UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS RUSSAS

CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FRANCISCO DAVI GOMES DE OLIVEIRA

EXPERIMENTAÇÃO DE PARÂMETROS DE UM ALGORITMO GENÉTICO PARA O PROBLEMA DO CORTE MÁXIMO

RUSSAS

2025

FRANCISCO DAVI GOMES DE OLIVEIRA

EXPERIMENTAÇÃO DE PARÂMETROS DE UM ALGORITMO GENÉTICO PARA O PROBLEMA DO CORTE MÁXIMO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Orientador: Prof. Dr. Pablo Luiz Braga Soares.

RUSSAS

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

O47e Oliveira, Francisco Davi Gomes de.
Experimentação de parâmetros de um Algoritmo Genético para o Problema do Corte Máximo / Francisco Davi Gomes de Oliveira. – 2025.
51 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Russas, Curso de Ciência da Computação, Russas, 2025.
Orientação: Prof. Dr. Pablo Luiz Braga Soares.

1. Problema do Corte Máximo. 2. Algoritmo Genético . 3. Configuração de Parâmetros . I. Título.
CDD 005

FRANCISCO DAVI GOMES DE OLIVEIRA

EXPERIMENTAÇÃO DE PARÂMETROS DE UM ALGORITMO GENÉTICO PARA O PROBLEMA DO CORTE MÁXIMO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Aprovada em: 06/03/2025

BANCA EXAMINADORA

Prof. Dr. Pablo Luiz Braga Soares (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Eurinardo Rodrigues Costa
Universidade Federal do Ceará (UFC)

Prof. Dr. Cenez Araújo De Rezende
Universidade Federal do Ceará (UFC)

Dedico este trabalho aos meus pais, Maria José Silvia Gomes e Francisco Damião de Oliveira, a meus irmãos, e principalmente a Deus.

AGRADECIMENTOS

À minha mãe, Silvia Gomes que sempre me apoiou, aconselhou e fez de tudo o que pôde para um dia ver um de seus filhos formado; ao meu pai, Francisco Damião, que me ajuda no que for necessário sem medir esforços, e não me deixa desistir nunca; aos meus irmãos, Caique e Attilo, por serem meus exemplos mais próximos dentro da família e por partilharmos alguns conselhos. Obrigado à todos da minha familia por fazerem isso acontecer.

Aos meus amigos, que fizeram parte da minha jornada acadêmica, em especial a Leticia Kevillan, Arthur Levi, Vinicius de Assis, Isaac Chaves, entre tantos outros que estiveram ao meu lado em diversos momentos decisivos, por partilhar de ideias e conselhos, e da maravilhosa amizade, seja com um abraço ou com uma simples conversa fiada. Muito obrigado a todos!

À minha amada namorada, Vivian Hellen, por estar ao meu lado e me apoiando em minhas decisões, por me ajudar a querer ser uma pessoa melhor a cada dia e não desistir dos meus/nossos planos. Muito obrigado!

Ao Prof. Dr. Pablo Soares, por além de me orientar neste trabalho de conclusão de curso, pelo convite à fazer parte da iniciação científica que culminou neste trabalho, e ser um ser humando impar, cara extremamente gente boa, acessível e inspirador, por ter contribuído bastante durante minha trajetória acadêmica e tenho-o como uma referência em minha vida. Muito obrigado pela oportunidade!

Aos professores participantes da banca examinadora, Prof. Dr. Cenez Araújo de Rezende e Prof. Dr. Eurinardo Rodrigues Costa, meus mais sinceros agradecimentos pelo tempo dedicado, pelas críticas construtivas e pelas sugestões de enriquecimento e aprimoramento significativo para este trabalho.

“A melhor forma de prever o futuro é criá-lo."

(Abraham Lincoln)

RESUMO

Este trabalho tem como finalidade a busca por boas soluções para o problema do corte máximo (Max-Cut) em um grafo, através da utilização de uma metaheurística baseada em algoritmo genético que, por sua vez, é inspirada no princípio teórico da evolução natural proposto por Charles Darwin. A ideia principal consiste na busca por uma configuração de parâmetros de refinamento do algoritmo genético que venha a apresentar bons resultados nas instâncias da literatura. O algoritmo genético foi implementado em linguagem Python, sendo centralizado em fornecer a otimização ao problema do corte máximo para grafos. Após a implementação do algoritmo, foram testadas 54 configurações diferentes de parâmetros de ajuste do algoritmo. Foram obtidos bons resultados, com o prevalecimento da seguinte configuração: "Sem ilha, Mutação em 1 gene, Crossover que prevalece a igualdade entre os genes e busca local a partir de 20 gerações sem convergências do melhor resultado($SIM1PIBL20$)".

Palavras-chave: problema do corte máximo; algoritmo genético; configuração de parâmetros.

ABSTRACT

This work aims to search for good solutions to the problem of maximum cut (Max-Cut) in a graph, through the use of a metaheuristic based on a genetic algorithm which, in turn, is inspired by the theoretical principle of natural evolution proposed by Charles Darwin. The main idea consists of searching for a configuration of parameters for refining the genetic algorithm that will present good results in the literature. The genetic algorithm was implemented in Python language, focusing on providing optimization to the maximum cut problem for graphs. After implementing the algorithm, 54 different configurations of algorithm tuning parameters were tested. Good results were obtained, with the following configuration prevailing: "No island, Mutation in 1 gene, Crossover that prevails equality between genes and local search from 20 generations without convergence of the best result ($SIM1PIBL20$)".

Keywords: maximum cut problem; genetic algorithm; parameter configuration.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE SÍMBOLOS

V	Conjunto de Vértices
E	Conjunto de Arestas
$\|V\|$	Quantidade de Vértices
$\|E\|$	Quantidade de Arestas
S	Subconjunto de Vértices do grafo
C_{int}	Corte interno
C_{ext}	Corte externo
P	População de soluções
I	Conjunto de instâncias testadas
CP	Conjunto de combinações de parâmetros
$\in$	Pertence
$\notin$	Não pertence
$\subseteq$	Contido em
$\forall$	Para todo
$>$	Maior
$\leq$	Menor ou igual
$\geq$	Maior ou igual
$\sum$	Somatório

SUMÁRIO

1 INTRODUÇÃO

Um grafo G consiste em um conjunto de vértices ou nós (V) conectadas por um conjunto de arestas ou arcos (E), onde cada aresta deste conjunto estará associado a, no máximo, um par de vértices. Dessa forma, com os vértices u e v $(u, v \in V)$, pode-se representar uma aresta como o par (u, v) ou (v, u), onde (u, v), $(v, u) \in E$, sendo ambos os pares considerados equivalentes quando se trata de um grafo não direcionado. Além disso, quando o par $(u, v) \in E$ recebe um peso $p(u, v)$, o grafo é classificado como ponderado (Khan Academy, 2022).

Os grafos podem ser usados na representação de alguns problemas, como *mapeamento, caminho mínimo ou máximo, fluxo máximo*, dentre outros (Jurkiewicz, 2009). Entre as diversas aplicações dos grafos, destaca-se o problema do corte máximo, que consiste em encontrar a melhor configuração/partição dos vértices de V em dois subconjuntos, S e S', de forma que a soma dos pesos das arestas que ligam S e S' seja maximizada (Soares, 2018). Por ser classificado como um problema NP-difícil (Karp, 1972), a obtenção exata do corte máximo em instâncias de tamanho moderado a grande se torna inviável, exigindo a utilização de métodos que forneçam soluções aproximadas em tempo computacional razoável.

Diversos estudos têm abordado o problema do corte máximo a partir de diferentes perspectivas. Enquanto abordagens exatas, como a apresentada por Lima *et al.* (2021), onde demonstram a viabilidade de métodos determinísticos para instâncias menores, métodos de aproximação – conforme discutido em Gosti *et al.* (1995) e Simone (1992) – oferecem estratégias para alcançar bons resultados em tempo reduzido para instâncias maiores. Além disso, abordagens incrementais, como expostas por Kim *et al.* (2016) evidenciam que o aprimoramento gradual das soluções pode ser uma alternativa promissora. Por fim, técnicas de busca local avançadas, tais como a utilizada por Benlic e Hao (2013) e métodos híbridos que combinam algoritmos evolutivos com intensificadores de busca (Bansal *et al.*, 2012), e em análises de desempenho como em Zhou *et al.* (2014), reforçam a importância da integração de estratégias exploratórias e intensificadoras para enfrentar os desafios do problema do corte máximo.

Nesse contexto, os algoritmos genéticos emergem como uma poderosa meta-heurística, inspirada nos princípios da seleção natural e evolução biológica propostos por Darwin (Lucas, 2002). Esses algoritmos exploram o espaço de soluções aplicando operadores como mutação e crossover, permitindo a geração de novas soluções sem a necessidade de avaliar exaustivamente todas as soluções possíveis (Goldbarg; Luna, 2005).

Além dos métodos clássicos de algoritmos genéticos, a incorporação de estratégias

híbridas tem se mostrado crucial para superar limitações como a convergência prematura (Talbi, 2009). Nesse contexto, os algoritmos meméticos – que combinam a exploração global dos algoritmos genéticos com a intensificação de busca local – evidenciam resultados superiores em problemas NP-difíceis, como o problema do corte máximo, conforme demonstrado em estudos que exploram técnicas inspiradas em métodos como o Tabu Search (Glover, 1989). Além disso, a integração de técnicas de busca local é apontada como um fator determinante para aprimorar a eficiência computacional e superar ótimos locais, fundamentando-se em princípios discutidos na obra de Eiben e Smith (2015). Dessa forma, o acoplamento entre mecanismos exploratórios e intensificadores se consolida como uma estratégia essencial para o desenvolvimento de sistemas robustos e adaptativos.

O presente trabalho propõe uma abordagem para a configuração de parâmetros em algoritmos genéticos aplicados ao problema do corte máximo. A ideia central consiste em ajustar a estratégia de busca e filtragem de soluções de forma a obter, de maneira otimizada, resultados que se aproximem dos valores ótimos teóricos. A metodologia adotada inclui a definição de uma população inicial, a aplicação de operadores genéticos, e a integração de mecanismos adicionais, como a verificação de restrições e o uso de *"ilhas"* para idenitificar soluções promissoras.

O restante deste trabalho está estruturado da seguinte forma: na Seção 2, comenta-se sobre os objetivos; na Seção 3, são apresentadas as definições e os conceitos teóricos que fundamentam o problema do corte máximo, os algoritmos genéticos e algoritmos híbridos; na Seção 4, descreve-se a metodologia adotada para o desenvolvimento e a otimização do algoritmo; na Seção 5, os experimentos computacionais aplicados; na Seção 6, apresenta-se os resultados obtidos; e, por fim, na Seção 7, são apresentadas as conclusões deste trabalho.

2 OBJETIVOS

Nesta seção, serão discutidos o principal objetivo deste trabalho, bem como os objetivos específicos que orientarão o desenvolvimento e a avaliação da proposta.

2.1 Objetivos Gerais

O objetivo geral deste trabalho é parametrizar um algoritmo genético para o problema do corte máximo em grafos, de forma a obter soluções de alta qualidade com custos computacionais reduzidos.

2.2 Objetivos Específicos

- Analisar e definir os principais parâmetros que influenciam o desempenho do algoritmo genético no problema do corte máximo;
- Desenvolver uma metodologia para a criação e avaliação da população, incorporando operadores genéticos (mutação, crossover, etc.);
- Integrar uma heurística de busca local e um mecanismo de filtragem por restrições para identificar soluções promissoras;
- Realizar experimentos computacionais com diferentes combinações de parâmetros e comparar os resultados com valores ótimos teóricos;
- Identificar a configuração de parâmetros que maximize a aproximação aos valores ótimos e discutir a influência individual de cada parâmetro.

3 FUNDAMENTAÇÃO TEÓRICA

Nesta seção, serão apresentados os principais conceitos que fundamentam este trabalho. Na Seção 3.1, apresenta-se o Problema do Corte Máximo, uma problemática na área de grafos que este trabalho vai abordar. Em seguida, na Seção 3.2, será apresentado o Algoritmo Genético, uma meta-heurística utilizada na busca por um espaço de soluções para o problema abordado. E por último, na Seção 3.3, apresenta-se o Algoritmo Híbrido, assim como a defição da heurística de Busca Local, uma etapa extra que será empregada ao nosso algoritmo.

3.1 Definição do problema de Corte Máximo

Seja $G(V,E)$ um grafo não direcionado e ponderado, onde V representa o conjunto de vértices, com $|V| = n$ vértices, e E representa o conjunto das arestas, com $|E| = m$ arestas. Para cada aresta (u,v) ou (v,u), será atribuído o peso $p(u,v)$ apenas para $u,v \in E$, $u \neq v$. Será atribuído valor 0 ao peso da aresta quando: $u,v \notin E$, e $p(u,u)$ para todo $1 \leq u \leq n$ (Soares, 2018). Devido à ampla gama de aplicações e pesquisas relacionadas ao problema, o mesmo é encontrado na literatura com diferentes denominações, dentre elas as mais comuns são *Maximum-2-Satisfiability*, *Weighted Signed Graph Balancing*, *Unconstrained Quadratic 0 - 1 Programming* (Boros; Hammer, 1991).

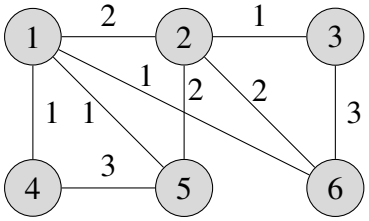
O problema do Corte Máximo (Max-Cut) ponderado consiste em encontrar o peso de valor máximo no somatório dos pesos das arestas que estejam ligando os subconjuntos da bipartição S e S', com a melhor configuração possível na distribuição das arestas entre esses subconjuntos. Assim, o subconjunto S contém uma parte dos vértices de V, enquanto S' contém os $|V| - |S|$ vértices restantes. Vale ressaltar que S ou S' podem ser conjuntos vazios (Soares, 2018). A equação que representa o somatório dos pesos $p(u,v)$ das arestas que ligam os subconjuntos S e S', obtendo o valor de corte máximo é definida da seguinte forma:

$$CM(G) = \sum_{\substack{\forall u \in S \\ \forall v \in S'}} p_{(u,v)} \tag{3.1}$$

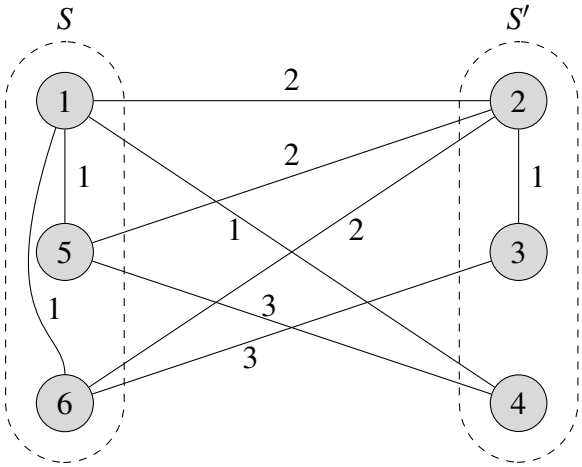
Na Figura 1, temos um exemplo do Corte Máximo em um grafo não direcionado e ponderado. Na Figura 1 (a), utilizamos um grafo com $n = 6$ vértices e $m = 9$ arestas com os pesos associados. Na Figura 1 (b), podemos observar a bipartição desses vértices entre S e S', de tal maneira que $\{1,5,6\} \subseteq S$ e $\{2,3,4\} \subseteq S'$. Com esta distribuição dos vértices, ao aplicar a

Equação 3.1 nas arestas que cruzam os subconjuntos S e S', obteremos um valor de corte igual a 13, que será o valor máximo a ser obtido neste grafo.

Figura 1 – Exemplo de um grafo não direcionado e ponderado (a) e o corte máximo aplicado a este mesmo grafo (b).



(a) Grafo não direcionado e ponderado



(b) Corte Máximo do grafo de exemplo

Fonte: elaborado pelo autor.

Em aplicações práticas, o problema do corte máximo é utilizado em diversas áreas, tais como a de *Redes de comunicação*, onde a separação de redes em clusters pode otimizar o tráfego de dados (Lima, 2022); *Agrupamentos de dados* (*Clustering*), onde os grafos são utilizados para dividir conjuntos de dados em grupos distintos para análise de padrões (Costa, 2014); *Segmentação de imagens*, onde o particionamento do grafo que modela a imagem facilita a identificação dos segmentos (Sousa *et al.*, 2013); dentre outras diversas aplicações da literatura. O problema do Corte Máximo é uma importante ferramenta para modelagem e solução de diversos desafios práticos relacionados à teoria dos grafos.

Devido à sua complexidade computacional, o problema do Corte Máximo é tratado por diferentes abordagens, que podem ser classificadas em *Algoritmos Exatos*, como *programação linear inteira* e métodos *branch-and-bound* que garantem a obtenção de soluções ótimas, porém com alto custo computacional, tornando-se inviáveis para grandes instâncias; *Heurísticas*,

como a *busca local* e *simulated annealing*, que buscam soluções aproximadas em tempo reduzido, porém sem garantir otimalidade; e *Metaheurísticas*, como *algoritmos genéticos* e *busca tabu*, que utilizam técnicas inspiradas na biologia ou em métodos de otimização combinatória para encontrar boas soluções aproximadas em problemas complexos.

Para Karp (1972), o problema do Corte Máximo (Max-Cut) é classificado como um problema NP-Difícil, o que leva a ser um desafio computacional para a obtenção do valor de corte máximo em instâncias de tamanho intermediários, como grafos com aproximadamente 50 vértices. Dessa forma, para instâncias maiores, é necessário a aplicação de abordagens heurísticas e metaheurísticas que forneçam soluções com ótima qualidade e ainda em tempo computacional aceitável.

3.2 Algoritmo Genético

O Algoritmo Genético é uma metaheurística de otimização inspirada na teoria evolucionária de Charles Darwin. Segundo essa teoria, os indivíduos mais adaptados ao ambiente possuem maior chance de reprodução, transmitindo suas características para as próximas gerações. No contexto computacional, essa ideia é traduzida em um processo iterativo de seleção, reprodução e mutação de soluções, permitindo que as melhores configurações sejam mantidas e aprimoradas ao longo do tempo (Lucas, 2002).

A evolução das espécies pode ser interpretada como um processo de otimização das espécies, ao passo que, no decorrer do tempo, os seres vivos vão se adaptando em um ambiente que evolui constantemente (Lucas, 2002). Portanto, o algoritmo genético tem uma estrutura em que as informações de cada sistema podem ser modeladas e processadas analogamente aos cromossomos biológicos (Ikeda, 2009).

Diversos cientistas contribuíram para a consolidação da teoria dos Algoritmos Genéticos, servindo aspectos importantes para a aproximação da teoria de evolução das espécies com o contexto computacional. Em 1809, Lamarck propôs uma hipótese evolucionária na qual os organismos adquirem características ao longo do tempo, o que pode ser associado ao mecanismo de mutação em Algoritmos Genéticos. Em 1866, Mendel estabeleceu as bases da genética moderna, mostrando que características são transmitidas hereditariamente, refletindo diretamente no operador de crossover utilizado em Algoritmos Genéticos. Já em 1975, John Holland foi o pioneiro na modelagem computacional da evolução, ao representar um cromossomo como uma sequência binária $(0, 1)$, permitindo a aplicação de operadores genéticos como mutação,

crossover e seleção para encontrar soluções ótimas em problemas de otimização (Goldbarg; Luna, 2005).

O Algoritmo Genético fundamenta-se em uma concepção generalista do processo adaptativo, permitindo ao modelador a flexibilidade de definir a estratégia de seleção natural e evolução de acordo com as particularidades do problema em questão. Essa flexibilidade se reflete na abstração do modelo, na qual o significado das cadeias cromossômicas é livre e adaptável a diferentes contextos. Em essência, um algoritmo genético requer a definição de premissas fundamentais para seu desenvolvimento, a saber:

- A determinação da população inicial de cromossomos, que representa as soluções candidatas ao problema;
- A definição de uma função de avaliação (*fitness*) que quantifique a qualidade das soluções;
- A especificação dos operadores genéticos, tais como crossover e mutação, que viabilizam a reprodução e a geração de novos indivíduos, incluindo a escolha do ponto de quebra do cromossomo e o tipo de mutação aplicado;
- A definição de parâmetros críticos para o fluxo do algoritmo, como condições de parada, tamanho da população e mecanismos para garantir a diversidade.

Essa abordagem generalista, que permite ao modelador concretizar o processo adaptativo conforme a situação, é justamente o que confere aos algoritmos genéticos sua natureza de metaheurísticas (Goldbarg; Luna, 2005).

O pseudo-algoritmo que representa o fluxo de um Algoritmo Genético pode ser definido da seguinte forma no algoritmo 1:

Algoritmo 1: Pseudo-Algoritmo Genético

Entrada: Um Grafo não direcionado e ponderado

Saída: Solução

início

  Gerar uma população inicial ;

  Avaliar o *fitness* dos indíviduos da população inicial;

  enquanto *Não atingir o critério de parada* **fazer**

    Seleção de pais da população;

    Aplicar crossover ou mutação nos pais para estimular a reprodução;

    Avaliar o *fitness* dos filhos gerados;

    Ordenar todos os indivíduos da população;

    Selecionar os melhores para permanecer para a próxima geração;

  fim

fim

Os Algoritmos Genéticos têm sido amplamente aplicados em diversas áreas, evidenciando sua versatilidade na resolução de problemas de otimização combinatória. Entre as aplicações práticas destacadas encontram-se o agendamento de processos em sistemas multiprocessadores (*Multiprocessor Scheduling*) — cujo o objetivo é reduzir os custos de comunicação em arquiteturas paralelas —, a modelagem de problemas em biologia molecular e físico-química, a otimização discreta de estruturas na engenharia, a inversão de formas de ondas sísmicas em geofísica, a integração com redes neurais para aprimoramento de classificadores e a compressão de dados, especialmente na codificação de imagens. Essas diversas aplicações não apenas ilustram a eficácia dos Algoritmos Genéticos na obtenção de soluções aproximadas para problemas complexos, mas também motivam a contínua melhoria dos métodos de seleção, mutação e cruzamento, bem como a integração dos Algoritmos Genéticos com outras técnicas de otimização, como *Simulated Annealing* e *Tabu Search* (Tsuruta; Narciso, 2000).

3.2.1 Mapeamento de conceitos

A proximidade ao processo biológico reflete nos Algoritmos Genéticos em métodos e mecanismos de seleção e evolução natural. A seguir, será feito o mapeamento dos conceitos para **gene**, **cromossomo**, **fitness**, **população**, **seleção**, **mutação** e **crossover**, utilizados no algoritmo genético junto ao problema do corte máximo aplicados nesta pesquisa.

- ***Gene:*** Unidade mínima da representação de uma solução do algoritmo genético. Em

termos computacionais e associado ao problema abordado, é uma posição do vetor que representa cada vértice do grafo. E nesta, pode estar preenchida com a valoração de 0 ou 1, onde 0 representa que o vértice está contido no subconjunto S e 1 representa que o vértice está contido no subconjunto S';

- ***Cromossomo:*** Sequência de genes que representam uma solução candidata ao problema. Ou seja, é uma sequência binária que representa a configuração de como estão particionados os vértices entre os subconjuntos S e S'.
- ***Fitness:*** Valor associado a aptidão de cada indivíduo/cromossomo;
- ***População:*** Conjunto de cromossomos que evoluem ao longo das gerações. Ou ainda, uma matriz com configurações de possíveis soluções.
- ***Seleção***: Método de escolha dos melhores indivíduos para reprodução. O método utilizado nesse trabalho é *elitista* e *uniforme*, onde garante a sobrevivência dos melhores cromossomos ao ser feito a ordenação e, a garantia para qualquer indivíduo poder ser escolhido sem a dependência da análise ao seu valor de *fitness*;
- ***Crossover (Cruzamento):*** Método de combinação de cromossomos pais para gerar novas soluções, ou ainda, obter novos filhos com o cruzamento dos genes pais, podendo ser de um ponto, múltiplos pontos ou uniforme;
- ***Mutação***: Método em que se aplica uma pequena alteração aleatória nos genes para diversificar as soluções e evitar convergência prematura.

3.3 Algoritmos Híbridos

Apesar do caráter genérico e versátil dos Algoritmos Genéticos na resolução de problemas de otimização combinatória, diversos estudos apontam que a forma convencional desses algoritmos pode não ser a solução mais eficiente para problemas complexos. Isso ocorre, em parte, devido à representação binária dos cromossomos e aos processos aleatórios empregados na seleção, cruzamento e mutação, que podem levar à formação prematura de populações homogêneas e, consequentemente, dificultar a exploração de ótimos locais (Ikeda, 2009). Essa limitação tem motivado a proposição de algoritmos híbridos, que combinam a capacidade exploratória dos Algoritmos Genéticos com técnicas de intensificação, como a busca local, para melhorar o desempenho global na obtenção de soluções.

Em nossa abordagem, a estratégia híbrida integra o algoritmo genético a uma busca local direcionada, aplicada em cromossomos pré-definidos do conjunto de soluções. A busca

local, nesse contexto, atua como um operador intensificador, refinando os melhores indivíduos quando ocorre um período de estagnação na evolução da população. Essa técnica permite ajustes finos na solução, ajudando a superar os desafios dos ótimos locais e garantindo uma maior robustez na solução final. Assim, a incorporação da busca local exemplifica o equilíbrio entre o caráter genérico dos Algoritmos Genéticos convencionais e as vantagens de métodos especializados, como os propostos em algoritmos meméticos (Moscato *et al.*, 2004), e outras metaheurísticas.

3.3.1 Busca Local

A Busca Local é uma técnica clássica de otimização combinatória que explora a vizinhança de uma solução corrente para encontrar melhorias incrementais na função objetivo. Fundamentanda nos princípios do "*Hill Climbing*", a abordagem consiste em realizar pequenas modificações na solução atual e adotar aquela que apresenta melhoria, repetindo esse processo iterativamente. Em versões mais sofisticadas, como o *simulated annealing*, o método permite a aceitação temporária de soluções inferiores para escapar de ótimos locais. Esse método, pela sua simplicidade e eficácia, é amplamente utilizado para resolver problemas NP-difíceis, onde a busca exaustiva se torna inviável (Hoos; Stutzle, 2018).

Essa estratégia híbrida, frequentemente denominada algoritmo memético, tem sido validada na literatura internacional (Moscato *et al.*, 2004) e em estudos nacionais (Benvenga, 2022) como uma forma eficaz de melhorar a qualidade final das soluções em problemas de otimização combinatória. Em ambientes híbridos, a busca local é frequentemente integrada a algoritmos globais, como os algoritmos genéticos, para intensificar a exploração em pontos estratégicos do espaço de soluções.

Segundo Filho (2021), a análise dos potenciais dos vértices – divididos em cálculos de **corte interno** e **corte externo** – torna-se uma estratégia fundamental para orientar a busca local no problema do corte máximo. Em nossa abordagem, para cada vértice, são calculados os seguintes valores:

- O **corte interno** (C_{int}), que representa a soma dos pesos das arestas que conectam o vértice aos demais vértices do mesmo subconjunto
- O **corte externo** (C_{ext}), que representa a soma dos pesos das arestas que ligam o vértice aos vértices do subconjunto oposto.

A diferença qualitativa entre essas duas medidas fornece uma indicação do benefício

potencial em alterar a atribuição do vértice. Se a conectividade interna for consideravelmente superior à externa, isso sugere que mover o vértice para o conjunto oposto poderá incrementar o valor total do corte. Assim, ao identificar vértices com forte predominância de conexões internas, o algoritmo opta por realocá-los, refinando progressivamente a solução global.

Em nosso método, a Busca Local é aplicada em posições previamente determinadas da solução, refinando os melhores indivíduos e contribuindo para a superação de estagnações na evolução das soluções. Para melhorar os ótimos locais, o algoritmo de Busca Local percorre toda a solução fornecida e avalia o potencial de cada gene do cromossomo (Filho, 2021). Calculando, para cada gene, o valor de corte interno (C_{int}) e o valor de corte externo (C_{ext}). Sempre que $C_{int} > C_{ext}$, o gene é alterado, promovendo sua migração para o outro subconjunto, e o cromossomo modificado é reavaliado quanto ao seu *fitness*. As novas soluções resultantes são, então, inseridas na população principal, onde o processo de ordenação *Insertion Sort* garante que apenas os indivíduos com melhores valores de aptidão sejam encaminhados para a próxima geração. O fluxo da lógica de Busca Local pode ser visto no algoritmo 2, logo abaixo:

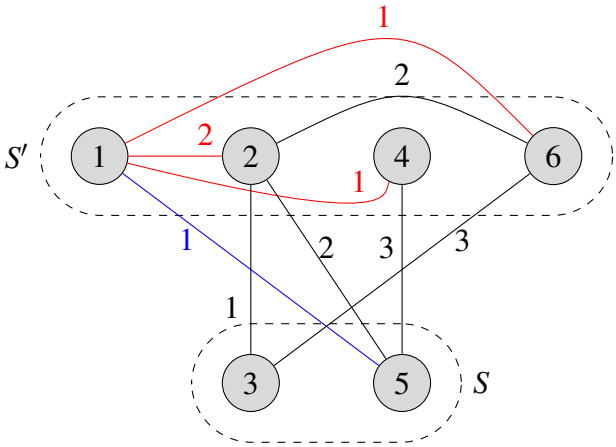
Algoritmo 2: Pseudo-Algoritmo Busca Local

```
Entrada: Uma solução do espaço de soluções
Saída: Soluções com melhorias
início
    Inicialização e avaliação do cromossomo;
    para cada gene faça
        Calcular os valores C_int e C_ext;
        se C_int > C_ext então
            Altera o valor do gene;
            Avaliação do fitness;
            Adiciona essa solução com melhoria à população;
        fim
    fim
    Aplica ordenação Insertion Sort;
fim
```

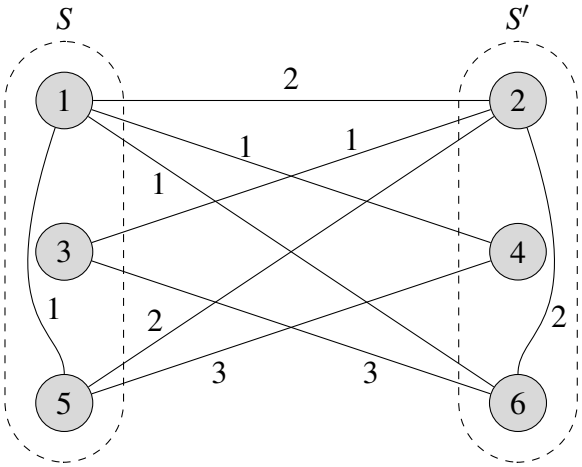
Exemplo: Considere a partição ilustrada na Figura 2 (a), onde os vértices são divididos em $S = \{3,5\}$ e $S' = \{1,2,4,6\}$, e o valor de aptidão é igual a 10. Partindo da verificação do vértice 1, que tem um potencial total igual a 5, com $C_{int}(1) = 4$ (devido às conexões internas com 2, 4 e 6) e $C_{ext}(1) = 1$ (relacionado à conexão com o vértice 5), observa-

se uma predominância interna que sugere que a mudança da atribuição do vértice 1 pode melhorar o corte global. Essa mudança é vista na Figura 2 (b), e sua configuração tem um valor de aptidão igual a 13.

Figura 2 – Exemplo de uma partição dos vértices do grafo (a) e uma nova partição obtida após a busca local aplicando mudança do vértice 1 (b).



(a) Partição com $S = \{3,5\}$ e $S' = \{1,2,4,6\}$



(b) Partição com $S = \{1,3,5\}$ e $S' = \{2,4,6\}$

Fonte: elaborado pelo autor.

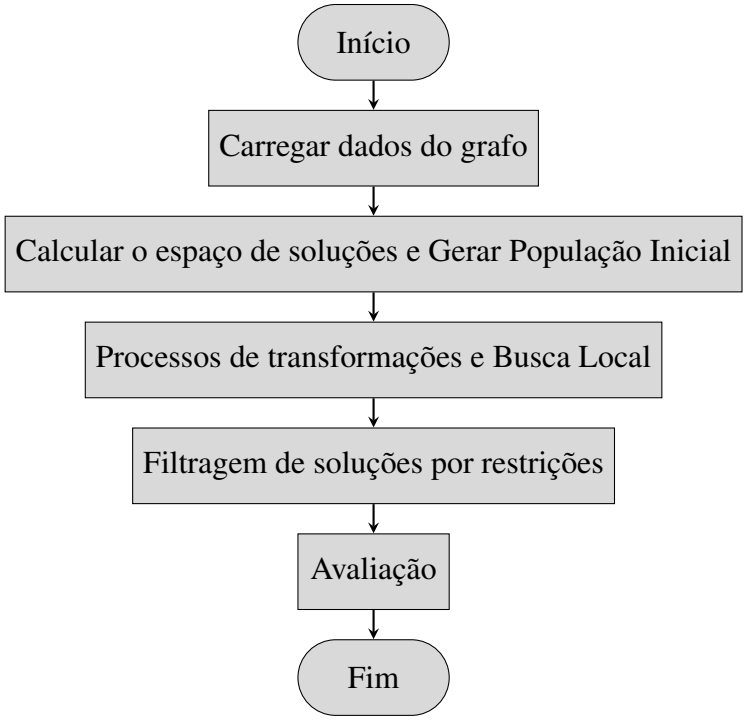
Portanto, a aplicação da Busca Local dentro do Algoritmo Genético traz diversos benefícios, como a melhoria da eficiência computacional, pois opera apenas com modificações na solução atual, reduzindo a necessidade de buscas exaustivas (Goldbarg; Luna, 2005). Além disso, esse refinamento local contribui para a qualidade das soluções ao escapar de ótimos locais, permitindo que indivíduos mais aptos sejam preservados (Eiben; Smith, 2015). Outra vantagem é o equilíbrio entre exploração e explotação, já que enquanto o Algoritmo Genético busca diversidade globalmente, a Busca Local aprofunda a investigação em regiões promissoras do espaço de soluções. Por fim, essa abordagem reduz a convergência prematura ao impedir que a população fique presa em soluções subótimas, favorecendo a obtenção de resultados mais robustos e globalmente competitivos.

4 METODOLOGIA

Esta seção apresenta a metodologia adotada no desenvolvimento deste trabalho, que consiste na aplicação do Algoritmo Genético para o tratamento do Problema de Corte Máximo.

O desenvolvimento do Algoritmo Genético, com representação da solução em binário, consiste em um método simples e de fácil implementação. À princípio, a implementação foi realizada em linguagem de programação *C*, e utilizando o ambiente de desenvolvimento Code Blocks. Mas em seguida, este trabalho foi migrado para a linguagem de programação *Python*, e programada no ambiente de desenvolvimento Visual Studio Code. O procedimento adotado para solucionar o problema do Corte Máximo de um grafo ponderado foi dividido nos seguintes passos que serão apresentados nas Seções durante este capítulo: Carregamento dos dados do Grafo (4.1); Cálculo do espaço de soluções e criação da população inicial (4.2); Processos de transformações e Busca Local (4.3); Filtragem de soluções por restrições (4.4); Avaliação final e retorno dos resultados (4.5). Abaixo está o fluxograma representando o fluxo do algoritmo abordado neste capítulo:

Figura 3 – Fluxograma do algoritmo



Fonte: elaborado pelo autor.

4.1 Carregamento dos dados do Grafo

O processo de carregamento dos dados do Grafo e sua posterior criação, porta-se de maneira simples e concisa. O algoritmo inicia carregando um conjunto de arestas de um grafo G ponderado a partir de um arquivo de texto, e em seguida parte para a criação do mesmo.

Na Figura 4, podemos ver como os dados do Grafo vão ser estruturados no arquivo de texto. Os dados da figura são referentes ao Grafo apresentado na Figura 1 (a). O arquivo possui, na primeira linha, a quantidade de vértices (n) e a quantidade de arestas (m), respectivamente. Nas linhas subsequentes, são fornecidas as informações referentes a cada aresta, onde o primeiro dado indica o vértice u, o segundo dado indica o vértice v e, por último, o tercerio remete ao peso da aresta (u,v).

Figura 4 – Exemplo da estrutura do arquivo de texto contendo os dados do Grafo

```
6 9
1 2 2
1 4 1
1 5 1
1 6 1
2 3 1
2 5 2
2 6 2
3 6 3
4 5 3
```

Fonte: elaborado pelo autor.

4.2 Cálculo do espaço de soluções e criação da população inicial

Após a leitura e processamento inicial dos dados do grafo G, será calculada a quantidade máxima de soluções possíveis para o determinado grafo, onde esse número é obtido na seguinte equação:

$$TS(G) = 2^n \tag{4.1}$$

tendo em vista que, n é a quantidade de vértices do grafo, e para cada vértice pode ser atribuído um dos dois valores para indicar a qual subconjunto o vértice vai pertencer (0 ou 1).

Em seguida, será criada a população inicial com um tamanho limitado t, definido a partir do método para a criação da população, considerando que o número total de soluções possa ter um crescimento de forma exponencial, a depender de n. Essa função de criação da população é definida no algoritmo 3, como veremos a seguir.

Algoritmo 3: Cria População

```
Entrada: n; TS(G)
Saída: populacao; tamPop
início
    totalPop ← TS(G);                    // Define totalPop a partir de TS(G)
    tamCromossomo ← n;
    se tamCromossomo < 3 então
        tamPop ← totalPop;
    senão se tamCromossomo < 10 então
        tamPop ← ⌊totalPop/2⌋;
    senão se tamCromossomo ≤ 14 então
        tamPop ← ⌊totalPop/4⌋;
    senão se tamCromossomo < 100 então
        tamPop ← 5000;
    senão
        tamPop ← 7000;
    fim
    populacao ← matriz de dimensão (tamPop × tamCromossomo);
    para i ← 0 até tamPop faça
        para j ← 0 até tamCromossomo faça
            populacao[i][j] ← aleatorio(0,1);
        fim
    fim
    retorna populacao,tamPop
fim
```

Logo após a criação da população, um vetor contendo os valores de corte (*fitness*) é gerado para cada cromossomo dessa população. Esse vetor vai ser utilizado para auxiliar a ordenação da população, utilizando o método de *Insertion Sort*. Paralelamente, a função vai ordenar o vetor com os valores de corte dos cromossomos e a matriz da população, de forma que os indivíduos com os maiores valores de corte sejam priorizados em ordem decrescente.

No procedimento do algoritmo, será criado uma nova matriz e um vetor, ambos de tamanho $t + \frac{t}{2}$, que servirão para auxiliar no armazenamento das novas populações resultantes das transformações que irão sofrer.

4.2.1 Fitness

Na formulação do problema do corte máximo, o objetivo é encontrar uma partição dos vértices do grafo de modo que a soma dos pesos das arestas que conectam os dois subconjuntos seja maximizada. Para refletir essa meta, a função de avaliação (*fitness*) foi construída de forma direta. Em nossa abordagem, cada cromossomo $c \in P$ representa uma possível partição dos vértices, onde P é o conjunto de todos os indíviduos (ou soluções candidatas) que compõem a população. Geralmente, a atribuição de 0 indica que um vértice pertence ao subconjunto S e 1 indica que pertence a S'. Assim, a função F é definida como a soma dos pesos $p(u,v)$ tais que $u \in S$ e $v \in S'$:

$$F(c) = \sum_{\substack{\forall u \in S \\ \forall v \in S'}} p_{(u,v)} \tag{4.2}$$

Dessa forma, maximizar $F(c)$ equivale a maximizar o corte do grafo. Quanto maior o valor de $F(c)$, melhor a partição obtida, pois ela resulta em uma maior soma dos pesos das arestas que cruzam os dois subconjuntos. Essa aproximação entre a função *fitness* e o problema do corte máximo garante que a metaheurística esteja alinhada com o objetivo central da otimização, utilizando a população P como conjunto de soluções para explorar o espaço de possibilidades.

4.3 Processos de transformações e Busca Local

Esse processo de transformações pode ser também intitulado por *Período de Gerações*, que ocorre logo após a adaptação do grafo G na metaheurística utilizada, e seu processo de criação e avaliação da população inicial.

Basicamente, ao entrar nesse *Período de Gerações*, essa população criada inicialmente vai passar por uma quantidade pré-definida de gerações, durante as quais são aplicadas operações de *Crossover* (4.3.1) e *Mutação* (4.3.2). A escolha da operação vai depender da probabilidade de cada transformação, visando a convergência da população para melhores soluções.

4.3.1 Crossover

Quando a variável de probabilidade de transformação prevalece para o *Crossover*, ocorre a *Seleção* aleatória de dois cromossomos pais para que seja feita a combinação de genes dos mesmos, com finalidade de obter novas ótimas soluções. Esse processo de combinação será repetido até que sejam gerados $\frac{t}{2}$ filhos.

Após a geração desses novos filhos, os mesmos serão inseridos na matriz auxiliar para a nova população e submetidos ao cálculo para medir os seus valores de *fitness*. Em seguida, serão ordenados pelo método de *Insertion Sort*.

Neste trabalho, foram elaboradas 3 técnicas diferentes para o *Crossover*, sendo elas: *Troca simples de 2 genes* (4.3.1.1); *Percentual* (4.3.1.2); *Prevalece igualdade* (4.3.1.3).

4.3.1.1 Crossover Troca simples de 2 genes

Nesta técnica, realiza-se a troca simples de dois genes entre dois cromossomos pais. Inicialmente, os dois genes são escolhidos de forma aleatória, garantindo que não sejam repetidos. A combinação é efetuada por meio da troca dos genes correspondentes entre os pais, gerando dois novos indivíduos que herdam características de ambos. O algoritmo 4 ilustra essa técnica.

Algoritmo 4: Crossover troca 2 genes

Entrada: pai_1; pai_2; $tamCromossomo$; $probCruzamento$; $probMutacao$

Saída: $filho_1$; $filho_2$; $probCruzamento$; $probMutacao$

início

$gene_1, gene_2 \leftarrow tamCromossomo + 1, tamCromossomo + 1$;

enquanto $gene_1 \geq tamCromossomo$ **ou** $gene_2 \geq tamCromossomo$ **fazer**

$gene_1, gene_2 \leftarrow$ sorted(random.sample(range(tamCromossomo), 2));

se $gene_1 = gene_2$ **então**

$gene_1, gene_2 \leftarrow tamCromossomo + 1, tamCromossomo + 1$;

fim

fim

$filho_1 \leftarrow pai_1$;

$filho_2 \leftarrow pai_2$;

$filho_2[gene_1] \leftarrow pai_1[gene_1]$;

$filho_2[gene_2] \leftarrow pai_1[gene_2]$;

$filho_1[gene_1] \leftarrow pai_2[gene_1]$;

$filho_1[gene_2] \leftarrow pai_2[gene_2]$;

$probCruzamento \leftarrow probCruzamento - (probCruzamento \times 0.03)$;

$probMutacao \leftarrow probMutacao + (probMutacao \times 0.05)$;

retorna $filho_1, filho_2, probCruzamento, probMutacao$;

fim

4.3.1.2 Crossover Percentual

Na técnica de crossover percentual, um ponto de divisão é definido aleatoriamente, escolhendo-se uma quantidade de genes (variando de 1 até o tamanho do cromossomo menos 1) a ser combinada entre os pais. Dessa forma, para cada par de cromossomos selecionados, o primeiro filho recebe os primeiros genes de um pai e o restante do outro, enquanto o segundo filho recebe a combinação inversa. Esse procedimento gera uma mescla proporcional dos genes de ambos os progenitores, preservando características relevantes dos pais e promovendo a diversidade na população. No algoritmo 5 é demonstrado a lógica desta técnica.

Algoritmo 5: Crossover Percentual

Entrada: pai_1; pai_2; $tamCromossomo$; $probCruzamento$; $probMutacao$

Saída: $filho_1$; $filho_2$; $probCruzamento$; $probMutacao$

início

$qtdGenes \leftarrow$ random.randint$(1, tamCromossomo - 1)$;

para $i \leftarrow 0$ **até** $qtdGenes$ **faça**

$filho_1[i] \leftarrow pai_1[i]$;

$filho_2[i] \leftarrow pai_2[i]$;

fim

para $j \leftarrow qtdGenes + 1$ **até** $tamCromossomo$ **faça**

$filho_1[j] \leftarrow pai_2[j]$;

$filho_2[j] \leftarrow pai_1[j]$;

fim

$probCruzamento \leftarrow probCruzamento - (probCruzamento \times 0.03)$;

$probMutacao \leftarrow probMutacao + (probMutacao \times 0.05)$;

retorna $filho_1, filho_2, probCruzamento, probMutacao$;

fim

4.3.1.3 Crossover Prevalece Igualdade

No desenvolvimento da proposta, foi implementado um operador de crossover especializado — denominado "crossover que prevalece os genes iguais" — para combinar soluções de forma a preservar características comuns entre os pais. Para cada posição do cromossomo, é verificado se os genes dos dois pais são idênticos. Caso sejam iguais, o gene é mantido nos descendentes; caso contrário, o valor do gene é sorteado entre as possíveis alternativas (0 ou 1).

Essa estratégia visa manter os traços genéticos de alta qualidade que se repetem entre os pais, intensificando a exploração de regiões promissoras do espaço de busca. Após a recombinação, as probabilidades de crossover e mutação são ajustadas para equilibrar a exploração e a intensificação da busca. A sua implementação é dada a seguir no algoritmo 6:

Algoritmo 6: Crossover Prevalece Igualdade

Entrada: pai_1; pai_2; $tamCromossomo$; $probCruzamento$; $probMutacao$

Saída: $filho_1$; $filho_2$; $probCruzamento$; $probMutacao$

início

  para $i \leftarrow 0$ **até** $tamCromossomo$ **faça**

    se $pai_1[i] = pai_2[i]$ **então**

      $filho_1[i] \leftarrow pai_1[i]$;

      $filho_2[i] \leftarrow pai_2[i]$;

    senão

      $filho_1[i] \leftarrow$ **aleatorio**$(0,1)$;

      $filho_2[i] \leftarrow$ **aleatorio**$(0,1)$;

  fim

  $probCruzamento \leftarrow probCruzamento - (probCruzamento \times 0.03)$;

  $probMutacao \leftarrow probMutacao + (probMutacao \times 0.05)$;

  retorna $filho_1, filho_2, probCruzamento, probMutacao$;

fim

4.3.2 Mutação

O processo para a aplicação da *Mutação* é análoga ao processo do *Crossover* (4.3.1). Porém, na *Seleção* será escolhido apenas um cromossomo pai por vez, esse processo irá se repetir até que sejam gerados $\frac{t}{2}$ novos filhos. Em seguida, esses novos indíviduos gerados, também irão passar pelo processo de inserção na matriz auxiliar para a nova população, cálculo do valor de *fitness*, e ordenados posteriormente com a função de *Insertion Sort*.

Para esta transformação, foram desenvolvidas 3 técnicas diferentes para a *Mutação*, sendo elas: *Mutação em 1 gene* (4.3.2.1); *Mutação em 2 genes* (4.3.2.2); *Mutação em vários genes* (4.3.2.3).

4.3.2.1 Mutação em 1 gene

Na estratégia de mutação de um gene, para cada cromossomo selecionado da população, um único gene é escolhido de forma aleatória e seu valor é invertido (por exemplo, de 0 para 1 ou vice-versa). Além disso, as probabilidades de cruzamento e de mutação são ajustadas, aumentando ligeiramente a probabilidade de cruzamento e diminuindo a de mutação, de forma a equilibrar a exploração e a intensificação do algoritmo.

Algoritmo 7: Mutação em 1 gene

Entrada: pai; $tamCromossomo$
Saída: $filho$
início
  $gene \leftarrow \text{random.randint}(0, tamCromossomo - 1)$;
  $filho \leftarrow pai$;
  se $pai[gene] = 0$ **então**
    $filho[gene] \leftarrow 1$;
  senão se $pai[gene] = 1$ **então**
    $filho[gene] \leftarrow 0$;
  fim
  $probCruzamento \leftarrow probCruzamento + (probCruzamento \times 0.05)$;
  $probMutacao \leftarrow probMutacao - (probMutacao \times 0.03)$;
  retorna $filho$;
fim

4.3.2.2 Mutação em 2 genes

Na estratégia de mutação de dois genes, para cada cromossomo selecionado, o algoritmo escolhe aleatoriamente dois índices distintos e inverte os valores dos genes correspondentes. Essa abordagem gera uma variação mais significativa na solução, promovendo maior diversidade e auxiliando na superação de possíveis ótimos locais.

Algoritmo 8: Mutação em 2 genes

Entrada: pai; $tamCromossomo$

Saída: $filho$

início

$gene_1, gene_2 \leftarrow$ random.sample(range($tamCromossomo$), 2);

$filho \leftarrow pai$;

se $pai[gene_1] = 0$ **então**

$filho[gene_1] \leftarrow 1$;

senão se $pai[gene_1] = 1$ **então**

$filho[gene_1] \leftarrow 0$;

fim

se $pai[gene_2] = 0$ **então**

$filho[gene_2] \leftarrow 1$;

senão se $pai[gene_2] = 1$ **então**

$filho[gene_2] \leftarrow 0$;

fim

$probCruzamento \leftarrow probCruzamento + (probCruzamento \times 0.05)$;

$probMutacao \leftarrow probMutacao - (probMutacao \times 0.03)$;

retorna $filho$;

fim

4.3.2.3 Mutação em vários gene

Na técnica de mutação para vários genes, para cada cromossomo que passa pelo operador, é selecionado um número aleatório de genes. Em seguida, cada gene escolhido sofre uma inversão de seu valor – se o gene é 0, ele é alterado para 1, e vice-versa. Essa mutação pontual permite a introdução de variações no cromossomo, contribuindo para a diversidade genética da população e possibilitando a exploração de novas regiões do espaço de soluções.

Algoritmo 9: Mutação em vários genes

```
Entrada: pai; tamCromossomo
Saída: filho
início
    qtdGenes ← random.randint(1, tamCromossomo);
    genesAleat ← random.sample(range(tamCromossomo), qtdGenes);
    para cada gene ∈ genesAleat faça
        se pai[gene] = 0 então
            filho[gene] ← 1;
        senão
            filho[gene] ← 0;
        fim
    fim
    probCruzamento ← probCruzamento + (probCruzamento × 0.05);
    probMutacao ← probMutacao − (probMutacao × 0.03);
    retorna filho ;
fim
```

4.3.3 Integração da Heurística de Busca Local ao AG

A escolha por integrar uma heurística de *Busca local* ao *Algoritmo genético* fundamenta-se na evidência de que métodos híbridos podem superar a convergência prematura, intensificando a exploração de regiões promissoras do espaço de soluções. Estudos recentes, como os apresentados por Benlic e Hao (2013) e Bansal *et al.* (2012) demonstram que a combinação de métodos evolutivos com técnicas de intensificação, pode melhorar significativamente o desempenho em problemas NP-difíceis.

Como vimos na subseção 3.3.1, a heurística de *Busca Local* vai contribuir positiva-

mente para o *Algoritmo Genético* na exploração em pontos estratégicos no espaço de soluções, intensificando a qualidade das soluções, o que vai contribuir diretamente para escapar de ótimos locais.

Esses dois métodos unidos geram um equilíbrio essencial, pois o *Algoritmo Genético*, com suas operações de *Crossover* e *Mutação*, promove a exploração de novas regiões do espaço de busca, enquanto a *Busca Local*, vai ajudar a refinar as soluções promissoras.

A *Busca Local* no *Algoritmo Genético* vai ser ativada no início de cada geração, caso as transformações não gerem cromossomos com corte maior em um determinado período, ou seja, após um período sem convergências. A busca vai ocorrer em posições específicas da população, sendo elas: na posição inicial, na segunda posição, na posição intermediária e na posição final. Esse comportamento pode ser visto no pseudo-algoritmo abaixo, que representa como ocorre a integração da heurística de *Busca Local* junto a metaheurística do *Algoritmo Genético*.

Algoritmo 10: Pseudo-Algoritmo Genético com integração da Busca Local

Entrada: Um Grafo não direcionado e ponderado; Período máximo sem convergência

Saída: Solução

```
início
    Gerar uma população inicial ;
    Avaliar o fitness dos indíviduos da população inicial;
    semConvergencia ← 0;
    enquanto Não atingir o critério de parada fazer
        se semConvergencia ≥ periodoMaximo então
            para pos ∈ [0, 1, t/2, t] faça
                BuscaLocal(pos)
            fim
        fim
        Seleção de pais da população;
        Aplicar crossover ou mutação nos pais para estimular a reprodução;
        Avaliar o fitness dos filhos gerados;
        Ordenar todos os indivíduos da população;
        Selecionar os melhores para permanecer para a próxima geração;
        se Corte do melhor cromossomo da geração atual é maior que o melhor
         cromossomo da geração anterior então
            semConvergencia ← 0;
        senão
            semConvergencia ← semConvergencia + 1;
        fim
    fim
fim
```

4.4 Filtragem de soluções por restrições

Com o objetivo de identificar soluções promissoras, foi implementado o conceito de **ilha**, um processo que verifica se os cromossomos gerados atendem às restrições definidas pelo problema. Cada cromossomo é analisado individualmente, avaliando se a configuração de seus genes satisfaz as condições estabelecidas para o corte máximo do grafo.

Os cromossomos que atendem aos critérios são armazenados em uma subpopulação separada, ou ilha, sem interferir diretamente no refinamento ou na evolução da população principal. Essa separação permite identificar quais soluções foram aceitas pelas restrições, auxiliando na análise da qualidade dos cromossomos sem impactar a busca local ou outras

transformações genéticas. A seguir, será apresentado o pseudo-algoritmo que descreve esse processo:

Algoritmo 11: Pseudo-Algoritmo de Ilha

```
Entrada: População; Valores de corte; Estrutura do grafo
Saída: Subpopulação com cromossomos que atendem às restrições
início
    para cada cromossomo na população faça
        Inicializar contador de restrições atendidas;
        para cada gene no cromossomo faça
            Calcular o valor de crote do gene;
            se atender às restrições então
                Incrementar contador;
            senão
                interromper verificação;
            fim
        fim
        se todos os genes atenderem às restrições então
            Verificar se o cromossomo já está na ilha;
            se não estiver e houver espaço disponível então
                Inserir cromossomo na ilha;
    fim
    retorna ilha com cromossomos aceitos
fim
```

As restrições utilizadas na verificação garantem que a separação dos vértices em subconjuntos preserve a coerência da solução e evite configurações inviáveis. Cada vértice do grafo possui um **valor de corte total** (x^{total}), determinado pela soma dos pesos das arestas incidentes. Para que um cromossomo seja aceito na ilha, cada vértice classificado como pertencente ao subconjunto S deve ter um **valor de corte na solução atual** (x^{atual}) menor ou igual à metade do seu valor total, enquanto os vértices no subconjunto S' devem possuir um valor de corte maior ou igual a metade do valor total, como descritas na Equação 4.3. Essas restrições garantem que a separação dos vértices maximize o corte sem violar as condições estruturais do problema.

$$\forall u \in V, \quad \begin{cases} u \in S \implies x_u^{\text{atual}} \leq x_u^{\text{total}}, \\ u \in S' \implies x_u^{\text{atual}} \geq x_u^{\text{total}}. \end{cases} \tag{4.3}$$

4.5 Avaliação final e retorno dos resultados

Ao final de todas as gerações, o algoritmo vai calcular a diferença entre o maior valor de corte que foi obtido na população inicial e o maior valor de corte alcançado após o processo evolutivo. Essa diferença, juntamente com o cromossomo que apresentou o melhor valor de corte no final das gerações, será então retornada como resultado do algoritmo.

Além de registrar a diferença final encontrada para os valores de corte, o algoritmo registra informações como qual operação de transformação que registrou domínio ao longo das gerações. E também, quais cromossomos passaram pelos critérios de aceitação e foram adicionados na *ilha*.

5 EXPERIMENTOS COMPUTACIONAIS

Para avaliar o desempenho do algoritmo proposto, foram realizados experimentos computacionais em **9 instâncias distintas**, variando a quantidade de vértices do grafo e explorando diferentes configurações de parâmetros. As instâncias testadas foram organizadas em três grupos, considerando os grafos com **60**, **80** e **100** vértices, de modo a verificar a escalabilidade e eficiência da abordagem em diferentes cenários.

Os experimentos computacionais foram conduzidos aplicando diferentes combinações de parâmetros do *Algoritmo Genético*, considerando os seguintes grupos:

1. Uso de ilha:
    - **SI**: Sem ilha;
    - **CI**: Com ilha.
2. Método de Mutação:
    - **M1**: Mutação em 1 gene dos cromossomos escolhidos;
    - **M2**: Mutação em 2 genes dos cromossomos escolhidos;
    - **MA**: Mutação em vários genes escolhidos.
3. Método de Crossover:
    - **P**: *Crossover* que faz a troca percentual, variando o tamanho escolhido, dos cromossomos;
    - **T2**: *Crossover* que faz a troca de 2 genes entre os cromossomos;
    - **PI**: *Crossover* que prevalece a igualdade, ou seja, dos dois cromossomos escolhidos na vez, para o novo cromossomo será mantido o gene quando na mesma posição nos pais tiverem ambos valores iguais, e quando não, será sorteado entre os dois valores para o gene.
4. Período sem convergência para ativação da Busca Local:
    - **BL5**: Busca local a partir de um período de 5 gerações sem convergências;
    - **BL10**: Busca local a partir de um período de 10 gerações sem convergências;
    - **BL20**: Busca local a partir de um período de 20 gerações sem convergências.

Essa configuração experimental permite uma análise detalhada do impacto de cada componente na qualidade das soluções e no tempo de execução do algoritmo. No próximo tópico, serão apresentados os resultados obtidos para cada combinação de parâmetros, discutindo o desempenho do modelo em diferentes cenários.

6 RESULTADOS

Ao todo, as configurações possibilitaram a obtenção de 54 combinações diferentes para a execução dos testes. Para cada instância, o fluxo do teste consiste em montar cada combinação de parâmetros por vez. Ao aplicar todas essas combinações nas 9 instâncias, foram obtidos os resultados para as instâncias com $n = 60$, $n = 80$ e $n = 100$, apresentados na Tabela 1, Tabela 2 e Tabela 3, respectivamente.

Com o total de 486 testes realizados, notou-se que os resultados obtidos se aproximam ou encontram os valores ótimos disponíveis no artigo do BiqMac (Wiegele, 2007). Sendo a combinação "$SIM1PIBL20$"como a prevalecente para os melhores resultados obtidos, com $77,78\%$ dos testes para essa combinação de parâmetros retornando o cromossomo com valor ótimo ou próximo ao valor ótimo, e as combinações "$SIMAT2BL5$"e "$CIMAT2BL5$"como prevalecentes para retornar os resultados mais distantes das melhores soluções com $22,22\%$ dos testes para ambas as combinações de parâmetros retornando cromossomos com valores mais distantes do valor ótimo.

Seja CP o conjunto com todas as combinações de parâmetros, teremos para cada $k \in CP$ a métrica do percentual de êxito total de cada combinação, dada na seguinte equação:

$$P(k) = \frac{\sum_{i \in I} v_i(k)}{\sum_{i \in I} b_i} \times 100\% \tag{6.1}$$

onde, I é o conjunto de instâncias testadas; $v_i(k)$ representa o valor de corte obtido para a instância i utilizando a combinação k de parâmetros; e b_i é o valor ótimo (teórico) para a instância i, conforme definido no BiqMac.

Portanto, com base no cálculo do percentual, podemos consolidar "$SIM1PIBL20$"como a combinação com melhores resultados obtidos, onde obteve percentual de $99,88\%$. Já para as combinações "$SIMAT2BL5$"e "$CIMAT2BL5$", que obtiveram resultados mais distantes, sustentou-se com percentuais de $98,65\%$ e $98,57\%$, respectivamente. Na Tabela 4, podemos ver os percentuais para cada combinação.

A análise dos resultados evidenciou padrões distintos na influência dos parâmetros (Tabela 5) sobre a aproximação dos valores de corte obtidos aos teóricos. Para cada instância, foi calculado o desempenho relativo de cada parâmetro, obtendo-se uma métrica que se aproxima de 1. Essa métrica foi definida como a razão entre a soma dos valores de cortes obtidos nas

combinações em que o parâmetro aparece e o produto do número de aparições do parâmetro pelo valor ótimo teórico.

Observa-se na Tabela 5 que parâmetros como a Mutação em 1 gene ($M1$) e o operador de Crossover que prevalece igualdade (PI) alcançaram valores muito próximos do máximo que pode ser obtido, demonstrando uma influência positiva e consistente na qualidade das soluções. Em contrapartida, estratégias mais "agressivas", como a Mutação em vários genes (MA), apresentaram valores um pouco inferiores, sugerindo que alterações menos intensas podem ser mais adequadas para a manutenção da qualidade das soluções. A análise das combinações reforça a ideia que a junção de configurações equilibradas, especialmente aquelas envolvendo uma busca local ativada após 20 gerações sem convergência ($BL20$), tende a maximizar a aproximação ao valor ótimo.

Portanto, com a análise da influência individual desses parâmetros para os resultados dos testes, pode-se comprovar individualmente os parâmetros **SI**, **M1**, **PI** e **BL20** como melhores influentes para os resultados obtidos. Acerca dessa análise individual dos parâmetros, foi possível destacar para cada grupo de instâncias as seguintes configurações:

- Para os testes com instâncias de $n = 60$ (Tabela 1), os resultados retornaram com melhores valores para soluções quando foram influenciados pelos seguintes parâmetros: **SI**; **M1**; **PI**; **BL20**.
- Para os testes com instâncias de $n = 80$ (Tabela 2), os resultados retornaram com melhores valores para soluções quando foram influenciados pelos seguintes parâmetros: **SI**; **M1**; **PI**, **P** ou **T2** ; **BL20**.
- Para os testes com instâncias de $n = 100$ (Tabela 3), os resultados retornaram com melhores valores para soluções quando foram influenciados pelos seguintes parâmetros: **SI**; **M1**; **P**; **BL10**.

Em relação ao tempo computacional, foi possível obter uma média de execução de teste para cada grupo de instâncias, como destacado a seguir:

- Para instâncias de $n = 60$, a média de tempo de execução foi de 7 minutos e 14 segundos;
- Para instâncias de $n = 80$, a média de tempo de execução foi de 11 minutos e 2 segundos.
- Para instâncias de $n = 100$, a média de tempo de execução foi de 27 minutos e 23 segundos.

Esses padrões indicam que tanto o efeito isolado quanto a interação entre os parâmetros são determinantes para a eficiência do algoritmo, e que o ajuste fino desses elementos é crucial para otimizar os resultados do problema do corte máximo.

Tabela 1 – Testes em instâncias com $n = 60$ vértices

Instância	$g05\_60.0$	$g05\_60.1$	$g05\_60.2$
Corte Máximo Teórico Combinação	536	532	529
CIM1PIBL5	534	531	527
CIM1PIBL10	**536**	531	**529**
CIM1PIBL20	**536**	**532**	528
CIM1PBL5	535	528	523
CIM1PBL10	**536**	531	**529**
CIM1PBL20	535	**532**	528
CIM1T2BL5	**536**	**532**	528
CIM1T2BL10	**536**	**532**	528
CIM1T2BL20	**536**	**532**	**529**
CIM2PIBL5	535	526	524
CIM2PIBL10	535	**532**	528
CIM2PIBL20	**536**	**532**	528
CIM2PBL5	**536**	531	523
CIM2PBL10	**536**	**532**	528
CIM2PBL20	**536**	**532**	**529**
CIM2T2BL5	529	531	523
CIM2T2BL10	**536**	**532**	**529**
CIM2T2BL20	**536**	**532**	**529**
CIMAPIBL5	**536**	531	523
CIMAPIBL10	**536**	531	528
CIMAPIBL20	**536**	**532**	527
CIMAPBL5	**536**	528	524
CIMAPBL10	534	**532**	**529**
CIMAPBL20	**536**	**532**	**529**
CIMAT2BL5	525	530	**529**
CIMAT2BL10	531	526	528
CIMAT2BL20	535	**532**	528
SIM1PIBL5	535	**532**	**529**
SIM1PIBL10	**536**	**532**	**529**
SIM1PIBL20	**536**	**532**	**529**
SIM1PBL5	535	**532**	526
SIM1PBL10	**536**	528	528
SIM1PBL20	**536**	**532**	528
SIM1T2BL5	**536**	531	**529**
SIM1T2BL10	535	**532**	528
SIM1T2BL20	**536**	**532**	528
SIM2PIBL5	**536**	525	521
SIM2PIBL10	**536**	**532**	528
SIM2PIBL20	**536**	528	**529**
SIM2PBL5	529	531	523
SIM2PBL10	534	**532**	**529**
SIM2PBL20	**536**	**532**	528
SIM2T2BL5	**536**	531	528
SIM2T2BL10	**536**	530	528
SIM2T2BL20	**536**	**532**	**529**
SIMAPIBL5	**536**	**532**	527
SIMAPIBL10	**536**	**532**	525
SIMAPIBL20	**536**	**532**	527
SIMAPBL5	534	**532**	526
SIMAPBL10	**536**	525	**529**
SIMAPBL20	535	531	527
SIMAT2BL5	533	520	523
SIMAT2BL10	533	531	528
SIMAT2BL20	535	531	527

Fonte: Elaborado pelo autor

Tabela 2 – Testes em instâncias com $n = 80$ vértices

Instância	$g05\_80.0$	$g05\_80.1$	$g05\_80.2$
Corte Máximo Teórico	929	941	934
Combinação			
CIM1PIBL5	926	935	930
CIM1PIBL10	925	**941**	931
CIM1PIBL20	921	**941**	933
CIM1PBL5	915	939	928
CIM1PBL10	**929**	**941**	**934**
CIM1PBL20	926	**941**	**934**
CIM1T2BL5	927	**941**	928
CIM1T2BL10	927	**941**	927
CIM1T2BL20	918	**941**	933
CIM2PIBL5	919	930	933
CIM2PIBL10	927	**941**	925
CIM2PIBL20	**929**	**941**	**934**
CIM2PBL5	915	**941**	931
CIM2PBL10	922	**941**	919
CIM2PBL20	927	**941**	928
CIM2T2BL5	922	914	921
CIM2T2BL10	926	**941**	921
CIM2T2BL20	925	**941**	**934**
CIMAPIBL5	904	939	914
CIMAPIBL10	913	**941**	919
CIMAPIBL20	917	**941**	922
CIMAPBL5	902	939	918
CIMAPBL10	919	937	916
CIMAPBL20	922	**941**	917
CIMAT2BL5	918	938	918
CIMAT2BL10	916	**941**	926
CIMAT2BL20	926	940	926
SIM1PIBL5	918	**941**	**934**
SIM1PIBL10	927	**941**	**934**
SIM1PIBL20	922	**941**	**934**
SIM1PBL5	918	932	924
SIM1PBL10	**929**	**941**	**934**
SIM1PBL20	920	**941**	931
SIM1T2BL5	926	933	931
SIM1T2BL10	926	**941**	923
SIM1T2BL20	921	**941**	**934**
SIM2PIBL5	923	917	923
SIM2PIBL10	**929**	**941**	**934**
SIM2PIBL20	**929**	**941**	**934**
SIM2PBL5	926	**941**	931
SIM2PBL10	925	**941**	917
SIM2PBL20	**929**	**941**	931
SIM2T2BL5	**929**	932	925
SIM2T2BL10	926	**941**	926
SIM2T2BL20	**929**	**941**	931
SIMAPIBL5	918	932	920
SIMAPIBL10	919	**941**	925
SIMAPIBL20	920	**941**	930
SIMAPBL5	906	910	924
SIMAPBL10	926	**941**	927
SIMAPBL20	914	**941**	933
SIMAT2BL5	923	926	912
SIMAT2BL10	926	935	928
SIMAT2BL20	920	935	931

Fonte: Elaborado pelo autor

Tabela 3 – Testes em instâncias com $n = 100$ vértices

Instância	*g*05_100.0	*g*05_100.1	*g*05_100.2
Corte Máximo Teórico	1430	1425	1432
Combinação			
CIM1PIBL5	1416	1401	1405
CIM1PIBL10	**1430**	1408	1416
CIM1PIBL20	1421	1424	1425
CIM1PBL5	1420	1416	1420
CIM1PBL10	1421	1424	1423
CIM1PBL20	1421	1418	1430
CIM1T2BL5	1426	1418	1409
CIM1T2BL10	1426	1418	**1432**
CIM1T2BL20	1424	1423	**1432**
CIM2PIBL5	1411	1415	1407
CIM2PIBL10	1428	1420	1424
CIM2PIBL20	1423	1417	1423
CIM2PBL5	1418	1415	1426
CIM2PBL10	1418	1418	1411
CIM2PBL20	1428	1421	1425
CIM2T2BL5	1410	1421	1413
CIM2T2BL10	1424	1424	**1432**
CIM2T2BL20	1425	1421	1421
CIMAPIBL5	1412	1423	1404
CIMAPIBL10	1410	1416	1405
CIMAPIBL20	1403	1400	1414
CIMAPBL5	1415	1410	1405
CIMAPBL10	1411	1417	1417
CIMAPBL20	1402	1412	1422
CIMAT2BL5	1411	1398	1397
CIMAT2BL10	1411	1401	1408
CIMAT2BL20	1413	1406	1413
SIM1PIBL5	1427	1416	1421
SIM1PIBL10	1422	1417	**1432**
SIM1PIBL20	**1430**	1422	**1432**
SIM1PBL5	1417	1418	**1432**
SIM1PBL10	1420	1422	1423
SIM1PBL20	1419	1414	1429
SIM1T2BL5	1425	1412	1418
SIM1T2BL10	1422	1420	1426
SIM1T2BL20	1419	1420	1430
SIM2PIBL5	1425	1412	1430
SIM2PIBL10	1411	1413	1430
SIM2PIBL20	1407	1419	1418
SIM2PBL5	1410	1423	1409
SIM2PBL10	1418	1422	1410
SIM2PBL20	1424	1419	1429
SIM2T2BL5	1410	1405	1430
SIM2T2BL10	1421	1424	**1432**
SIM2T2BL20	1420	1423	1428
SIMAPIBL5	1411	1416	1395
SIMAPIBL10	1422	1413	1419
SIMAPIBL20	1413	1413	1413
SIMAPBL5	1410	1413	1421
SIMAPBL10	1419	1404	1427
SIMAPBL20	1415	1400	1427
SIMAT2BL5	1410	1414	1410
SIMAT2BL10	1409	1419	1418
SIMAT2BL20	1413	1406	1415

Fonte: Elaborado pelo autor

Tabela 4 – Percentual de êxito das combinações

Combinação	Percentual
CIM1PIBL5	99,04%
CIM1PIBL10	99,53%
CIM1PIBL20	99,69%
CIM1PBL5	99,26%
CIM1PBL10	99,77%
CIM1PBL20	99,74%
CIM1T2BL5	99,51%
CIM1T2BL10	99,76%
CIM1T2BL20	99,77%
CIM2PIBL5	98,99%
CIM2PIBL10	99,68%
CIM2PIBL20	99,71%
CIM2PBL5	99,40%
CIM2PBL10	99,27%
CIM2PBL20	99,76%
CIM2T2BL5	98,80%
CIM2T2BL10	99,74%
CIM2T2BL20	99,72%
CIMAPIBL5	98,83%
CIMAPIBL10	98,98%
CIMAPIBL20	98,90%
CIMAPBL5	98,72%
CIMAPBL10	99,13%
CIMAPBL20	99,14%
CIMAT2BL5	98,57%
CIMAT2BL10	98,85%
CIMAT2BL20	99,21%
SIM1PIBL5	99,60%
SIM1PIBL10	99,79%
SIM1PIBL20	**99,88**%
SIM1PBL5	99,38%
SIM1PBL10	99,69%
SIM1PBL20	99,56%
SIM1T2BL5	99,46%
SIM1T2BL10	99,60%
SIM1T2BL20	99,69%
SIM2PIBL5	99,13%
SIM2PIBL10	99,61%
SIM2PIBL20	99,46%
SIM2PBL5	99,25%
SIM2PBL10	99,31%
SIM2PBL20	99,78%
SIM2T2BL5	99,29%
SIM2T2BL10	99,72%
SIM2T2BL20	99,78%
SIMAPIBL5	98,84%
SIMAPIBL10	99,36%
SIMAPIBL20	99,27%
SIMAPBL5	98,71%
SIMAPBL10	99,38%
SIMAPBL20	99,25%
SIMAT2BL5	98,65%
SIMAT2BL10	99,30%
SIMAT2BL20	99,14%

Fonte: Elaborado pelo autor

Tabela 5 – Influência dos parâmetros

Ilha

Parâmetro	SI	CI
*g*05_60_0	**0,99848**	0,99779
*g*05_60_1	0,99708	**0,99798**
*g*05_60_2	**0,99671**	0,99664
*g*05_80_0	**0,99366**	0,99123
*g*05_80_1	0,99539	**0,99772**
*g*05_80_2	**0,99377**	0,99136
*g*05_100_0	0,99117	**0,9914**
*g*05_100_1	**0,99335**	0,99298
*g*05_100_2	**0,99328**	0,98953

Fonte: Elaborado pelo autor

Mutação

Parâmetro	M1	M2	MA
g05_60_0	**0,99927**	0,99813	0,99699
g05_60_1	**0,99875**	0,9976	0,99624
g05_60_2	**0,998**	0,99601	0,99601
g05_80_0	0,99396	**0,99611**	0,98726
g05_80_1	**0,99852**	0,99581	0,99534
g05_80_2	**0,99673**	0,99322	0,98775
g05_100_0	**0,99479**	0,99188	0,98718
g05_100_1	0,99458	**0,9954**	0,98951
g05_100_2	**0,99453**	0,99309	0,98658

Fonte: Elaborado pelo autor

Crossover

Parâmetro	PI	P	T2
g05_60_0	**0,99948**	0,99824	0,99668
g05_60_1	**0,99781**	0,9976	0,99718
g05_60_2	0,99622	0,99622	**0,99758**
g05_80_0	0,99187	0,99091	**0,99456**
g05_80_1	0,99693	**0,99717**	0,99557
g05_80_2	**0,99387**	0,99197	0,99185
g05_100_0	**0,99153**	0,99091	0,99141
g05_100_1	0,99279	**0,99361**	0,9931
g05_100_2	0,9898	**0,99263**	0,99178

Fonte: Elaborado pelo autor

Busca Local

Parâmetro	BL5	BL10	BL20
g05_60_0	0,99627	0,99855	**0,99959**
g05_60_1	0,99561	0,9976	**0,99937**
g05_60_2	0,99307	**0,99853**	0,99842
g05_80_0	0,98882	**0,99492**	0,9936
g05_80_1	0,99067	0,99941	**0,99959**
g05_80_2	0,99007	0,99132	**0,99631**
g05_100_0	0,99005	**0,99235**	0,99145
g05_100_1	0,99205	**0,99415**	0,99329
g05_100_2	0,98743	0,99259	**0,99418**

Fonte: Elaborado pelo autor

7 CONCLUSÕES

Este trabalho apresentou uma abordagem inovadora para a configuração de parâmetros em algoritmos genéticos aplicados ao problema do Corte Máximo de um grafo. Os experimentos realizados demonstraram que diversas combinações de parâmetros geram soluções com valores de corte próximos aos ótimos teóricos e até mesmo o próprio ótimo teórico, evidenciando a eficácia da estratégia proposta.

Os experimentos demonstraram que a escolha dos parâmetros influencia diretamente a qualidade das soluções obtidas pelo algoritmo genético. Em particular, **Mutação em 1 gene** ($M1$) e **Busca Local** com $BL20$ mostraram um impacto positivo consistente, favorecendo um equilíbrio entre exploração e intensificação. O operador de **crossover que preserva genes iguais** (PI) supera as outras técnicas de recombinação, sugerindo que a manutenção de boas características estruturais nos cromossomos melhora o desempenho. Em relação ao mecanismo de ilha, embora não interfira diretamente na seleção ou no refinamento da população principal, permite identificar e armazenar soluções promissoras. Essa estratégia facilita a análise da qualidade das soluções isoladas, mas, ao mesmo tempo, acarreta um custo adicional em termos de tempo computacional, devido à necessidade de verificação e armazenamento dessas soluções. Dessa forma, a aplicação do conceito de ilha deve ser balanceada, considerando que, embora contribua para a identificação de bons candidatos, pode impactar a eficiência global do algoritmo.

Em particular, a configuração "$SIM1PIBL20$" destacou-se como a mais eficiente, proporcionando resultados que minimizam os custos computacionais e maximizam a qualidade das soluções encontradas. Esses achados reforçam a importância de um ajuste fino dos parâmetros no contexto de metaheurísticas, contribuindo para a otimização de problemas complexos.

Como perspectivas para trabalhos futuros, sugere-se a incorporação de novos parâmetros e operadores genéticos que possam ampliar ainda mais a diversidade da população, bem como a investigação de métodos alternativos para a geração da população inicial, com o intuito de explorar de forma mais abrangente o espaço de soluções. Além disso, investigações adicionais podem avaliar se o impacto da filtragem por ilhas pode ser otimizado para preservar diversidade sem comprometer a qualidade das soluções. Dessa forma, o presente estudo abre caminhos para aprimoramentos que poderão tornar a abordagem ainda mais robusta e aplicável em cenários práticos de otimização.

REFERÊNCIAS

BANSAL, R.; SRIVASTAVA, K.; SRIVASTAVA, S. A hybrid evolutionary algorithm for the cutwidth minimization problem. In: IEEE. **2012 IEEE Congress on Evolutionary Computation**. [*S. l.*], 2012. p. 1–8.

BENLIC, U.; HAO, J.-K. Breakout local search for the max-cutproblem. **Engineering Applications of Artificial Intelligence**, Elsevier, v. 26, n. 3, p. 1162–1173, 2013.

BENVENGA, M. A. C. **APLICAÇÃO DE ALGORITMO METAHEURÍSTICO HÍBRIDO COM MECANISMO PARA ACELERAÇÃO DE CONVERGÊNCIA NA OTIMIZAÇÃO DE PROCESSOS DO AGRONEGÓCIO**. Tese (Doutorado) – Universidade Paulista, 2022.

BOROS, E.; HAMMER, P. L. The max-cut problem and quadratic 0–1 optimization; polyhedral aspects, relaxations and bounds. **Annals of Operations Research**, Springer, v. 33, n. 3, p. 151–180, 1991.

COSTA, J. M. de S. **Algoritmos espectrais de agrupamento em redes sociais**. Dissertação (Mestrado) – Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, Rio de Janeiro, 2014. Dissertação (Mestrado).

EIBEN, A. E.; SMITH, J. E. **Introduction to evolutionary computing**. [*S. l.*]: Springer, 2015.

FILHO, J. A. F. d. C. Heurística probabilística guiada pelos potenciais dos vértices para o problema do corte máximo. Simpósio brasileiro de pesquisa operacional, 2021.

GLOVER, F. Tabu search—part i. **ORSA Journal on computing**, Informs, v. 1, n. 3, p. 190–206, 1989.

GOLDBARG, M. C.; LUNA, H. P. L. **Otimização combinatória e programação linear: modelos e algoritmos**. [*S. l.*]: Elsevier, 2005.

GOSTI, W.; NGUYEN, G.; WAN, M.; ZHOU, M. Approximation algorithms for the max-cut problem. **URL: http://citeseer. ist. psu. edu/489604. html**, Citeseer, 1995.

HOOS, H. H.; STUTZLE, T. Stochastic local search. In: **Handbook of Approximation Algorithms and Metaheuristics**. [*S. l.*]: Chapman and Hall/CRC, 2018. p. 297–307.

IKEDA, P. A. Introdução aos algoritmos genéticos. **Rio de Janeiro**, 2009.

JURKIEWICZ, S. Grafos–uma introdução. **São Paulo: OBMEP**, 2009.

KARP, R. M. **Reducibility among combinatorial problems**. [*S. l.*]: Springer, 1972.

Khan Academy. **Descrevendo Grafos**. 2022. https://pt.khanacademy.org/computing/computer-science/algorithms/graph-representation/a/describing-graphs. Acessado em 06 set. 2022.

KIM, J.; YOON, Y.; MOON, B.-R. Solving maximum cut problem with an incremental genetic algorithm. In: **Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion**. [*S. l.*: *s. n.*], 2016. p. 49–50.

LIMA, G. L. Algoritmos para resolução do problema do corte máximo: abordagem exata e meta-heurísticas. 2022.

LIMA, G. L.; MELO, I. E. S. de; LINS, S. L. S. Problema do corte máximo simples: Implementação de uma nova abordagem exata com árvores de busca e da meta-heurística grasp-vnd. 2021.

LUCAS, D. C. Algoritmos genéticos: uma introdução. **Universidade Federal do Rio Grande do Sul**, v. 24, 2002.

MOSCATO, P.; COTTA, C.; MENDES, A. *et al.* Memetic algorithms. **New optimization techniques in engineering**, Springer Berlin, Heidelberg, v. 141, p. 53–85, 2004.

OLIVEIRA, F. D. G. de; SANTOS, M. C.; SOARES, P. L. B. Parametrização de um algoritmo genético para o problema do corte máximo. **Anais do Encontro de Computação do Oeste Potiguar ECOP/UFERSA (ISSN 2526-7574)**, v. 1, n. 6, p. 42–45, 2022.

SIMONE, C. D. **The max cut problem**. [*S. l.*]: Rutgers The State University of New Jersey, School of Graduate Studies, 1992.

SOARES, P. L. B. Problemas quadráticos binários: abordagem teórica e computacional. Fortaleza, 2018. 124 f. – Tese (Doutorado) – Curso de Ciência da Computação, Departamento de Computação.

SOUSA, S. de; HAXHIMUSA, Y.; KROPATSCH, W. G. Estimation of distribution algorithm for the max-cut problem. In: SPRINGER. **Graph-Based Representations in Pattern Recognition: 9th IAPR-TC-15 International Workshop, GbRPR 2013, Vienna, Austria, May 15-17, 2013. Proceedings 9**. [*S. l.*], 2013. p. 244–253.

TALBI, E.-G. **Metaheuristics: from design to implementation**. [*S. l.*]: John Wiley & Sons, 2009.

TSURUTA, J. H.; NARCISO, M. G. **Um estudo sobre algoritmos genéticos**. [*S. l.*]: Embrapa Informática Agropecuária, 2000. Cap. 8, p. 13.

WIEGELE, A. Biq mac library—a collection of max-cut and quadratic 0-1 programming instances of medium size. **Preprint**, v. 51, p. 112–127, 2007.

ZHOU, Y.; LAI, X.; LI, K. Approximation and parameterized runtime analysis of evolutionary algorithms for the maximum cut problem. **IEEE transactions on cybernetics**, IEEE, v. 45, n. 8, p. 1491–1498, 2014.

ANEXO A – ARTIGO APRESENTADO NA V ECOP

Esse trabalho de conclusão de curso é fruto de uma pesquisa de iniciação científica na área de grafos e otimização de algoritmos, onde o artigo **Parametrização de um Algoritmo Genético para o Problema do Corte Máximo** (Oliveira *et al.*, 2022) foi apresentado no V Encontro de Computação do Oeste Potiguar.

Disponível em: https://periodicos.ufersa.edu.br/ecop/article/view/11834