L(3,2,1)-Labeling on Graphs: Comparison between Greedy Heuristics and Genetic Algorithm

Jhon Kennedy C. de L. Queiroz[1] **, Atílio G. Luiz**[1]

[1]Universidade Federal do Ceará (UFC), Campus Quixadá, Ceará, Brasil

jhonqueiroz@alu.ufc.br, gomes.atilio@ufc.br

Abstract. *Several graph labeling problems have applications in the assignment of frequency channels to transmitters. One such problem is the $L(3,2,1)$-Labeling Problem, in which labels from the set $\{0,1,\ldots,k\}$ are assigned to the vertices of a graph in such a way that, for any two vertices $u, v \in V(G)$, the constraint $|f(u)-f(v)| \geq 4-dist(u,v)$ must be satisfied, where $dist(u,v)$ denotes the distance between u and v in G. Since this problem is NP-complete, this work proposes heuristic and metaheuristic approaches to obtain approximate solutions. Two greedy algorithms and one genetic algorithm were implemented. The results show that the greedy approach is more efficient in terms of runtime, while the genetic algorithm produces higher-quality solutions.*

Resumo. *Diversos problemas de rotulação em grafos encontram aplicação na atribuição de canais de frequência a transmissores. Um deles é o Problema da Rotulação-$L(3,2,1)$, no qual rótulos do conjunto $\{0,1,\ldots,k\}$ são atribuídos aos vértices de um grafo de forma a satisfazer a restrição de que, para quaisquer dois vértices $u, v \in V(G)$, deve-se ter que $|f(u) - f(v)| \geq 4 - dist(u,v)$, onde $dist(u,v)$ é a distância entre u e v em G. Como esse problema é NP-completo, este trabalho propõe abordagens heurísticas e meta-heurísticas para obter soluções aproximadas. Foram implementados dois algoritmos gulosos e um algoritmo genético. Os resultados mostram que a abordagem gulosa é mais eficiente em tempo de execução, enquanto o algoritmo genético gera soluções de melhor qualidade.*

1. Introdução

O *Problema da Atribuição de Canais de Frequência* (PAC) é um problema de otimização que pode ser enunciado da seguinte forma: dado um conjunto de transmissores distribuídos geograficamente sobre uma determinada área, deve-se atribuir canais de frequência a cada transmissor de modo a satisfazer um conjunto de restrições, minimizando o valor de uma função objetivo [Murphey et al. 1999]. Dependendo da rede de transmissores sob consideração (rádio, celular, wi-fi, etc.), a função objetivo pode variar ou até mesmo o conjunto de restrições pode não ser o mesmo. Essa classe de problemas existe desde o início do século XX, quando surgiram as aplicações marítimas do telégrafo sem fio [Jansky 1977].

Os primeiros problemas de atribuição de canais de frequência surgiram com a descoberta de que transmissores atribuídos à mesma frequência ou a frequências muito próximas tinham potencial para interferir uns com os outros. Assim, uma das abordagens para a atribuição de frequências foi minimizar ou eliminar essa potencial interferência.

Em resposta, matemáticos desenvolveram muitos modelos e resultados baseados na teoria dos grafos, introduzindo a teoria de *T-coloração* para modelar diversas instâncias do problema [Hale 1980]. Um modelo em grafo do problema de atribuição de frequências geralmente é obtido ao considerar os transmissores como os vértices de um grafo, sendo que as arestas entre os vértices representam possíveis interferências. As frequências são atribuídas aos vértices com o objetivo de minimizar o espectro utilizado ou minimizar a interferência. Dentre as formulações matemáticas baseadas em teoria dos grafos derivadas do PAC, destaca-se a *Rotulação-*$L(h,k)$, na qual inteiros não negativos (*rótulos*) são atribuídos aos vértices de um grafo, respeitando restrições baseadas na distância entre os vértices [Calamoneri 2006]. Por exemplo, a Rotulação-$L(2,1)$, introduzida por Griggs e Yeh [Griggs and Yeh 1992], estabelece que vértices adjacentes devem ter rótulos que difiram em módulo de pelo menos duas unidades, enquanto vértices à distância 2 devem receber rótulos distintos.

Em 2004, motivados pelo conceito de Rotulação-$L(2,1)$, Shao e Liu [Shao and Liu 2004] criaram uma nova variante, introduzindo o conceito de *Rotulação-*$L(3,2,1)$. Formalmente, dado um grafo $G = (V,E)$, uma função $f\colon V \rightarrow \{0,1,\ldots,k\}$ é uma *k-rotulação-*$L(3,2,1)$ de G se, dados dois vértices $u,v \in V$ quaisquer, as seguintes restrições forem satisfeitas:

(a) $|f(u) - f(v)| \geq 3$ se u e v forem adjacentes;
(b) $|f(u) - f(v)| \geq 2$ se u e v estiverem à distância 2;
(c) $|f(u) - f(v)| \geq 1$ se u e v estiverem à distância 3.

A Figura 1 ilustra um grafo com uma rotulação-L(3,2,1).

Figura 1. Grafo com uma rotulação-L(3,2,1).

O maior rótulo k atribuído a um vértice do grafo por uma k-rotulação-L(3,2,1) f é chamado *span* da função f, também denotado por $\lambda(f)$. O *número cromático L(3,2,1)* de um grafo G, denotado por $\lambda_{3,2,1}(G)$, é o menor *span* que uma rotulação-L(3,2,1) de G pode ter, i.e., $\lambda_{3,2,1}(G) = \min\{\lambda(f) \mid f \text{ é uma rotulação-L(3,2,1) de } G\}$. O *Problema da Rotulação-L(3,2,1)* consiste em determinar o valor do parâmetro $\lambda_{3,2,1}(G)$ para qualquer grafo G. O objetivo aqui, então, consiste em encontrar uma rotulação-L(3,2,1) de um grafo que tenha o menor *span* possível. Como a versão de decisão desse problema é NP-completo, o uso de heurísticas e meta-heurísticas, ou até mesmo métodos exatos para instâncias de tamanho moderado, torna-se uma estratégia viável para a obtenção de soluções aproximadas em tempo aceitável [Murphey et al. 1999, Cormen et al. 2009].

O Problema da Rotulação-L(3,2,1) tem sido largamente tratado até então pelo viés da teoria dos grafos e, apesar de avanços teóricos em classes de grafos específicas [Clipperton et al. 2005, Chia et al. 2011, Florencio and Luiz 2021], determinar o parâmetro $\lambda_{3,2,1}(G)$ para grafos arbitrários continua sendo um desafio computacional relevante. Até onde sabemos, o único trabalho publicado até agora que aborda a Rotulação-L(3,2,1) por um viés computacional é o trabalho de Shao e Vessel [Shao and Vesel 2013], que propõem um modelo de programação linear inteira para o problema e o utilizam para determinar o parâmetro $\lambda_{3,2,1}(G)$ para algumas classes de produtos de grafos.

Neste trabalho, propomos um Algoritmo Genético (AG) para o Problema da Rotulação-$L(3,2,1)$. Até onde sabemos, e o que indica a Revisão Sistemática da Literatura (RSL) realizada, esta é a primeira proposta de uma meta-heurística de uso geral aplicada ao problema. O AG é comparado com duas heurísticas gulosas clássicas de coloração de grafos: uma com ordem aleatória dos vértices e outra com ordenação dos vértices por decrescente de graus. Os três algoritmos foram avaliados sobre um mesmo conjunto de instâncias, composto por grafos de diferentes classes, selecionados para formar um benchmark representativo. A comparação considerou tanto a qualidade das soluções obtidas *(span)* quanto o tempo de execução. Os resultados mostram que, embora as heurísticas gulosas apresentem vantagens em tempo, o algoritmo genético se destaca pela qualidade superior das soluções geradas, especialmente em instâncias maiores ou mais densas.

2. Trabalhos Relacionados

A Rotulação-$L(3,2,1)$ foi introduzida em 2004 e, desde então, alguns avanços teóricos foram registrados na literatura [Shao and Liu 2004]. Em 2005, Clipperton et al. [Clipperton et al. 2005] determinaram o número cromático L(3,2,1) para diversas classes de grafos simples, tais como: caminhos, ciclos, *caterpillars*, árvores n-árias, grafos completos e grafos bipartidos completos. Ademais, os autores propuseram um algoritmo guloso para a Rotulação-L(3,2,1) de grafos. Com base nesse algoritmo, eles provam que todo grafo G com grau máximo Δ possui $\lambda_{3,2,1}(G) \leq \Delta^3 + \Delta^2 + 3\Delta$. Posteriormente, em 2011, Chia et al. [Chia et al. 2011] melhoraram esse limitante superior, estabelecendo o Teorema 1, que continua sendo o melhor limitante superior conhecido.

Teorema 1 *Se G é um grafo com grau máximo Δ, então $\lambda_{3,2,1}(G) \leq \Delta^3 + 2\Delta$.*

Chia et al. [Chia et al. 2011] também apresentam limitantes inferiores para o parâmetro $\lambda_{3,2,1}(G)$, como apresentados a seguir.

Teorema 2 *Se G é um grafo com grau máximo $\Delta > 0$, então $\lambda_{3,2,1}(G) \geq 2\Delta + 1$. Ademais, se G for um grafo Δ-regular, então $\lambda_{3,2,1}(G) \geq 2\Delta + 2$.*

No contexto de abordagens meta-heurísticas, até onde se sabe, não foram publicadas propostas de algoritmos dessa natureza para o problema da Rotulação-$L(3,2,1)$. No entanto, existem propostas de algoritmos genéticos e outras meta-heurísticas para o problema relacionado da Rotulação-$L(2,1)$ [Han and Kim 2008, Panda and Goel 2010].

Em 2008, Han e Kim [Han and Kim 2008] investigaram a aplicação de algoritmos genéticos para a Rotulação-$L(2,1)$, onde o foco principal foi otimizar a função objetivo através de operadores evolutivos tradicionais; eles utilizaram codificação por permutação dos vértices para representar uma solução. Os autores também analisaram diferentes estratégias de cruzamento e mutação, e validaram seus métodos com experimentos em três grafos multipartidos completos. Posteriormente, Panda e Goel [Panda and Goel 2010] propuseram dois algoritmos gulosos e algoritmos genéticos para a Rotulação-L(2,1), variando a ordem de processamento dos vértices com heurísticas como *smallest-last* e *largest-first*. Seus experimentos mostraram que os algoritmos genéticos superam as abordagens gulosas em qualidade da solução, embora exijam mais tempo de execução.

No contexto de algoritmos exatos, destacamos o trabalho de Shao e Vesel [Shao and Vesel 2013] que propuseram um modelo de Programação Inteira (PI) para

o Problema da Rotulação-$L(3,2,1)$. A formulação foi aplicada a grafos derivados de produtos fortes e cartesianos de ciclos e caminhos. Apesar de apresentar resultados exatos, a escalabilidade do modelo é limitada a grafos de pequeno porte.

Uma Revisão Sistemática da Literatura (RSL) foi conduzida e identificou diversos trabalhos relacionados à Rotulação-$L(3,2,1)$, em sua maioria composta por contribuições teóricas para classes específicas de grafos. Embora alguns estudos proponham heurísticas ou algoritmos híbridos, não foram encontrados trabalhos que explorem meta-heurísticas aplicadas de forma geral ao problema. A Tabela 1 apresenta uma comparação entre os principais trabalhos da literatura e o presente estudo.

Tabela 1. Comparação entre trabalhos da literatura e o presente estudo

Critério	Panda e Goel (2010)	Clipperton et al. (2005)	Shao e Vesel (2013)	Este Trabalho
Problema abordado	Rotulação-$L(2,1)$	Rotulação-$L(3,2,1)$	Rotulação-$L(3,2,1)$	Rotulação-$L(3,2,1)$
Tipo de abordagem	Heurística / Genético	Teórica	Exata (PLI)	Heurística / Genético
Famílias de grafos	Aleatórios	Caminhos, árvores, completos	Produtos de ciclos e caminhos	Diversas (paths, grids, bipartidos)
Avaliação computacional	Sim	Não	Sim (instâncias pequenas)	Sim (benchmark diverso)
Algoritmo genético	Sim	Não	Não	Sim (com elitismo)

3. Heurísticas Gulosas

Uma *estratégia gulosa* é uma técnica utilizada para resolver problemas computacionais por meio de decisões locais imediatas que parecem ser as melhores em cada etapa, sem reconsiderar escolhas anteriores. Embora não garanta a solução ótima global, sua simplicidade e rapidez tornam-na uma estratégia vantajosa para problemas combinatórios [Cormen et al. 2009]. Clipperton et al. [Clipperton et al. 2005] foram os primeiros a propor um algoritmo guloso para a Rotulação-L(3,2,1). Antes de apresentar o algoritmo, precisamos de uma definição adicional. Dado um vértice v de um grafo G, seja $N_k(v) = \{w \mid dist(v,w) = k\}$, ou seja, $N_p(v)$ é o conjunto dos vértices que estão à distância k de v em G. O pseudocódigo do algoritmo guloso é exibido no Algoritmo 1.

O Algoritmo 1 recebe como entrada uma ordenação dos vértices e constrói uma rotulação-L(3,2,1) viável. A ordem em que os vértices são visitados impacta diretamente no *span* da solução. É possível provar o resultado que segue.

Teorema 3 *Existe pelo menos uma permutação π dos vértices do grafo G que, quando dada como entrada para o algoritmo guloso, produz como saída uma função de rotulação-$L(3,2,1)$ f do grafo G tal que $\lambda(f) = \lambda_{3,2,1}(G)$.*

Pelo Teorema 3, o algoritmo guloso pode atingir uma solução ótima, desde que receba uma ordem adequada dos vértices. Porém, como essa ordem é desconhecida, faz-se necessário testar diferentes estratégias de ordenação.

Dado esse contexto, a proposta deste trabalho consiste em usar Algoritmos Genéticos para encontrar boas ordenações dos vértices que possam ser usadas como entrada para o Algoritmo 1.

3.1. *Random-Greedy (RG)* e *Largest-Degree-First-Greedy (LDFG)*

Neste trabalho, vamos comparar o nosso algoritmo genético com o Algoritmo 1, para os casos em que este último recebe entradas específicas. Chamaremos o Algoritmo 1 de *Random-Greedy (RG)* quando ele receber como entrada uma ordenação aleatória dos

vértices de um grafo G. Por outro lado, chamaremos o Algoritmo 1 de *Largest-Degree-First-Greedy (LDFG)* quando ele receber como entrada uma ordenação $(v_1, v_2, \ldots, v_n)$ dos vértices de um grafo G tal que $d(v_1) \geq d(v_2) \geq \cdots \geq d(v_n)$, ou seja, os vértices estão em ordem decrescente de grau.

Algoritmo 1: Algoritmo Guloso para Rotulação-$L(3,2,1)$

Entrada: Grafo $G = (V, E)$ com $|V| = n$, ordenação $O = (v_1, \ldots, v_n)$ dos vértices de G
Saída: maior rótulo utilizado k e uma k-rotulação-$L(3,2,1)$ f de G

```
1  k ← 0;
2  for i ← 1 to n do
3      f(v_i) ← −1 ;                          // v_i não está colorido
4  for i ← 1 to n do
5      P ← ∅ ;                  // Conjunto dos rótulos proibidos
6      foreach u ∈ N_1(v_i) do
7          if f(u) ≠ −1 then
8              P ← P ∪ {f(u) − 2, f(u) − 1, f(u), f(u) + 1, f(u) + 2};
9      foreach u ∈ N_2(v_i) do
10         if f(u) ≠ −1 then
11             P ← P ∪ {f(u) − 1, f(u), f(u) + 1};
12     foreach u ∈ N_3(v_i) do
13         if f(u) ≠ −1 then
14             P ← P ∪ {f(u)};
15     f(v_i) ← min{{0, ..., Δ^3 + Δ^2 + 3Δ}\P};
16     k ← max{k, f(v_i)};
17 return k, f;
```

4. Algoritmo Genético (AG)

Os Algoritmos Genéticos (AGs) são métodos de busca estocástica pertencentes à classe das meta-heurísticas evolutivas, inspirados nos princípios da seleção natural e da genética biológica. Formalizados por Holland na década de 1970 [Holland 1975], os AGs operam sobre uma população de soluções candidatas, que evoluem ao longo de gerações por meio de operadores de seleção, cruzamento e mutação. A cada geração, indivíduos com maior aptidão têm maior probabilidade de contribuir com a próxima população, promovendo, assim, um processo adaptativo de melhoria progressiva. Os algoritmos genéticos têm sido especialmente eficazes na resolução de problemas de otimização combinatória com grandes espaços de busca, múltiplos ótimos locais e funções objetivo não diferenciáveis [Goldberg 1989, Mitchell 1998]. Tais características os tornam apropriados para abordar o problema de Rotulação-L(3,2,1), cuja versão de decisão é NP-completa. Nesse contexto, os AGs oferecem uma alternativa promissora para a construção de soluções aproximadas de boa qualidade, mesmo em instâncias grandes com estruturas complexas.

Nas subseções a seguir, apresentamos os componentes principais do algoritmo genético proposto: a codificação da solução, a função de avaliação de cada indivíduo, e

os operadores de cruzamento, mutação e seleção que foram utilizados.

4.1. Codificação da solução

No algoritmo genético proposto, cada indivíduo da população é representado por uma permutação dos vértices do grafo $G = (V, E)$, com $V = \{v_0, v_1, \ldots, v_{n-1}\}$. Formalmente, um indivíduo é codificado como uma sequência $\pi = (v_{\pi(0)}, v_{\pi(1)}, \ldots, v_{\pi(n-1)})$, onde $\pi : \{0, \ldots, n-1\} \to \{0, \ldots, n-1\}$. Essa permutação define uma ordem de visitação dos vértices sobre a qual é aplicada a heurística gulosa descrita na Seção 3. A cada vértice $v_i \in \pi$, é atribuído o menor rótulo $r \in \mathbb{N}$ que satisfaça as restrições da rotulação-$L(3, 2, 1)$, considerando os vértices já rotulados. O custo da solução correspondente é dado pelo maior rótulo atribuído, isto é, $\lambda(f_\pi) = \max_{v \in V} f_\pi(v)$, onde f_π é a função de rotulação construída a partir da permutação π. Essa escolha visa garantir a integridade estrutural da solução, evitando repetições e inconsistências. Representações por permutação são amplamente utilizadas em problemas como coloração de grafos e escalonamento [Gen and Cheng 1997]. A Figura 2 ilustra a codificação de uma solução.

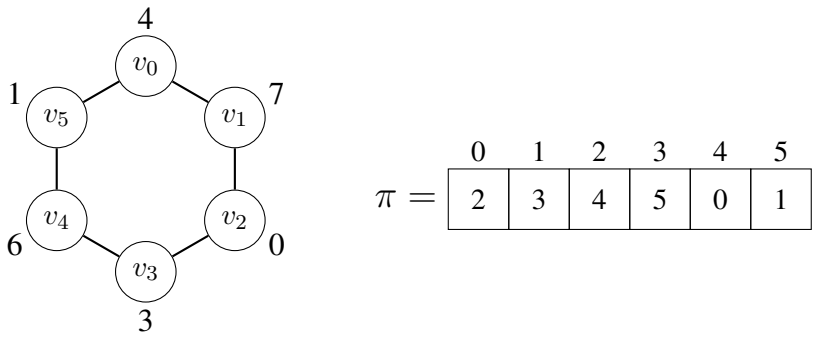

Figura 2. Um grafo ciclo C_6 munido de uma rotulação-L(3,2,1) e, ao lado, um cromossomo π que define uma ordem de visitação dos vértices que, quando passada para o Algoritmo 1, produz a rotulação-L(3,2,1) vista na figura.

O Algoritmo 2 apresenta a estrutura geral do algoritmo genético proposto.

4.2. Função de Aptidão (*Fitness*)

A *aptidão* (do inglês, *fitness*) de cada indivíduo é avaliada com base no maior rótulo atribuído ao cromossomo após a aplicação da heurística gulosa apresentada no Algoritmo 1. Essa estratégia, conhecida como abordagem híbrida, é recorrente na literatura e combina a capacidade exploratória dos AGs com a eficiência construtiva de heurísticas gulosas, favorecendo tanto a qualidade quanto a velocidade de convergência [Hart et al. 2005].

4.3. Operador de Seleção

Foi empregada a *seleção por torneio binário* ($k = 2$). Nessa abordagem, dois indivíduos são escolhidos aleatoriamente e o de melhor aptidão é selecionado para reprodução. Esse método fornece um bom equilíbrio entre *pressão seletiva* e *diversidade populacional*, contribuindo para evitar a convergência prematura [Goldberg 1989].

Algoritmo 2: Algoritmo Genético para Rotulação-$L(3, 2, 1)$

Entrada: Grafo $G = (V, E)$; número máximo de gerações T; taxa de cruzamento p; taxa de mutação m; taxa de elitismo e; fração da população α

Saída: Melhor permutação π^* com menor $\lambda(f)$

Inicializar P com $\alpha \cdot |V|$ permutações de V; Avaliar $\pi \in P$; Armazenar π^*;
for $t \leftarrow 1$ **to** T **do**
    $P' \leftarrow \emptyset$;
    Inserir $e\%$ melhores de P em P' ; `// Elitismo`
    while $|P'| < |P|$ **do**
        Selecionar π_1, π_2 via torneio binário;
        if *random()* $< p$ **then**
            $(\text{filho}_1, \text{filho}_2) \leftarrow \text{Cruzamento}(\pi_1, \pi_2)$;
        else
            $\text{filho}_1 \leftarrow \pi_1$; $\text{filho}_2 \leftarrow \pi_2$;
        if *random()* $< m$ **then**
            Mutação em filho_1;
        if *random()* $< m$ **then**
            Mutação em filho_2;
        Inserir filhos em P';
    $P \leftarrow P'$; Avaliar P; Atualizar π^* se necessário;
return π^*, $\lambda(f)$

4.4. Operador de Cruzamento

Foi implementado um operador de cruzamento *Position-based Order Preserving* (POP), descrito em [Mumford 2006], também conhecido como cruzamento baseado em prefixo. Para cada filho, seleciona-se aleatoriamente um ponto de corte, e os genes do início até esse ponto são herdados diretamente de um dos pais. Em seguida, os vértices restantes são inseridos na ordem relativa do segundo pai, excluindo duplicatas. Esse operador garante a validade da permutação resultante e preserva blocos de ordem dos pais originais, sendo adequado para problemas de rotulação e coloração de vértices [Gen and Cheng 1997].

4.5. Operador de Mutação

A mutação utilizada foi a *mutação por troca de vizinhos* (*neighbors swap mutation*). Nesse operador, um vértice é escolhido aleatoriamente e os seus vizinhos têm as suas posições trocadas. Essa mutação mantém a viabilidade da permutação e é eficiente para introduzir variação genética no processo evolutivo [Gen and Cheng 1997].

4.6. Elitismo

Foi adotada uma política de *elitismo*, na qual os 10% melhores indivíduos de cada geração são diretamente copiados para a próxima. Essa técnica visa preservar boas soluções e acelerar a convergência do algoritmo, reduzindo o risco de regressão no desempenho entre gerações [Mitchell 1998].

5. Configuração Experimental

Este estudo avaliou três abordagens heurísticas para o Problema da Rotulação-$L(3, 2, 1)$: as heurísticas gulosas *Random Greedy* e *Largest-Degree-First-Greedy* e um algoritmo genético baseado em permutação.

Os testes foram realizados sobre um conjunto diversificado de 66 grafos, composto pelas seguintes classes: (i) 27 grafos da classe grade $P_m \square P_n$, gerados com uso da biblioteca *NetworkX* [Hagberg et al. 2008]; (ii) 3 grafos da coleção *CELAR*; (iii) 5 grafos da subcoleção *SUBCELAR6*; (iv) 3 grafos da coleção *DIMACS* [Johnson and Trick 1996]; e (v) 28 grafos da coleção *Harwell-Boeing*, que oferecem exemplos de grafos esparsos que permitem testar os nossos algoritmos em grafos com baixa densidade [Duff et al. 2025]. As classes CELAR e SUBCELAR6 são frequentemente usadas para testes em algoritmos desenvolvidos especificamente para problemas de atribuição de canais de frequência [CELAR 2025]. Já os grafos da coleção DIMACS são tradicionalmente utilizados em testes em problemas de coloração de vértices.

As implementações foram desenvolvidas em *Python* e executadas em um *MacBook Air* (2022), equipado com processador Apple M2 de 8 núcleos, 8 GB de RAM e sistema macOS 15, utilizando o ambiente *Visual Studio Code*.

No algoritmo genético, o tamanho da população foi fixado em $\frac{N}{4}$, onde N representa o número de vértices do grafo. O número de gerações foi estabelecido em 300, esse valor foi definido com base em uma análise empírica de convergência realizada a partir da implementação de um mecanismo de rastreio que registra as gerações em que ocorrem melhorias no valor de *fitness* da solução. A análise dos dados indicou que a maior parte das melhorias ocorre nas primeiras 200 gerações; no entanto, em algumas instâncias, como grafos das classes *DIMACS* e *Harwell-Boeing*, observou-se a ocorrência de melhorias em estágios mais avançados da execução. As taxas utilizadas no algoritmo genético, taxa de cruzamento de $0{,}8$, taxa de mutação de $0{,}2$ e elitismo de 10%, seguem recomendações clássicas da literatura [Goldberg 1989]. Já para os algoritmos gulosos, foram geradas 150 instâncias para cada grafo. O algoritmo genético foi executado exatamente 30 vezes para cada grafo.

A fim de avaliar a qualidade dos *spans* das rotulações-L(3,2,1) retornadas pelo nosso algoritmo genético e pelas heurísticas gulosas, nós comparamos os valores obtidos por esses algoritmos com os melhores limitantes superiores e inferiores conhecidos para o parâmetro $\lambda_{3,2,1}(G)$ (ver os Teoremas 1 e 2). No caso dos grafos grade $P_m \square P_n$ que foram usados nos experimentos, todos eles têm o número cromático L(3,2,1) conhecido, o valor é $\lambda_{3,2,1}(P_m \square P_n) = 11$ para $n \geq m \geq 4$ [Chia et al. 2011].

6. Resultados e Discussões

A Tabela 2 apresenta os resultados obtidos especificamente sobre os grafos da classe *GRID*. A Tabela 3, por sua vez, resume os resultados obtidos pelas três abordagens nas demais famílias de grafos consideradas. Para cada instância, são informados: a classe do grafo, o número de vértices ($|V|$), a densidade, o limitante inferior para o valor do *span* (**LI**), o melhor limitante superior conhecido para o valor do *span* (**LS**) dado pelo Teorema 1, o menor valor de $\lambda(f)$ obtido pelo algoritmo genético (AG_$\lambda(f)$), a média e o desvio padrão das 30 execuções (AG_AVG ± DP), o valor obtido pelas heurísticas

RG e LDFG, além dos tempos médios de execução (em segundos) das heurísticas gulosas (RG_Tempo e LDFG_Tempo).

De acordo com os experimentos realizados, percebe-se que o algoritmo genético (AG) demonstra um desempenho superior em relação às heurísticas gulosas (RG e LDFG) na maioria das instâncias testadas. Essa vantagem é facilmente identificada em famílias de grafos com maior número de vértices ou maior densidade, como as classes *DIMACS* e *CELAR*, onde as heurísticas tendem a encontrar soluções subótimas com maior frequência. Por outro lado, o tempo de execução do AG é mais elevado, principalmente em instâncias mais complexas. De todo modo, tratando-se de aplicações práticas em que a qualidade da solução é prioritária, tal custo computacional pode ser justificado. Em instâncias menores (ou mais esparsas) como os grafos da família GRID ou SUBCELAR6, as heurísticas gulosas apresentaram desempenho competitivo, alcançando valores de $\lambda(f)$ próximos aos do AG, com tempos de execução significativamente inferiores.

Sobre os grafos da classe *GRID*, os valores ótimos do parâmetro $\lambda_{3,2,1}$ são conhecidos na literatura. Segundo Chia et al. (2011) [Chia et al. 2011], o valor exato de $\lambda_{3,2,1}(P_m \times P_n)$ foi determinado para todas as configurações de m e n utilizadas nestes testes. O AG não foi capaz de alcançar o valor ótimo dessas instâncias, porém sua proximidade com esses valores indica uma boa capacidade de exploração do espaço de busca; além disso, o GA apresentou um desvio padrão baixo, demonstrando consistência nas soluções. Já as heurísticas gulosas, apesar de serem mais rápidas, tiveram resultados ainda mais distantes.

Tabela 2. Resultados experimentais dos três algoritmos aplicados aos grafos da classe Grid.

Classe	Grafo	|V|	Densidade	$\lambda_{3,2,1}$	AG_$\lambda(f)$	AG_AVG (±DP)	RG_$\lambda(f)$	LDFG_$\lambda(f)$	GR_Tempo	LDFG_Tempo
Grid	grid_6x6	36	0.09524	11	13	14.13 (±0.90)	15	14	0.00010	0.00010
Grid	grid_6x7	42	0.08246	11	13	14.47 (±0.78)	16	14	0.00012	0.00014
Grid	grid_7x7	49	0.07143	11	13	14.87 (±0.63)	16	15	0.00014	0.00015
Grid	grid_7x8	56	0.06299	11	14	14.87 (±0.51)	17	15	0.00018	0.00016
Grid	grid_8x8	64	0.05556	11	15	15.53 (±0.51)	17	16	0.00019	0.00020
Grid	grid_8x9	72	0.04969	11	14	15.50 (±0.63)	18	17	0.00025	0.00023
Grid	grid_9x9	81	0.04444	11	15	15.93 (±0.58)	17	17	0.00026	0.00027
Grid	grid_9x10	90	0.04020	11	15	16.03 (±0.76)	18	17	0.00032	0.00029
Grid	grid_10x10	100	0.03636	11	15	16.20 (±0.66)	19	18	0.00040	0.00035
Grid	grid_10x11	110	0.03319	11	15	16.57 (±0.63)	19	18	0.00047	0.00037
Grid	grid_11x11	121	0.03030	11	15	16.87 (±0.73)	19	18	0.00045	0.00043
Grid	grid_11x12	132	0.02787	11	16	16.73 (±0.52)	20	18	0.00049	0.00047
Grid	grid_12x12	144	0.02564	11	16	16.97 (±0.56)	19	18	0.00049	0.00052
Grid	grid_12x13	156	0.02374	11	16	17.20 (±0.61)	19	19	0.00056	0.00059
Grid	grid_13x13	169	0.02198	11	16	17.23 (±0.57)	20	19	0.00064	0.00064
Grid	grid_13x14	182	0.02046	11	16	17.57 (±0.73)	20	20	0.00067	0.00071
Grid	grid_14x14	196	0.01905	11	17	17.73 (±0.58)	20	20	0.00075	0.00081
Grid	grid_14x15	210	0.01782	11	16	17.67 (±0.71)	21	20	0.00092	0.00081
Grid	grid_15x15	225	0.01667	11	17	18.13 (±0.73)	21	20	0.00090	0.00082
Grid	grid_15x16	240	0.01566	11	17	18.27 (±0.64)	20	20	0.00091	0.00096
Grid	grid_16x16	256	0.01471	11	17	18.30 (±0.60)	22	20	0.00106	0.00102
Grid	grid_16x17	272	0.01386	11	18	18.33 (±0.48)	21	20	0.00105	0.00107
Grid	grid_17x17	289	0.01307	11	18	18.63 (±0.61)	21	21	0.00112	0.00115
Grid	grid_17x18	306	0.01236	11	17	18.57 (±0.73)	22	21	0.00126	0.00120
Grid	grid_18x18	324	0.01170	11	17	18.80 (±0.66)	21	22	0.00135	0.00132
Grid	grid_18x19	342	0.01110	11	18	18.93 (±0.64)	22	22	0.00143	0.00139
Grid	grid_19x19	361	0.01053	11	18	18.90 (±0.61)	21	22	0.00145	0.00156

Tabela 3. Resultados experimentais dos três algoritmos aplicados aos grafos das demais classes.

Classe	Grafo	|V|	Densidade	LI	LS	AG_$\lambda(f)$	GA_AVG (±DP)	RG_$\lambda(f)$	LDFG_$\lambda(f)$	GR_Tempo	LDFG_Tempo
Harwell-Boeing	ash85	85	0.06134	19	747	28	28.57 (±0.63)	31	28	0.00057	0.00057
Harwell-Boeing	bcspwr01	39	0.06208	11	135	12	13.10 (±0.61)	14	13	0.00009	0.00009
Harwell-Boeing	bcspwr02	49	0.05017	13	228	15	15.47 (±0.57)	17	16	0.00015	0.00016
Harwell-Boeing	bcspwr03	118	0.02593	19	747	21	21.73 (±0.52)	24	23	0.00051	0.00062
Harwell-Boeing	bcspwr04	274	0.01789	31	3405	42	43.20 (±0.66)	47	44	0.00290	0.00327
Harwell-Boeing	bcsstk01	48	0.15603	23	1353	44	44.70 (±0.70)	47	45	0.00085	0.00084
Harwell-Boeing	bcsstk03	112	0.04247	11	135	17	18.67 (±0.66)	21	21	0.00036	0.00037
Harwell-Boeing	can__24	24	0.24638	17	528	24	25.20 (±0.61)	24	24	0.00012	0.00014
Harwell-Boeing	can__61	61	0.13552	49	13872	52	54.50 (±1.43)	59	57	0.00111	0.00122
Harwell-Boeing	can__62	62	0.04125	13	228	14	15.63 (±0.61)	17	16	0.00016	0.00017
Harwell-Boeing	can__73	73	0.05784	17	528	29	30.33 (±0.84)	32	29	0.00068	0.00071
Harwell-Boeing	ck104	104	0.08290	19	747	31	32.30 (±0.75)	35	34	0.00077	0.00087
Harwell-Boeing	curtis54	54	0.08665	31	3405	31	31.77 (±0.73)	34	34	0.00041	0.00044
Harwell-Boeing	dwt__59	59	0.06078	11	135	16	17.63 (±0.67)	20	19	0.00022	0.00021
Harwell-Boeing	dwt__66	66	0.05921	11	135	15	16.17 (±0.53)	18	18	0.00018	0.00018
Harwell-Boeing	dwt__72	72	0.02934	9	72	10	10.63 (±0.56)	11	12	0.00012	0.00012
Harwell-Boeing	dwt__87	87	0.06068	25	1752	33	34.40 (±1.00)	38	34	0.00072	0.00070
Harwell-Boeing	fidap005	27	0.35897	29	2772	37	38.60 (±1.00)	39	40	0.00024	0.00028
Harwell-Boeing	ibm32	32	0.18145	23	1353	34	35.27 (±1.05)	37	35	0.00036	0.00032
Harwell-Boeing	impcol_b	59	0.16423	35	4947	59	61.67 (±1.73)	67	63	0.00157	0.00161
Harwell-Boeing	jgl009	9	0.88889	17	528	21	22.60 (±0.86)	22	23	0.00003	0.00004
Harwell-Boeing	jgl011	11	0.89091	21	1020	28	28.43 (±0.50)	29	30	0.00006	0.00008
Harwell-Boeing	olm100	100	0.03980	11	135	18	18.53 (±0.51)	20	20	0.00035	0.00034
Harwell-Boeing	pores_1	30	0.23678	19	747	24	25.13 (±0.94)	26	27	0.00017	0.00021
Harwell-Boeing	rgg010	10	1	20	747	27	27.00 (±0.00)	28	28	0.00005	0.00006
Harwell-Boeing	rw136	136	0.04063	15	357	29	30.97 (±0.76)	34	33	0.00112	0.00117
Harwell-Boeing	tub100	100	0.02990	7	33	12	13.57 (±0.68)	15	14	0.00023	0.00023
Harwell-Boeing	will57	57	0.07957	21	1020	24	25.37 (±0.72)	27	27	0.00030	0.00034
CELAR	scen02	200	0.062	89	85272	93	94.70 (±1.37)	103	104	0.00684	0.00791
CELAR	scen06	200	0.066	89	85272	94	96.43 (±1.30)	102	110	0.00723	0.00879
CELAR	scen07	400	0.035	125	238452	140	144.90 (±1.95)	158	158	0.02176	0.02366
DIMACS	r125.5	125	0.495	199	970497	249	253.70 (±2.60)	261	271	0.03703	0.04719
DIMACS	r250.1	250	0.027	27	2223	39	39.77 (±0.63)	43	40	0.00225	0.00252
DIMACS	dsjc125.1	125	0.094	47	12213	123	125.10 (±1.37)	132	124	0.01502	0.01533
SUBCELAR6	CELAR6-SUB0	32	0.44960	51	15675	61	64.50 (±1.28)	65	70	0.00067	0.00086
SUBCELAR6	CELAR6-SUB1	28	0.83069	55	19737	71	72.93 (±1.17)	73	69	0.00066	0.00066
SUBCELAR6	CELAR6-SUB2	32	0.74395	63	29853	77	79.53 (±1.25)	81	92	0.00095	0.00121
SUBCELAR6	CELAR6-SUB3	36	0.69683	71	42945	86	88.60 (±1.48)	90	92	0.00132	0.00163
SUBCELAR6	CELAR6-SUB4	44	0.52748	73	46728	84	87.53 (±1.85)	93	109	0.00170	0.00235

Na classe *Harwell-Boing*, o algoritmo genético (AG) demonstrou uma vantagem clara em termos de qualidade de solução, obtendo valores de $\lambda(f)$ inferiores aos das heurísticas gulosas, mantendo uma boa consistência mesmo em grafos com mais de 100 vértices.

Já nas instâncias das classes *CELAR* e da *SUBCELAR6* — que são conhecidas por modelar cenários realistas de alocação de frequências — o algoritmo genético teve desempenho notavelmente superior em todas as instâncias. AG se mantém superior às heurísticas gulosas, sobretudo quando a densidade do grafo é maior. Os tempos de execução das heurísticas, como esperado, foram menores, mas sem ganhos de qualidade competitivos, mesmo em instâncias menores.

Por fim, na classe *DIMACS*, o algoritmo genético demonstrou desempenho robusto e estável. Em todas as instâncias analisadas, o AG superou as heurísticas gulosas. Por exemplo, na instância *r125.5*, com densidade $0{,}495$, o AG obteve $\lambda(f) = 249$, contra 261 da RG e 271 da LDFG. Os tempos de execução foram mais altos do que nas heurísticas, mas permanecem viáveis para aplicações onde a qualidade da solução é prioridade.

7. Conclusão

A análise dos resultados experimentais confirma a eficácia do Algoritmo Genético (AG) em relação às heurísticas gulosas Random-Greedy (RG) e Largest-Degree-First-Greedy (LDFG) na resolução do problema de rotulação-L(3,2,1). Em praticamente todas as famílias de grafos testadas, alcançou menores valores de $\lambda(f)$. Seu desempenho foi especialmente expressivo em grafos complexos e densos, como os da base *DIMACS e CELAR*,

onde a busca estocástica demonstrou maior capacidade de escapar de ótimos locais.

Embora os tempos de execução do AG sejam superiores aos das heurísticas gulosas, principalmente nas instâncias maiores, esse custo computacional se justifica frente à melhoria significativa na qualidade das soluções encontradas. Isso reforça o papel do GA como uma alternativa eficiente para problemas onde a qualidade da rotulação é mais crítica do que o tempo de execução.

Para aplicações práticas que exigem soluções rápidas, as heurísticas ainda são relevantes, especialmente o LDFG, que obteve resultados competitivos em bases regulares como os grafos Grid. No entanto, quando o objetivo é minimizar ao máximo o *span* — como em sistemas sensíveis à interferência de frequências — o uso de estratégias baseadas em meta-heurísticas, como o AG aqui proposto, mostra-se mais vantajoso.

Além dos resultados obtidos, nota-se que existe um grande espaço para aprimoramentos do Algoritmo Genético, principalmente no que diz respeito ao ajuste fino dos parâmetros — como tamanho da população, número de gerações, taxas de cruzamento, mutação e elitismo — quanto na forma de codificação das soluções candidatas. Tais direções podem potencialmente reduzir o tempo de convergência e melhorar a qualidade das soluções encontradas.

Referências

Calamoneri, T. (2006). The l(h, k)-labelling problem: A survey and annotated bibliography. *The Computer Journal*, 49(5):585–608.

CELAR (2025). Celar radio link frequency assignment instances. `http://www.cs.st-andrews.ac.uk/~ianm/graphs/`. Acesso em 2025.

Chia, M. L., Kuo, D., Yang, H. L. C. H., and Yeh, R. K. (2011). L(3,2,1) labeling of graphs. *Taiwanese Journal of Mathematics*, 15:2439–2457.

Clipperton, J., Gehrtz, J., Szaniszlo, Z., and Torkornoo, D. (2005). L(3,2,1)-labeling of simple graphs. In *Proceedings of the 36th Southeastern International Conference on Combinatorics, Graph Theory and Computing*, pages 1–14.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. MIT press, 3 edition.

Duff, I. S., Grimes, R. G., and Lewis, J. G. (2025). Harwell-boeing sparse matrix collection. `https://math.nist.gov/MatrixMarket/`. Acesso em 2025.

Florencio, D. G. and Luiz, A. G. (2021). Limites superiores para a rotulação-L(3,2,1) de grafos com grau máximo três. In *Anais do LIII Simpósio Brasileiro de Pesquisa Operacional (SBPO)*, pages 11–15, João Pessoa, Brasil. Galoá.

Gen, M. and Cheng, R. (1997). *Genetic Algorithms and Engineering Design*. John Wiley & Sons, New York.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Boston.

Griggs, J. R. and Yeh, R. K. (1992). Labelling graphs with a condition at distance 2. *SIAM Journal on Discrete Mathematics*, 5(4):586–595.

Hagberg, A. A., Schult, D. A., and Swart, P. J. (2008). Networkx: Python software for the analysis of networks. Disponível em: `https://networkx.org/`.

Hale, W. K. (1980). Frequency assignment: Theory and applications. *Proceedings of the IEEE*, 68(12):1497–1514.

Han, K. and Kim, C. (2008). Solving l(2,1)-labeling problem of graphs using genetic algorithms. *The KIPS Transactions:PartB*, 15B(2):131–136.

Hart, E., Ross, P., and Nelson, J. (2005). A survey of hybrid metaheuristics. *Studies in Computational Intelligence*, 1:1–26.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI.

Jansky, D. (1977). *Spectrum Management Techniques*. Multi-volume EMC encyclopedia series. Don White Consultants.

Johnson, D. S. and Trick, M. A. (1996). Cliques, coloring, and satisfiability: Second dimacs implementation challenge. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, volume 26. American Mathematical Society.

Mitchell, M. (1998). *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA.

Mumford, C. L. (2006). New order-based crossovers for the graph coloring problem. In Runarsson, T. P., Beyer, H.-G., Burke, E., Merelo-Guervós, J. J., Whitley, L. D., and Yao, X., editors, *Parallel Problem Solving from Nature - PPSN IX*, pages 880–889, Berlin, Heidelberg. Springer Berlin Heidelberg.

Murphey, R. A., Pardalos, P. M., and Resende, M. G. C. (1999). *Frequency Assignment Problems*, pages 295–377. Springer US, Boston, MA.

Panda, B. S. and Goel, P. (2010). Heuristic algorithms for the l(2,1)-labeling problem. *Lecture Notes in Computer Science,Swarm, Evolutionary, and Memetic Computing*, Not available:214–221.

Shao, J. and Liu, X. (2004). Labeling graphs with maximum degree three and four. *Discrete Mathematics*, 275(1-3):339–349.

Shao, Z. and Vesel, A. (2013). Integer linear programming model and satisfiability test reduction for distance constrained labellings of graphs: the case of l(3,2,1)labelling for products of paths and cycles. *IET Communications*, 7(8):715–720.