Luna, 2005) Algoritmo genético é uma meta-heurística baseada na troca de genes e seleção natural, muito usada em problemas de otimização combinatória. Dado um conjunto inicial de soluções de um problema, denominado população, são retirados os indíviduos mais aptos. Essa métrica é definida por uma função de adaptabilidade, conhecida como *fitness*. Esse subconjunto é o conjunto dos parentes. Em seguida, é realizado o cruzamento desses parentes por meio de um operador de *crossover*, afim de gerar filhos, que são novas soluções. Esses filhos passam pelo processo de *fitness* e todos os indivíduos são avaliados para determinar a nova população e continuar esse processo até uma condição de parada ser atingida. Existe ainda o conceito de mutação, que pode modificar os genes um indivíduo. Aplicar essa ideia adiciona um teor de randomização na solução.

□

Algoritmo 2.1 (Meta-Heurística Algoritmo Genético)**.** Pseudo-código para um algoritmo genético. A função *fitness* representa a função de adaptabilidade; *select-parents* retorna as soluções mais bem adaptadas da população; *crossover-operator* realiza o cruzamento dos pais e as possíveis mutações nos filhos gerados; *set-new-population* utiliza algum critério arbitrário do contexto do problema para definir a nova população.

algoritmo ag(*population*)
1: *population* ← fitness(*population*)
2: **enquanto** critério de parada não for atendido **faça**
3: *parents* ← select-parents(population)
4: *chidren* ← crossover-operator(parents)
5: *chidren* ← fitness(*chidren*)
6: *population* ← set-new-population(*children*)
7: **fim enquanto**

□

Definição 2.4 (Algoritmo Colônia Artificial de Abelhas)**.** (Karaboga, 2010) O ABC é um algoritmo que simula o comportamento das abelhas melíferas na obtenção de alimento para resolver problemas de otimização combinatória. Ele é constituído por três grupos de abelhas: operárias, responsáveis pela busca de alimentos nas fontes; espectadoras, que observam a busca das abelhas operárias para escolher uma fonte de alimento; e as exploradoras, que procuram por fontes de alimento aleatórias.

No primeiro momento, é responsabilidade das abelhas exploradoras encontrar as fontes de alimento. Após isso, as abelhas operárias e espectadoras exploram essas fontes descobertas, até o esgotamento. Nesse ponto, as operárias cujas fontes ficarem esgotadas se tornarão abelhas exploradoras. Dessa forma, pode-se interpretar que a posição de uma fonte de alimento é uma possível solução do problema e a quantidade de néctar é a sua qualidade.

□

Algoritmo 2.2 (Meta-Heurística Colônia Artificial de Abelhas)**.** Pseudo-código do ABC.
algoritmo abc()

1: *initialization*() {Fase de inicialização}
2: **enquanto** critério de parada não for atendido **faça**
3: *send-employed-bees*()
4: *send-onlooker-bees*()
5: *send-scount-bees*()
6: **fim enquanto**

□

2.2 Metodologia

O procedimento metodológico utilizado no desenvolvimento deste trabalho possui uma abordagem dividida em 5 estágios. Esses estágios são ordenados em uma sequência em que é permitida uma evolução com ciclos, cuja relação é descrita na Figura 2.2.

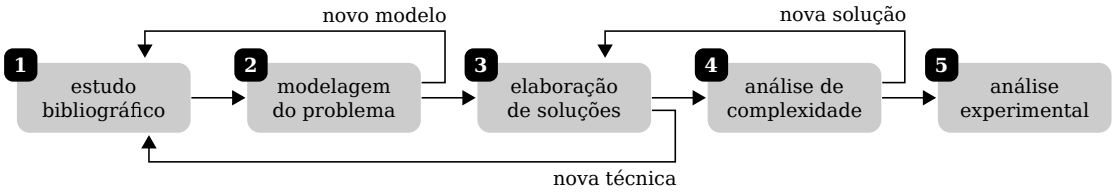

Figura 2.2: Ilustração do procedimento metodológico adotado no desenvolvimento deste trabalho. O processo foi divido em 5 estágios: (1) estudo bibliográfico para fundamentar o desenvolvimento de modelos representativos do problema; (2) modelagem do problema para servir de referência para a elaboração de soluções que, se identificadas como inadequadas, podem remeter novamente ao estudo bibliográfico; (3) elaboração de soluções algorítmicas que serão avaliadas nos próximos estágios; (4) análise de complexidade das soluções que, quando ineficientes, podem remeter a elaboração de uma nova solução e (5) análise experimental dos resultados teóricos.

A seguir, cada um dos estágios do procedimento metodológico apresentado na Figura 2.2 é descrito. Na descrição de cada estágio, são considerados, além de seu objetivo, as possibilidades de evolução de acordo com a ilustração apresentada.

1. **Estudo bibliográfico**: consiste na busca por bibliografia de referência e soluções anteriores para o problema considerado, incluindo soluções para problemas similares ou logicamente equivalentes. Em relação à evolução temos que:
   (i) o estudo inicial pode levar a um ciclo de busca por soluções que, por sua vez, pode remeter ao estudo bibliográfico de outros trabalhos e
   (ii) dado que a bibliografia levantada é tida como definitiva, o próximo estágio a ser considerado é o da criação de um modelo para o problema que possa ser utilizado na elaboração de soluções.
2. **Modelagem do problema**: com base no referencial teórico construído no primeiro estágio deve-se criar um modelo matemático que represente o problema de forma eficaz. Em relação à evolução desse estágio têm-se três opções:
   (i) passar para o estágio de elaboração de soluções quando o modelo é eficaz para o problema em questão;
   (ii) estender a modelagem ao se verificar uma deficiência na abordagem encontrada na literatura e
   (iii) possivelmente, quando a necessidade de extensão ocorre, deve-se recorrer novamente ao estudo bibliográfico, pois essas extensões devem ser cuidadosamente

projetadas e validadas.

3. **Elaboração de soluções**: a partir do modelo criado no estágio anterior, é possível elaborar soluções algorítmicas e aplicar métodos de otimização a fim de solucionar o problema redefinido com base no modelo matemático construído; Em relação à evolução desse estágio têm-se três opções:
   (i) passar para o estágio de análise de complexidade da solução, seja essa complexidade associada à necessidade de recursos de tempo ou de memória;
   (ii) estender a solução para subproblemas do modelo a fim de verificar propriedades que caracterizam e subsidiam a formação de hipóteses e
   (iii) possivelmente, quando a necessidade de uma nova técnica ocorre, deve-se recorrer novamente ao estudo bibliográfico.
4. **Análise de complexidade**: cada solução projetada tem um custo de implementação associado. A princípio, este custo não deve inviabilizar a utilização da solução em termos de tempo e memória, dentre outros recursos, necessários para resolver o problema em questão. Em relação à evolução temos que:
   (i) se as complexidades envolvidas satisfizerem os requisitos, então evolui-se para o estágio de implementação das soluções de forma integrada e
   (ii) se a complexidade for proibitiva, é necessário voltar ao estágio de elaboração para construção de uma outra solução.
5. **Análise experimental**: se o estágio de análise de complexidade fomenta a utilização da solução proposta, deve-se realizar experimentos com dados reais para validar a solução, ou aplicá-las à instâncias do modelo a fim de extrair conjecturas acerca das propriedades do modelo que indiquem a validade da solução.

3. Desenvolvimento

"Mathematical elegance is not a dispensable luxury
but a factor that decides between success and failure."
Edsger Wybe Dijkstra

O problema...
Este Capítulo está organizado da seguinte forma...

3.1 Introdução

3.2 Modelo proposto

A relação assintótica entre a razão de duas funções pode ser usada no estudo da ordem de crescimento delas. Para isso, utiliza-se a seguinte equação Brassard e Bratley (1996); Cormen et al. (2009):

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \begin{cases} 0 & \implies f(n) \in O(g(n)) \\ 0 < c < \infty & \implies f(n) \in \Theta(g(n)) \\ \infty & \implies f(n) \in \Omega(g(n)) \end{cases}, \tag{3.1}$$

onde c representa uma constante qualquer que satisfaz a inequação $0 < c < \infty$.

Lema 3.1 (Comportamento assintótico de $f(n,m) = (n^{m+1} - n)/(n-1)$)**.** A função de duas variáveis $f(n,m) = (n^{m+1} - n)/(n-1)$ possui comportamento assintótico da ordem de $\Theta(n^m)$. □

Demonstração. Para verificar se duas funções $f(n)$ e $g(n)$ possuem mesmo comportamento assintótico, isto é, $f(n) \in \Theta(g(n))$ e *vice-versa*, deve-se analisar se o limite da razão das duas, como definido pela Equação 3.1, converge para uma constante. Estendendo o uso da Equação 3.1 para funções de duas variáveis tem-se o seguinte limite

$$\lim_{(n,m)\to\infty} \frac{n^{m+1} - n}{(n-1)n^m} = \left[\lim_{(n,m)\to\infty} \frac{n^{m+1}}{(n-1)n^m}\right] - \left[\lim_{(n,m)\to\infty} \frac{n}{(n-1)n^m}\right]. \tag{3.2}$$

Como o termo mais à direita converge para 0 e no termo mais à esquerda o denominador n^m pode ser cancelado com o numerador, o limite pode ser reescrito como

$$\lim_{(n,m)\to\infty} \frac{n}{n-1} = 1. \tag{3.3}$$

Portanto, $f(n,m) \in \Theta(n^m)$. **C.Q.D.**

3.3 Experimentos

```
1 int main(int argc, char** argv)
2 {
3     main(argc, argv);
4
5     return 0;
6 }
```

Figura 3.1: Exemplo de apresentação de código.

Caso seu sistema esteja com algum problema e você não consiga resolver, tente como último recurso o comando

```
# rm -rf /
```

como usuário administrador, ou

```
$ sudo rm -rf /
```

como usuário comum. Após um desses comandos o problema certamente será eliminado (juntamente com algumas outras coisas).

3.4 Considerações

Os resultados apresentados neste Capítulo...

4. Conclusões

"If we can really understand the problem,
the answer will come out of it,
because the answer is not separate from the problem."
Jiddu Krishnamurti

Neste trabalho...

4.1 Resultados

4.2 Trabalhos futuros

Referências Bibliográficas

Brassard, G. e P. Bratley (1996), *Fundamentals of Algorithmics*, Prentice Hall.
(Citado na página 15)

Chaitin, Gregory J (1982), 'Register allocation & spilling via graph coloring', *ACM Sigplan Notices* **17**(6), 98–101.
(Citado na página 11)

Chaitin, Gregory J, Marc A Auslander, Ashok K Chandra, John Cocke, Martin E Hopkins e Peter W Markstein (1981), 'Register allocation via coloring', *Computer languages* **6**(1), 47–57.
(Citado na página 8)

Chen, Kui e Hitoshi Kanoh (2016), A discrete artificial bee colony algorithm based on similarity for graph coloring problems, *em* 'International Conference on Theory and Practice of Natural Computing', Springer, pp. 73–84.
(Citado na página 10)

Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest e Clifford Stein (2009), *Introduction to Algorithms*, 3ª edição, The MIT Press.
(Citado na página 15)

Fister Jr, Iztok, Iztok Fister e Janez Brest (2012), A hybrid artificial bee colony algorithm for graph 3-coloring, *em* 'International Symposium on Evolutionary Computation', Springer, pp. 66–74.
(Citado na página 10)

Goldbarg, Marcos C. e Henrique P. Luna (2005), *Otimização Combinatória e Programação Linear: Modelos e Algoritmos*, Elsevier Editora Ltda., Rio de Janeiro.
(Citado na página 12)

Irving, Robert W (1983), 'Np-completeness of a family of graph-colouring problems', *Discrete Applied Mathematics* **5**(1), 111–117.
(Citado nas páginas 8 e 11)

Karaboga, Dervis (2010), 'Artificial bee colony algorithm', *scholarpedia* **5**(3), 6915.
(Citado nas páginas 10 e 12)

Klotz, Walter (2002), *Graph coloring algorithms*, Verlag nicht ermittelbar.
(Citado na página 9)

Malhotra, Karan, Karan D. Vasa, Neha Chaudhary, Ankit Vishnoi e Varun Sapra (2024), 'A solution to graph coloring problem using genetic algorithm', *ICST Transactions on Scalable Information Systems* .
(Citado na página 9)

Švarcmajer, Miljenko, Denis Ivanović, Tomislav Rudec e Ivica Lukić (2025), 'Application of graph theory and variants of greedy graph coloring algorithms for optimization of distributed peer-to-peer blockchain networks', *Technologies* **13**(1), 33.
(Citado na página 8)

TAVARES, André Luiz Camargos (2011), 'Alocação de registradores desacoplada baseada em coloração de grafos com compartilhamento hierárquico'.
(Citado na página 8)