



UFOP

Universidade Federal
de Ouro Preto

**Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas**

Heurística Matemática Aplicada ao Problema da Coloração de Grafos

Mariane Regina Afonso Vieira

TRABALHO DE CONCLUSÃO DE CURSO

**ORIENTAÇÃO:
George Henrique Godim da Fonseca**

**Julho, 2018
João Monlevade—MG**

Mariane Regina Afonso Vieira

Heurística Matemática Aplicada ao Problema da Coloração de Grafos

Orientador: George Henrique Godim da Fonseca

Monografia apresentada ao curso de Engenharia de Computação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina “Trabalho de Conclusão de Curso II”.

Universidade Federal de Ouro Preto

João Monlevade

Julho de 2018

A Ficha Catalográfica é elaborada exclusivamente pela Biblioteca. Substitua esta página pelo documento gerado na versão final da sua monografia.

FOLHA DE APROVAÇÃO DA BANCA EXAMINADORA

Heurística Matemática Aplicada ao Problema da Coloração de Grafos

Mariane Regina Afonso Vieira

Monografia apresentada ao Instituto de Ciências Exatas e Aplicadas da Universidade Federal de Ouro Preto como requisito parcial da disciplina CSI499 – Trabalho de Conclusão de Curso II do curso de Bacharelado em Engenharia de Computação e aprovada pela Banca Examinadora abaixo assinada:

George Henrique Godim da Fonseca
Doutor em Engenharia Elétrica
DECSI – UFOP

Tatiana Alves Costa
Doutora em Engenharia Elétrica
Examinador
DECSI – UFOP

Fernando Bernardes de Oliveira
Doutor em Engenharia Elétrica
Examinador
DECSI – UFOP

João Monlevade, 17 de julho de 2018

ATA DE DEFESA

No dia 17 do mês de Julho de 2018, às 14:00 horas, na sala do Instituto de Ciências Exatas e Aplicadas, foi realizada a defesa de Monografia pelo(a) aluno(a) **Mariane Regina Afonso Vieira**, sendo a Comissão Examinadora constituída pelos professores: George Henrique Godim da Fonseca, Tatiana Alves Costa, Fernando Bernardes de Oliveira. O(a) candidato(a) apresentou a monografia intitulada: “**Heurística Matemática Aplicada ao Problema da Coloração de Grafos**”. A comissão examinadora deliberou, por unanimidade, pela aprovação do candidato, com nota _____ (_____), concedendo-lhe o prazo de 15 dias para incorporação das alterações sugeridas ao texto final. Na forma regulamentar, foi lavrada a presente ata que é assinada pelos membros da Comissão Examinadora e pelo(a) graduando(a).

George Henrique Godim da Fonseca
Doutor em Engenharia Elétrica
DECSI – UFOP

Tatiana Alves Costa
Doutora em Engenharia Elétrica
Examinador(a)
DECSI – UFOP

Fernando Bernardes de Oliveira
Doutor em Engenharia Elétrica
Examinador(a)
DECSI – UFOP

Mariane Regina Afonso Vieira

TERMO DE RESPONSABILIDADE

Eu, **Mariane Regina Afonso Vieira** declaro que o texto do trabalho de conclusão de curso intitulado “*Heurística Matemática Aplicada ao Problema da Coloração de Grafos*” é de minha inteira responsabilidade e que não há utilização de texto, material fotográfico, código fonte de programa ou qualquer outro material pertencente a terceiros sem as devidas referências ou consentimento dos respectivos autores.

João Monlevade, 17 de julho de 2018

Mariane Regina Afonso Vieira

Dedico esse trabalho primeiramente a Deus, por ser essencial em minha vida, e à minha mãe Maria Dolores, que não mediu esforços para que eu chegasse até esta etapa.

Agradecimentos

Agradeço a todos que de alguma forma contribuíram para a elaboração deste trabalho: aos professores do curso de graduação em Engenharia de Computação da Universidade Federal de Ouro Preto, em especial ao meu orientador Prof. Dr. George Henrique Godim da Fonseca pela orientação, apoio, confiança e empenho dedicado à elaboração deste trabalho. Aos meus amigos, que fizeram parte de minha formação. Ao colegiado de Engenharia de Computação, departamento de Computação e Sistemas, secretaria e direção do Instituto de Ciências Exatas e Aplicadas (ICEA). À Universidade Federal de Ouro Preto pelo auxílio fornecido por meio do Programa de Iniciação à Pesquisa (PIP) administrado e financiado pela mesma. Agradeço também à minha família, que sempre incentivou meus estudos, em especial a Ulysses que, em todos os momentos, esteve caminhando ao meu lado, oferecendo apoio e auxílio. À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) que, através do programa Ciências sem Fronteiras foi de inestimável contribuição à minha formação acadêmica.

“A tarefa não é tanto ver aquilo que ninguém viu, mas pensar o que ninguém ainda pensou sobre aquilo que todo mundo vê.”

— Arthur Schopenhauer

Resumo

Dado um grafo $G = (V, E)$ composto por um conjunto de vértices V e um conjunto de arestas E , o Problema da Coloração de Grafos (PCG) consiste em atribuir uma cor a cada vértice do grafo, de modo que vértices adjacentes não possuam a mesma cor, utilizando o menor número de cores possível. Esse é um dos problemas mais estudados na área de Otimização Combinatória devido a suas inúmeras aplicações. Este trabalho propõe a aplicação de uma heurísticas matemáticas, *mipHeuristic*, baseada em decomposição para resolver o Problema da Coloração de Grafos, faz uma comparação dos resultados da heurística matemática proposta com algoritmos já consagrados na literatura, como a heurística de refinamento *recol* e a meta-heurística *Simulated Annealing*, e apresenta a análise dos resultados obtidos em relação ao estado da arte. Os resultados obtidos sugerem que a heurística matemática *mipHeuristic* funciona melhor quando aplicada à instâncias menores e que a utilização de heurísticas matemáticas para o Problema da Coloração de Grafos (PCG) apresenta resultados promissores.

Palavras-chaves: problema da coloração de grafos. heurísticas. heurística matemática. decomposicao.

Abstract

Given a graph $G = (V, E)$ composed of a set of vertex $V = \{v_1, v_2, \dots, v_n\}$ and a set of edges $E = \{e_1, e_2, \dots, e_m\}$, the *Graph Coloring Problem* (GCP) consists in assigning a color to each vertex of the graph, so that adjacent vertex, using the smallest number of colors that is possible. This is one of the most studied problems in Combinatorial Optimization due to its numerous applications. This work proposes the application of a math-heuristic based on decomposition to solve the *Graph Coloring Problem*, makes a comparison of the results of the proposed math-heuristic with algorithms already established in the literature, such as, the heuristic refinement recol and the meta-heuristic Simulated Annealing, and presents the analysis of the results obtained in relation to the state of the art. The results suggest that the mathematical heuristics mipHeuristic works best when applied to smaller instances and that the use of mathematical heuristics for Problema da Coloração de Grafos (PCG) presents promising results.

Key-words: graph coloring problem. heuristic. math-heuristic. decomposition.

Lista de ilustrações

Figura 1 – Problema da Ponte de Königsberg	18
Figura 2 – <i>Simulated Annealing</i> - Probabilidade x Temperatura	34

Lista de tabelas

Tabela 1 – Caracterização das instâncias (Grupo 1)	40
Tabela 2 – Caracterização das instâncias (Grupo 2)	41
Tabela 3 – Fatores e Níveis - SA	42
Tabela 4 – Experimentos - SA	43
Tabela 5 – Fator e Níveis - <i>mipHeuristic</i>	44
Tabela 6 – Experimentos <i>mipHeuristic</i>	44
Tabela 7 – Geração da Solução Inicial (Grupo 1)	45
Tabela 8 – Geração da Solução Inicial (Grupo 2)	46
Tabela 9 – Resultados <i>mipHeuristic</i> (Grupo 1)	46
Tabela 10 – Resultados <i>mipHeuristic</i> (Grupo 2)	47
Tabela 11 – Heurísticas de Refinamento (Grupo 1)	48
Tabela 12 – Heurísticas de Refinamento (Grupo 2)	48
Tabela 13 – Comparação com a Literatura (Grupo 1)	49
Tabela 14 – Comparação com a Literatura (Grupo 2)	49
Tabela 15 – Experimentos <i>recol</i> (Grupo 1)	56
Tabela 16 – Experimentos <i>recol</i> (Grupo 2)	56
Tabela 17 – Experimentos SA (Grupo 1)	57
Tabela 18 – Experimentos SA (Grupo 2)	58

Índice de algoritmos

1	Algoritmo Guloso	30
2	Algoritmo <i>D-Satur</i>	31
3	Heurística Simples	33
4	Simulated Annealing	35
5	Vizinho	36
6	Função Objetivo	37
7	Heurística <i>mipHeuristic</i>	38

Lista de abreviaturas e siglas

FF *First Fit*

GCP *Graph Coloring Problem*

GPS *Global Positioning System*

IDE Ambiente de Desenvolvimento Integrado

IDO *Incident Degree Ordering*

LDO *Largest Degree Ordering*

PCG Problema da Coloração de Grafos

PL Programação Linear

PLI Programação Linear Inteira

PLIM Programação Linear Inteira Mista

PLIP Programação Linear Inteira Pura

SA *Simulated Annealing*

SDO *Saturated Degree Ordering*

USAF *US Air Force*

Lista de símbolos

$\mathcal{O}$	Complexidade do algoritmo
Δ	Diferença entre a função objetivo de duas soluções diferentes
P	Probabilidade de uma solução pior ser aceita
e	Exponencial
k	Constante de Boltzmann
T	Temperatura corrente

Sumário

1	INTRODUÇÃO	18
1.1	O problema	19
1.2	Objetivos	20
1.3	Metodologia	20
1.4	Organização do trabalho	21
2	FUNDAMENTAÇÃO TEÓRICA	22
2.1	Problema da Coloração de Grafos	22
2.2	Programação Linear	22
2.2.1	Programação Linear Inteira	22
2.3	Heurísticas	23
2.3.1	Meta-heurísticas	24
2.3.2	Heurísticas Matemáticas	24
3	REVISÃO BIBLIOGRÁFICA	26
4	DESENVOLVIMENTO	29
4.1	Formulação Matemática	29
4.2	Solução Inicial	29
4.2.1	Algoritmo Guloso	30
4.2.2	Algoritmo <i>D-Satur</i>	31
4.3	Heurísticas de Refinamento	32
4.3.1	Heurística Simples	32
4.3.2	<i>Simulated Annealing</i>	33
4.3.2.1	Vizinhança	35
4.3.2.2	Função Objetivo	36
4.3.3	Heurística Matemática	38
5	EXPERIMENTOS COMPUTACIONAIS	39
5.1	Ambiente Computacional	39
5.2	Caracterização das Instâncias	39
5.3	Planejamento Experimental	41
5.3.1	Definição dos parâmetros para a meta-heurística <i>Simulated Annealing</i>	41
5.3.2	Definição dos parâmetros para a heurística matemática <i>mipHeuristic</i>	43
5.4	Resultados Obtidos	44
5.4.1	Geração da Solução Inicial	44

5.4.2	Heurística Matemática - <i>mipHeuristic</i>	46
5.5	Análise de Resultados	47
5.5.1	Comparação das Heurísticas de Refinamento	47
5.5.2	Comparação com a Literatura	49
5.5.3	Discussão de Resultados	50
6	CONCLUSÃO	51
	REFERÊNCIAS	52
	APÊNDICES	55
	APÊNDICE A – EXPERIMENTOS <i>RECOL</i>	56
A.1	Testes Heurística de Refinamento - <i>recol</i>	56
	APÊNDICE B – EXPERIMENTOS <i>Simulated Anealing</i> (SA)	57
B.1	Testes Meta-Heurística - <i>Simulated Anealing</i> (SA)	57

1 Introdução

Um grafo é uma estrutura matemática composta por um conjunto de vértices e um conjunto de arestas, que são ligações entre esses vértices. Os grafos possuem diversas aplicações reais, além do seu enorme valor teórico, o que o torna objeto de interesse de diversos estudos e pesquisas. Sistemas de navegação por *Global Positioning System* (GPS) usam algoritmos de caminho mínimo em grafos para traçar as melhores rotas, estudos em redes sociais são feitos através da modelagem da rede como um grafo e da análise das conexões, redes de computadores têm o pacote de dados otimizados através de algoritmos em grafos, dentre outras aplicações (JENSEN; TOFT, 2011).

Ostroski e MENONCINI (2009) apresentam, em seu trabalho, um pouco da história da Teoria dos Grafos, que surgiu no ano de 1736 por meio do problema das pontes de Königsberg, uma cidade da antiga Prússia do século XVIII. Na cidade haviam duas ilhas que eram ligadas por sete pontes entre elas e a parte continental. O problema consistia em atravessar todas as pontes sem repeti-las. Para resolvê-lo, Leonard Euler modelou o problema transformando o caminho das pontes em retas e suas interseções em pontos, criando assim um grafo. A Figura 1 mostra as pontes da cidade de Königsberg e o grafo criado por Euler.

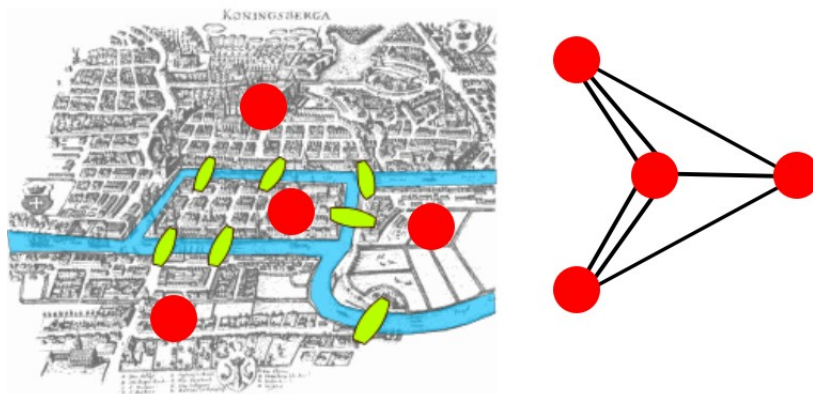


Figura 1 – Problema da Ponte de Königsberg

Fonte – Retirado de Dallaqua (2016)

Em 1847, Gustav Robert Kirchhoff utilizou modelos de grafos para estudar circuitos elétricos, criando a Teoria das Árvores. Desta forma, outros cientistas começaram a utilizar a ideia de árvores para outras aplicações. Um importante auxílio para o desenvolvimento da Teoria de Grafos foi dado por Willian Rowan Hamilton que inventou um jogo que consistia na busca de um percurso fechado que passasse por todos os vértices de um decaedro regular, de forma que cada um fosse visitado apenas uma única vez e assim deu origem aos estudos de grafos hamiltonianos. A teoria dos grafos teve outro grande avanço

em 1970 devido ao desenvolvimento dos computadores que impulsionaram o uso da teoria e algoritmos em grafos para resolver problemas do mundo real. No ano de 1968 foram apresentados trabalhos referentes a Teoria dos Grafos no I Simpósio Brasileiro de Pesquisa Operacional, dando início aos estudos e a realizações de trabalhos sobre o assunto no Brasil (OSTROSKI; MENONCINI, 2009).

Dentro da teoria dos grafos, existe o Problema da Coloração de Grafos (PCG) - em inglês *Graph Coloring Problem* - que consiste em atribuir uma cor aos vértices de um grafo de modo que nenhum vértice adjacente, ou seja, vértices que são ligados por uma aresta em comum possuam a mesma cor, utilizando o menor número de cores possíveis.

Nesse trabalho é proposta a utilização de uma heurística matemática baseada em decomposição como heurística de refinamento (SOUZA, 2008) para o PCG, utilizando o algoritmo *D-Satur* (BRÉLAZ, 1979) para encontrar uma solução inicial. O trabalho apresenta também uma comparação entre a heurística matemática proposta (*mipHeuristic*), uma heurística de refinamento (*recol*) e a meta-heurística *Simulated Annealing* (KIRKPATRICK; JR; VECCHI, 1983), além de fazer uma análise dos resultados obtidos comparando-os com resultados encontrados na literatura.

1.1 O problema

A coloração de grafos surgiu em 1852 quando o estudante Francis Guthrie propôs o problema das quatro cores, no qual questionava se qualquer mapa poderia ser colorido com apenas quatro cores, de maneira que países com fronteiras em comum não fossem coloridos com a mesma cor. Em 1879, o matemático Kempe apresentou uma solução para o problema através da primeira demonstração do teorema das 4 cores e, em 1880, o físico matemático Peter Guthrie Tait iniciou o estudo da coloração das arestas com o objetivo de solucionar esse problema, apresentando assim, outra demonstração para o teorema. Porém, em 1890, o matemático Percy John Heawood apontou erros na demonstração feita por Kemp e, em 1891, o matemático Julius Peter Christian Petersen também mostrou que a demonstração de Tait era falha. Ao longo dos anos, mais estudos e contribuições foram feitas com o intuito de resolver o problema proposto por Guthrie. Em 1976, os matemáticos Hapel e Akken o resolveram utilizando computadores em sua demonstração, o que gerou debate na matemática. Eles definiram 1936 configurações que deveriam ser verificadas por computador, usando, aproximadamente, 1200 horas de computação (LUIZ, 2015). Assim, foi provado que qualquer grafo planar, ou seja, um grafo que pode ser desenhado no plano sem que nenhuma de suas arestas se cruzem, é 4-colorível. Entretanto, o problema da coloração de grafos vai mais além, ele busca encontrar um número mínimo de cores para colorir um grafo qualquer.

O problema da coloração de grafos se destaca dentro da teoria dos grafos, sendo

um dos problemas mais estudados na comunidade acadêmica. Ele é um famoso problema da classe NP-difícil (GAREY; JOHNSON, 2002), e, desse modo, ainda não se conhece um algoritmo que apresente uma resposta exata para ele em tempo polinomial. Assim, diversas técnicas são utilizadas para encontrar a melhor solução possível para esse problema.

Por pertencer a essa classe de problemas, o problema da coloração de grafos desperta muito interesse da comunidade científica de Computação e Pesquisa Operacional. Além disso, vários trabalhos relacionam o PCG com problemas do cotidiano como, por exemplo, agendamento de horários educacionais (NEUFELD; TARTAR, 1974), planejamento de projetos (ZUFFEREY, 2015), alocação de registradores (CHAITIN, 1982; CHOW; HENNESSY, 1990), atribuições de canais de redes sem fio (LIMA et al., 2014), agendamento de tarefas em máquinas (KOUIDER et al., 2015), agendamento de competições esportivas (DREXL; KNUST, 2007), planejamento de tráfego aéreo (BARNIER; BRISSET, 2004), entre outras aplicações.

1.2 Objetivos

O principal objetivo do presente trabalho é estudar e desenvolver uma heurística matemática baseada em decomposição aplicada ao Problema da Coloração de Grafos. Como objetivos específicos, cita-se:

- Realizar um levantamento bibliográfico das melhores estratégias computacionais para solucionar o problema;
- Propor a primeira heurística matemática para a Coloração de Grafos;
- Analisar a viabilidade e o comportamento do algoritmo para grafos oriundos de variadas aplicações.

1.3 Metodologia

Esse trabalho tem como objeto de pesquisa o PCG e o desenvolvimento de uma nova estratégia para resolver o problema. Com esse objetivo, foram utilizadas instancias do *DIMACS Implementation Challenge* (DIMACS, 1995) para avaliar os resultados da estratégia desenvolvida. Esse conjunto de instâncias contempla problemas de variadas dimensões, graus de dificuldade e aplicações. Além disso, é um conjunto amplamente utilizado pela comunidade científica. As etapas principais do desenvolvimento do trabalho foram assim definidas:

Revisão da literatura Levantamento e estudo da bibliografia dos trabalhos relevantes relacionados aos conceitos utilizados e algoritmos estado-da-arte para a solução do problema.

Reprodução do algoritmo *D-Satur* Implementação do algoritmo *D-Satur* e de outros algoritmos relevantes na literatura recente.

Desenvolvimento de formulação de programação matemática Formulação do problema como um problema de Programação Inteira e sua respectiva implementação.

Implementação da heurística matemática Estudo de estratégias para a decomposição do problema e implementação da heurística matemática.

Implementação da meta-heurística Implementação da meta-heurística *Simulated Annealing*.

Avaliação computacional das técnicas desenvolvidas Realização de experimentos computacionais sistemáticos sobre as técnicas desenvolvidas no trabalho e análise desses resultados para concluir sobre a eficiência do método proposto.

Análise Análise e discussão dos experimentos realizados.

1.4 Organização do trabalho

O restante deste trabalho é organizado como se segue. O [Capítulo 2](#) apresenta uma fundamentação teórica dos algoritmos e métodos utilizados. O [Capítulo 3](#) apresenta a revisão bibliográfica do trabalho. No [Capítulo 4](#) está a o desenvolvimento do trabalho, bem como os algoritmos desenvolvidos. O [Capítulo 5](#) mostra os experimentos realizados e os resultados obtidos. Por fim, o [Capítulo 6](#) apresenta a conclusão do trabalho.

2 Fundamentação Teórica

2.1 Problema da Coloração de Grafos

Rego e Santos (2001 apud CHARTRAND; ZHANG, 2008) definem formalmente o Problema da Coloração de Grafos (PCG) da seguinte maneira: dado um grafo não-direcionado $G = (V, E)$, onde V é um conjunto de vértices e E um conjunto de arestas que incidem sobre os vértices de G , o PCG consiste em atribuir k -cores para os vértices de G , de modo que dois vértices adjacentes não possuam a mesma cor. A coloração pode ser considerada uma função $c : V(G) \rightarrow \mathbb{N}$, sendo $\mathbb{N}$ o conjunto dos números inteiros positivos, tal que, $c(u) \neq c(v)$ se u e v são adjacentes em G . O PCG pode ser formulado de duas formas diferentes, como um problema de decisão cujo objetivo é identificar se um grafo é k -colorível, ou com o objetivo de encontrar um valor k que seja mínimo. Esse trabalho trata dessa última formulação.

2.2 Programação Linear

A Programação Linear (PL) é uma técnica da Matemática Aplicada. Os modelos de PL são a base da investigação operacional. O termo “programação” se relaciona com a planificação e resolução de tarefas, e o termo “linear” diz sobre as expressões ou condições envolvidas em cada modelo serem lineares. O objetivo da PL é otimizar os problemas de decisão por meio da utilização de modelos que representam o problema, em que a solução ótima é um mínimo ou um máximo que deve ser alcançado observando as condições existentes. Em 1826, Jean-Baptiste Joseph Fourier deu origem aos estudos sobre otimizar uma função linear sujeita a restrições. Em 1939, o matemático russo Leonid Vitalyevich Kantorovich apresentou um algoritmo para otimização na administração das organizações utilizando a PL, porém esse só se tornou conhecido em 1950. Em 1947, George Dantzig, consultor matemático do *US Air Force* (USAF) apresentou uma forma sistematizada de resolução de problemas de Programação Linear, o Algoritmo Simplex. As inovações da segunda metade do século XX foram base para o desenvolvimento da PL aplicada em diversos problemas de decisão em diferentes domínios (PAIVA, 2008).

2.2.1 Programação Linear Inteira

Um problema de Programação Linear Inteira (PLI) é um programa de PL em que as variáveis são discretas, ou seja, assumem valores inteiros. Quando todas as variáveis são totalmente inteiras o problema é conhecido como problema de Programação Linear Inteira

Pura (PLIP) e quando apenas algumas variáveis são inteiras trata-se de um problema de Programação Linear Inteira Mista (PLIM) (ALVES; DELGADO, 1997).

2.3 Heurísticas

O termo heurístico é derivado da palavra grega *heuriskein* que significa descobrir. Uma heurística pode ser definida como um procedimento algorítmico desenvolvido a partir de um modelo cognitivo, por meio de regras baseadas na experiência dos desenvolvedores. Elas apresentam um conhecimento sobre o comportamento do problema e, dessa forma, exploram um número bem menor de soluções em comparação com os métodos exatos (CORDENONSI, 2008).

Os métodos heurísticos são algoritmos que buscam resolver problemas. Eles encontram a melhor solução possível naquele contexto e em um tempo computacional razoável mas não garantem a solução ótima. Em seu trabalho, Bueno (2009) explica que essa subjetividade dos métodos heurísticos é uma particularidade análoga à inteligência humana que, muitas vezes, resolve problemas sem conhecê-los com precisão. Um exemplo é que ao adoçar uma bebida pouco é conhecido sobre as propriedades do soluto e do solvente, porém, a melhor solução imediata para adoçar é encontrada e adotada, em detrimento de uma solução comprovadamente ótima e precisa.

Semelhante ao que foi apresentado por Arroyo et al. (2002) em seu trabalho, os métodos heurísticos utilizados para o desenvolvimento desse trabalho também podem ser divididos em três classes com base na forma como exploram o espaço de soluções do problema:

Classe das heurísticas construtivas Constroem uma solução inicial para o PCG utilizando regras específicas associadas com a estrutura do problema para adicionar componentes à essa solução. Essas heurísticas são especializadas em um dado problema, nesse caso, no PCG.

Classe das heurísticas de refinamento Já iniciam com uma solução completa do problema, e a partir de então, constroem uma vizinhança contendo todas as soluções alcançáveis através de uma regra de modificação da solução inicial. A partir dessa vizinhança, escolhe-se uma nova solução que possua uma solução melhor que a solução inicial e essa passa a ser a nova solução inicial, continuando esse processo até encontrar um ótimo local ou, caso seja uma heurística que tenha um determinado tempo de execução predefinido, até atingir o tempo de execução previamente determinado. Dessa forma, é possível perceber que a eficiência dessa classe de heurísticas depende da escolha da solução inicial e da definição de vizinhança. Uma vez alcançado um

ótimo local, as heurísticas não são capazes de escapar desse ótimo em busca de um ótimo global.

Classe das meta-heurísticas Métodos que possuem uma estrutura com componentes genéricos que são adaptados ao problema que se quer resolver. Esses métodos permitem a escolha estratégica de soluções piores que as já encontradas, com o objetivo de escapar dos ótimos locais em busca dos ótimos globais.

2.3.1 Meta-heurísticas

Meta-heurística como um método científico para resolver problemas é um fenômeno moderno. O termo meta-heurística foi, provavelmente, introduzido em 1986 por Fred Glover que descrevia o *Tabu Search* como sendo uma “meta-heurística” sobreposta a outra heurística (GLOVER, 1986). As meta-heurísticas são estratégias de alto nível para explorar espaços de busca usando métodos diferentes. Elas guiam o processo da busca com o objetivo de explorar eficientemente o espaço de busca disponível, a fim de encontrar soluções ótimas. As técnicas que constituem algoritmos meta-heurísticos variam de procedimentos de busca local simples para processos complexos de aprendizagem. Elas possuem mecanismos para evitar ficarem presas em ótimos locais e podem usar experiência de pesquisa, incorporada em alguma forma de memória, para orientar a pesquisa (BLUM; ROLI, 2003).

De acordo com Freitas et al. (2010 apud GLOVER, 2003), o processo geral de execução de uma meta-heurística trata da busca de soluções guiando-se por funções de satisfação, sendo a função de satisfação uma função matemática que atribui um valor a cada solução do espaço de busca. Esse processo faz com que apenas algumas das soluções existentes sejam visitadas para a determinação de uma solução. A forma como ocorre o processo de busca é específico para cada meta-heurística. Entretanto, não existe garantia de que a solução retornada por uma meta-heurística seja a melhor possível para aquele problema, ou seja, a solução ótima. Sendo assim, o uso das meta-heurísticas se justifica devido à complexidade interna do problema, que impede a aplicação de técnicas exatas, e à quantidade muito grande de possíveis soluções, que impedem a utilização de técnicas exaustivas.

2.3.2 Heurísticas Matemáticas

Uma heurística matemática é um algoritmo que se utiliza de um modelo de PLI para obter soluções heurísticas para um problema, ou seja, propõe uma solução possível para um problema cuja solução exata não pode ser encontrada em tempo viável. Nesse contexto, a abordagem utilizada nesse trabalho, é decompor o problema original e otimizar um subproblema por iteração. Dessa forma, o problema é subdividido em problemas menores (subproblemas) e a heurística busca a solução do subproblema, de forma que, ao final,

obtem-se uma solução para o problema global (PUREZA, 2010 apud BALL; MAGAZINE, 1981). Denomina como métodos de composição aqueles em que os subproblemas se resolvem em sequência, e métodos de partição quando os subproblemas são independentes entre si.

Os primeiros relatos de aplicação dessa estratégia são recentes, o que indica que há ainda muitos estudos e aplicações a serem feitos. Até onde se tem conhecimento essa abordagem nunca foi aplicada ao PCG apesar de ter obtido resultados expressivos quando aplicado a problemas que possuem estrutura similar (FONSECA; SANTOS; CARRANO, 2016; SANTOS et al., 2016).

3 Revisão bibliográfica

Em seu trabalho, [Rego e Santos \(2001\)](#) usaram uma heurística construtiva combinada com uma meta-heurística para resolver o [PCG](#). A heurística construtiva gerou uma solução inicial e a meta-heurística foi usada para refinar a solução previamente gerada. Em primeiro lugar, os autores definiram o conceito de vizinhança. Um vizinho s' de uma solução s foi gerado através do movimento de troca da cor de um vértice. O algoritmo construtivo apresentado no artigo, cria uma classe de cor c_1 e então percorre todos os vértices que não tenham nenhuma coloração e não possuem conflito com algum vértice que já esteja na classe c_1 . Quando um vértice é inserido em c_1 ele não pode ser inserido em outra classe de cor e seus vértices adjacentes ficam proibidos de serem inseridos na classe c_1 . Esse algoritmo gera uma solução viável que, em uma segunda etapa, passa pela meta-heurística de refinamento. Essa meta-heurística usa um processo de busca local que começa reduzindo o número de cores obtidas na solução inicial de k para $k - 1$ selecionando aleatoriamente uma classe de cor e removendo todos os elementos dessa classe. Logo após, ele percorre os elementos que estão sem cor buscando gerar uma solução viável apenas com as classes de cores ainda existentes. Esse processo se repete até que não seja possível mais reduzir o número de classes de cores. Para os testes, os autores selecionaram instâncias de grafos com números de vértices variando de 125 a 1000, número de arestas aleatório e densidades igual a 0, 1, 0, 5, e 0, 9. Os resultados mostraram que a heurística construtiva proposta apresenta seus melhores resultados com grafos de baixa densidade e a meta-heurística que usa busca local é mais eficiente em grafos mais densos.

[Al-Omari e Sabri \(2006\)](#) apresentam duas novas heurísticas baseada em algoritmos já conhecidos. O primeiro algoritmo proposto é uma modificação do algoritmo *Largest Degree Ordering* ([LDO](#)) - algoritmo que escolhe o vértice com o maior grau¹ para ser colorido primeiro - através da combinação dele com o algoritmo *Incident Degree Ordering* ([IDO](#)) - algoritmo que escolhe o vértice com maior grau de incidência² para ser colorido primeiro. Esse primeiro algoritmo funciona como o [LDO](#), porém quando encontra dois vértices com o mesmo grau usa-se o [IDO](#) para escolher entre eles qual será colorido primeiro. O segundo algoritmo combina os algoritmos *Saturated Degree Ordering* ([SDO](#)) - algoritmo que escolhe o vértice com maior grau de saturação³ para ser colorido primeiro - e o algoritmo [LDO](#). O segundo algoritmo funciona como o [SDO](#), porém quando encontra dois vértices com o mesmo grau usa-se o [LDO](#) para escolher entre eles qual será colorido primeiro. Para os testes foram usados grafos aleatórios com diferentes número de vértices e densidades e os

¹ Grau de um vértice: número de arestas incidentes ao vértice

² Grau de incidência de um vértice: número de vértices adjacentes já coloridos

³ Grau de saturação de um vértice: número de cores diferentes que já foram usadas nos vértices adjacentes a esse vértice

algoritmos usados para comparar os resultados foram os LDO, IDO e SDO originais e o algoritmo *First Fit* (FF), que atribui sequencialmente a cada vértice a menor cor legal. Os testes mostraram que os novos algoritmos apresentaram bons resultados.

O trabalho apresentado por Lozano (2007) mostra a modelagem matemática e aplicações do problema de coloração em grafos. O PCG foi formulado matematicamente de duas formas, ambas através de modelos de PLI. A primeira formulação, chamada Formulação por Conjuntos Independentes Maximais, associa uma variável binária para cada conjunto independente maximal⁴ do grafo, com o objetivo de determinar a quantidade mínima de conjuntos independentes maximais necessária para cobrir todos os vértices do grafo. Já a segunda formulação é baseada em restrições de atribuição, com o objetivo de minimizar o número de cores necessárias para colorir o grafo. Essa formulação é simétrica, uma vez que qualquer permutação de cores fornece soluções factíveis e com o mesmo valor da função objetivo. Porém, essa simetria define um número exponencial de colorações, tonando sua utilização inviável para instâncias grandes. Para solucionar essa questão, o autor apresentou algumas inequações que melhoram essa segunda formulação, excluindo algumas soluções simétricas do conjunto solução do problema. Para a resolução, foram apresentados dois métodos, o primeiro através da geração de colunas e o segundo utilizando um algoritmo *Branch-and-Cut*. O primeiro método considera apenas um subconjunto de colunas do problema principal, e então avalia se existem colunas que não estão no problema principal, mas que podem melhorar a função objetivo ao serem adicionadas. Já no segundo, o algoritmo *Branch-and-Cut* inclui uma fase de pré-processamento com o objetivo de eliminar soluções sub-ótimas e restrições redundantes diminuindo o tamanho do problema. Essa fase é iniciada através de uma heurística simples que busca o maior clique⁵ do grafo, obtendo assim um limite inferior válido para o número cromático do grafo baseado no tamanho do clique, dado que os vértices do clique devem receber cores diferentes. Um limite superior para o número cromático é encontrado através da solução encontrada pelo algoritmo *D-Satur*. Os testes mostraram que o algoritmo *Branch-and-Cut* é capaz de resolver instâncias de coloração em grafos que o *D-Satur* não consegue, e também que esse método produz uma melhoria nos limites inferior e superior do número cromático do grafo.

Douiri e Elbernoussi (2015) propuseram em seu trabalho um novo algoritmo genético híbrido baseado em uma heurística de busca local chamada DBG para encontrar uma solução para o PCG. Esta abordagem parte de um número de cores inicial k . A escolha dos indivíduos $p = \langle c(1), c(2), \dots, c(N) \rangle$ corresponde a uma atribuição das k cores a todos os vértices do grafo G . Se é encontrada uma k -coloração válida, o número k é atualizado ($k \leftarrow k - 1$). A inicialização do número de cores k é feita usando um algoritmo que

⁴ Conjunto independente maximal é o maior subconjunto de vértices em que não exista nenhuma aresta entre qualquer par de elementos desse subconjunto

⁵ Clique é um conjunto de vértices dois a dois adjacentes, ou seja, para cada par de vértices, existe uma aresta os conectando

particiona os vértices do grafo em k classes de cores. Essa heurística é baseada na redução de variáveis através de um multiplicador w da restrição substituta. Os indivíduos da população são gerados através do mesmo algoritmo, porém, com uma ligeira modificação com o objetivo de diversificar a população inicial. A função objetivo é relacionada ao número de conflitos para cada k fixo. Busca-se que, a cada estágio, o número de conflitos seja reduzido até chegar a zero, o que corresponde a uma coloração válida. Os autores apresentaram também os operadores de seleção, *cossover*, e mutação. Nos experimentos, para o algoritmo genético, foi utilizada uma população igual a 120, a probabilidade de *crossover* $p_c = 0,7$, probabilidade de mutação $p_m = 0,15$ e número de iterações variando entre 250 e 2000. Os experimentos computacionais mostraram resultados competitivos para diversos grafos encontrados na literatura, inclusive para grafos com 900, 1000, 2000 e 4000 vértices.

Em seu artigo, [Marappan, Sethumadhavan e Srihari \(2016\)](#) mostram dois novos métodos de aproximação para resolver o PCG baseados na mediana dos graus dos vértices do grafo. Esses métodos foram criados combinando as estratégias de algoritmos gulosos e divisão e conquista, sendo a estratégia gulosa utilizada para minimizar o número de vértices conflitantes. O primeiro método apresentado inicialmente calcula a mediana do grafo, colore todos os vértices com grau maior ou igual a mediana e, após, colore os vértices remanescentes. No segundo método, a estratégia de divisão e conquista é aplicada para dividir os vértices do grafo em dois subconjuntos menores: os da esquerda e os da direita. Inicialmente, é chamado o procedimento de coloração para o subconjunto da esquerda e, no momento de colorir o subconjunto da direita, é levado em consideração os vértices adjacentes ao grupo da esquerda para evitar conflitos. Os resultados apresentados mostraram que o método que utiliza divisão e conquista (método 2) teve uma melhor performance para a maioria dos grafos testados em comparação com o método 1.

[Macedo, Lima e Tolentino \(2017\)](#) apresentaram uma meta-heurística baseada em *Simulated Annealing* aplicada ao PCG para otimização de processos. Sua proposta foi verificar a fidelidade e efetividade da meta-heurística SA baseado na comparação desse método com o método de coloração sequencial. Foi utilizado um método de perturbação e a procura por vizinhos próximos para garantir que outras partes do espaço de soluções possam ser averiguadas, e assim analisar o máximo de soluções locais. Como parâmetros de entrada, foi passada uma temperatura inicial = 100, temperatura final = 0, taxa de perturbação = 10, número de soluções = 100, taxa de resfriamento = 10 e temperatura atual = temperatura inicial. Para os testes foram montados três grafos com diferentes números de arestas que, após obter uma solução inicial, foram submetidos ao SA e suas soluções comparadas. O primeiro grafo possuía 23 vértices e 37 arestas, o segundo 23 vértices e 42 arestas e o terceiro com 23 vértices e 55 arestas. Com os resultados obtidos, foi possível perceber que o método utilizado obteve melhores resultados comparado com a coloração sequencial, inclusive em grafos com quantidades maiores de arestas adjacentes.

4 Desenvolvimento

4.1 Formulação Matemática

Dado um grafo $G = (V, E)$ onde $V = \{1, \dots, n\}$ é o conjunto de vértices do grafo, $E = \{(v_1, v_2), \dots\}$ é o conjunto de arestas e $C = \{1, \dots, m\}$ é o conjunto de cores disponíveis, o problema da coloração de grafos consiste em encontrar uma coloração que usa o menor número de cores possíveis, para colorir os vértices do grafo. A formulação matemática apresentada nesse trabalho tem o objetivo de minimizar o número de cores necessárias para colorir um grafo G , de forma que o nenhum vértice V receba uma cor semelhante ao seu adjacente.

As variáveis do problema foram modeladas como nas [Equação 4.1](#) e [Equação 4.2](#) apresentadas a seguir:

$$x_{ij} = \begin{cases} 1, & \text{se vértice } i \text{ tem cor } j \\ 0, & \text{caso contrário} \end{cases} \quad (4.1)$$

$$w_j = \begin{cases} 1, & \text{se cor } j \text{ foi usada} \\ 0, & \text{caso contrário} \end{cases} \quad (4.2)$$

Dessa forma, a função objetivo consiste em encontrar o número cromático mínimo para o grafo, como mostra a [Equação 4.3](#):

$$\min \sum_{j \in C} w_j \quad (4.3)$$

Sujeito a duas restrições: (i) a de que um vértice pode receber apenas uma cor, apresentada na [Equação 4.4](#); (ii) dois vértices adjacentes não podem receber cores iguais, apresentada na [Equação 4.5](#).

$$\min \sum_{j \in C} x_{ij} = 1, \quad \forall i \in V \quad (4.4)$$

$$x_{uj} + x_{vj} \leq 1, \quad \forall (u, v) \in E, \forall j \in C \quad (4.5)$$

4.2 Solução Inicial

O primeiro passo para o desenvolvimento do trabalho foi encontrar uma solução inicial para as instâncias utilizadas. Nessa busca, foram utilizadas heurísticas construtivas, que são aquelas que encontram soluções iniciais para o problema utilizando regras específicas

associadas ao [PCG](#). Uma heurística construtiva para o Problema da Coloração de Grafos, consiste em construir uma boa coloração para os vértices de forma que nenhum vértice adjacente possua a mesma cor. Ela parte de uma solução vazia e determina um critério para a atribuição de cores, até que todos os vértices sejam coloridos respeitando a restrição dada. Para esse trabalho foram utilizadas duas heurísticas construtivas, o algoritmo guloso e o algoritmo *D-Satur*.

4.2.1 Algoritmo Guloso

Um algoritmo guloso tem como característica escolher a melhor solução existente no momento, ou seja, ele faz uma escolha localmente ótima com o intuito de que ela leve para uma solução ótima global. Esses algoritmos tem como vantagem a simplicidade e são de fácil implementação. Suas decisões são baseadas nas informações disponíveis na iteração corrente e, por isso, podem ser chamados míopes. Os algoritmos gulosos nem sempre produzem a melhor solução mas são de rápida execução ([ROCHA; DORINI, 2004](#)).

O algoritmo guloso implementado nesse trabalho usou a seguinte estratégia: através de uma lista de cores possíveis ele identificou a menor cor que poderia ser atribuída àquele vértice sem ferir a propriedade de coloração. Cada vértice é analisado em uma iteração e recebe a menor cor disponível naquele momento. A implementação foi feita como apresentado no Algoritmo 1:

Algoritmo 1: Algoritmo Guloso

Entrada: $G(V, E)$

Saída: s

inicio

$s \leftarrow \{1, \dots, |V|\};$

$cores \leftarrow \{1, \dots, |V|\};$

para cada $u \in V$ **fazer**

$coresPossiveis \leftarrow cores;$

para cada $i \in cores$ **fazer**

para cada *adjacente* v de u **fazer**

se $(s[v] = i)$ e $(i \in coresPossiveis)$ **então**
 $coresPossiveis \leftarrow coresPossiveis - \{i\};$

fim

fim

$s[u] \leftarrow coresPossiveis[0];$

fim

fim

retorna $s;$

fim

4.2.2 Algoritmo *D-Satur*

O algoritmo *D-Satur* - *Degree of Saturation* - foi apresentado por Daniel Brélaz em 1979 em seu trabalho chamado “*New Methods to Color the Vertices of a Graph*”. O *D-Satur* foi descrito como uma nova heurística para colorir os vértices de um grafo baseada na comparação entre os graus de saturação dos vértices. É um algoritmo exato para grafos bipartidos. (BRÉLAZ, 1979). Hoje, essa heurística é umas das técnicas mais aplicadas para solucionar o problema da coloração de grafos. Ele é sequencial e atribui a menor cor possível a um vértice por iteração. A seleção do vértice a ser processado é definida pelo seu grau de saturação, onde vértices com maior grau de saturação são escolhidos primeiro. Possui complexidade $\mathcal{O}(n^2)$ e, como é um método de baixa complexidade, apresenta bons resultados para grafos em geral (NETO; GOMES, 2015; DU, 2013). A implementação foi feita como apresentado no Algoritmo 2.

Algoritmo 2: Algoritmo *D-Satur*

Entrada: $G(V, E)$

Saida: s

inicio

$s \leftarrow \{-1, \dots, -1\};$

$cores \leftarrow \{1, \dots, |V|\};$

$vertices \in \text{vértices em ordem decrescente de grau de saturação};$

enquanto ($!vertices.isEmpty()$) **fazer**

$n \leftarrow vertices.remove(0);$

$u \leftarrow n.id;$

para cada $i \in cores$ **fazer**

para cada *adjacente* v de u **fazer**

se ($s[v] = i$) e ($i \in coresPossiveis$) **então**
 $coresPossiveis \leftarrow coresPossiveis - \{i\};$

fim

fim

$s[u] \leftarrow coresPossiveis[0];$

fim

$\text{grau de saturação de } u \leftarrow coresPossiveis[0];$

$vertices \in \text{vértices em ordem decrescente de grau de saturação};$

fim

retorna $s;$

fim

Para esse trabalho, a estratégia utilizada foi, primeiramente, ordenar os vértices do grafo. O primeiro critério de ordenação foi em ordem decrescente de grau de saturação, o segundo critério foi ordem decrescente de grau e o terceiro critério foi ordem crescente de id, ou seja, o número do vértice. Com os vértices já ordenados, o algoritmo atribuiu a

menor cor possível a cada vértice por iteração, sempre respeitando a restrição de nenhum vértice ter a mesma cor do seu adjacente. Nessa parte, o *D-Satur* funciona como um algoritmo guloso, porém, o que faz toda diferença é a ordenação por grau de saturação feita antes da atribuição das cores.

4.3 Heurísticas de Refinamento

As heurísticas de refinamento são técnicas de busca local baseadas em noção de vizinhança que tem o objetivo de melhorar uma solução já encontrada através de modificações nessa solução. Elas partem de uma solução inicial qualquer e caminham em cada iteração nos vizinhos dessa solução tentando melhorar a solução já construída. Essas heurísticas funcionam até encontrar um ótimo local, ou seja, uma solução melhor que a já obtida, ou até atingir determinado parâmetro, como por exemplo, uma quantidade de tempo definida pelo usuário (SOUZA, 2008).

Bueno (2009 apud COLIN, 2007) mostra uma analogia entre as heurísticas e o problema de localizar o ponto mais alto da terra. Na busca pela montanha mais alta, parte-se de um ponto qualquer na superfície e, durante o processo, várias montanhas são escaladas e suas alturas comparadas. Quando um ponto mais alto for encontrado (ótimo local), a montanha mais alta será atualizada para a nova descoberta. Esse processo ocorre até que as buscas se encerram por algum motivo, que pode ser a ausência de descobertas melhores durante um longo período, ou a falta de segurança, ou restrições financeiras e de tempo. Assim, o ponto mais alto é definido com a melhor descoberta. Porém, esse resultado não possui comprovação científica, ou seja, não se sabe se outra montanha mais alta ainda poderia ser descoberta. É dessa forma que funciona essas heurísticas: é uma busca contínua e empírica que encontra vários ótimos locais e apresenta o melhor resultado encontrado em determinadas condições.

Nesse trabalho, foram usadas duas heurísticas como heurísticas de refinamento, uma heurística simples chamada *recol* e a meta-heurística *Simulated Annealing*.

4.3.1 Heurística Simples

A heurística *recol* é uma heurística de refinamento simples que tem o objetivo de melhorar a solução inicial passada. Sua estratégia é remover uma cor da lista de cores utilizadas na solução inicial e tentar recolorir o grafo somente com o novo número de cores disponível. A atribuição da cor é considerada válida se nenhum dos vértices adjacentes possuir a mesma cor. A cada iteração uma cor é diminuída da solução inicial, levando a uma coloração com um menor número de cores, caso a solução inicial possa ser melhorada. A heurística *recol* funciona durante um tempo determinado, e no fim desse tempo ela retorna a melhor solução encontrada até o momento. Como todas as heurísticas de sua

classe, caso encontre um ótimo local, a *recol* não consegue sair em busca de um ótimo global, retornando, assim, o ótimo local encontrado. A implementação foi feita como apresentado no Algoritmo 3:

Algoritmo 3: Heurística Simples

Entrada: $s, G(V, E)$

Saida: s

inicio

$cores \leftarrow$ lista contendo as cores utilizadas em s ;

enquanto ($tempoDisponível > 0$) **fazer**

$u \leftarrow$ vértice aleatório de V ;

$c \leftarrow$ cor aleatória de $cores$, exceto a última;

$s' \leftarrow s$;

 atribua a cor c ao vértice u em s' ;

se (atribuição for válida) **então**

$coresCand \leftarrow$ lista contendo as cores utilizadas em s ;

se $|coresCand| \leq |cores|$ **então**

$s \leftarrow s'$

$cores \leftarrow coresCand$

fim

fim

fim

retorna s ;

fim

4.3.2 Simulated Annealing

O *Simulated Annealing* é uma meta-heurística baseada nos processos da metalurgia proposta em 1983 por Kirkpatrick, Jr e Vecchi (1983). Ela baseia em um processo chamado recozimento de metais que tem o objetivo de alterar a estrutura cristalina de metais dando a eles características mecânicas e estruturais desejadas. Nesse processo, os metais são aquecidos à uma temperatura muito elevada até alcançarem um estado líquido e são resfriados lentamente para formar cristais mais estáveis com menor energia interna. Da mesma forma como os átomos buscam aleatoriamente a condição de menor energia interna, o algoritmo busca a solução de menor custo, sendo assim, a energia mínima é equivalente a solução de menor custo nos problemas de otimização. No SA parte-se de uma solução viável e passa-se a aceitar soluções vizinhas. Em um primeiro momento, nas altas temperaturas, há grande probabilidade de qualquer solução vizinha ser aceita, mas a medida que ocorre o resfriamento, há maior probabilidade de aceitar soluções melhores (BUENO, 2009).

A cada iteração do método, é feita uma modificação aleatória no estado corrente e é gerado um novo estado a partir dessa modificação. Se a energia do estado novo é menor

que a do estado corrente, o estado novo passa a ser o corrente, e se o estado novo tem energia maior que a do estado corrente em Δ unidades, onde Δ é a variação de custo, é calculada a probabilidade de mudar o estado corrente para o novo estado. O cálculo usado para probabilidade é apresentado na [Equação 4.6](#) a seguir:

$$P = e^{-\Delta/(k * T)} \quad (4.6)$$

onde k é a constante de Boltzmann e T temperatura corrente. Este procedimento é repetido até se atingir o equilíbrio térmico. Na [Figura 2](#) é possível ver a probabilidade de aceitação de uma solução em função da temperatura ([CAMPOS, 2018](#)).

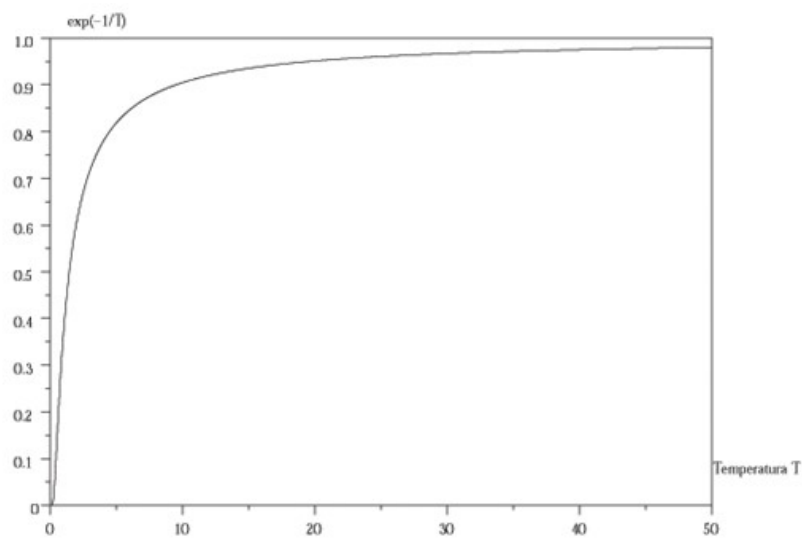


Figura 2 – *Simulated Annealing* - Probabilidade x Temperatura

Fonte – Retirado de [Campos \(2018\)](#)

Quando o equilíbrio térmico é atingido em uma dada temperatura, ela é diminuída e os passos são executados novamente. O algoritmo termina quando a temperatura se aproxima de zero. Com a temperatura elevada, é maior a probabilidade de aceitar soluções de piora, e essas soluções são aceitas com o objetivo de escapar de mínimos locais. Nesse momento, a solução sofre uma piora para que no futuro tenha mais chance de encontrar o ótimo global. A probabilidade de aceitar a solução de piora depende do parâmetro temperatura. No final do processo, praticamente não se aceita movimentos de piora e, dessa forma, o método passa a ter um comportamento semelhante ao método da descida/subida ([CAMPOS, 2018](#)). A implementação foi feita como apresentado no

Algoritmo 4:

Algoritmo 4: Simulated Annealing

Entrada: $f(\cdot)$, $N(\cdot)$, α , $SAmax$, T_0 , s

Saida: s

inicio

$s^* \leftarrow s$;

$IterT \leftarrow 0$;

$T \leftarrow T_0$;

enquanto ($elapsedTime < timeLimit$) **fazer**

enquanto ($IterT < SAmax$) **fazer**

$IterT \leftarrow IterT + 1$;

 Gere um vizinho qualquer $s' \in N(s)$;

$\Delta = f(s') - f(s)$;

se ($\Delta < 0$) **então**

$s \leftarrow s'$;

se ($f(s') < f(s^*)$) **então**

$s^* \leftarrow s'$;

fim

fim

senão

 Tome $x \in [0, 1]$;

se ($x < e^{\frac{-\Delta}{T}}$) **então**

$s \leftarrow s'$;

fim

fim

fim

$T \leftarrow \alpha \times T$;

$IterT \leftarrow 0$;

fim

$s \leftarrow s^*$;

retorna s ;

fim

4.3.2.1 Vizinhaça

A meta-heurística [SA](#) busca soluções melhores entre os vizinhos da solução inicial. Portanto, é importante entender como os vizinhos são definidos nesse trabalho. A operação principal, nesse caso, é trocar uma cor da solução, ou seja, um vizinho de uma solução seria outra solução, igualmente viável, porém com um vértice com cor diferente. A ideia é refinar cada vez mais a solução. Dessa forma, os vizinhos são encontrados através do

Algoritmo 5 descrito abaixo:

Algoritmo 5: Vizinho

Entrada: s

Saida: s'

inicio

$cores \leftarrow$ lista contendo as cores utilizadas em s ;

$u \leftarrow$ vértice aleatório de V ;

$c \leftarrow$ cor aleatória de cores, exceto a última;

$s' \leftarrow s$;

 Se possível atribua a cor c ao vértice u em s' ;

retorna s'

fim

4.3.2.2 Função Objetivo

A função objetivo é uma forma de adicionar um custo a uma solução com o objetivo de compará-la com as demais encontradas e, assim, encontrar a melhor solução entre as disponíveis. Nas heurísticas a função objetivo é parte principal, uma vez que algumas heurísticas tem como objetivo minimizar ou maximizar o custo de uma solução. A heurística SA busca minimizar o custo da solução, ou seja, ele compara as soluções possíveis e verifica qual delas possui uma função objetivo menor em comparação com a outra, e então aceita como solução a de menor custo. Nesse trabalho, foram implementadas duas formas para o cálculo da função objetivo. Na forma mais simples, a função objetivo é igual ao número de cores que a solução passada possui, ou seja, uma solução com menos cores é melhor que uma solução com mais cores. Na segunda forma, o cálculo da função objetivo se torna um pouco mais complexa, uma vez que leva em consideração a forma com que o grafo foi colorido, o número de vértices e, também número de cores utilizadas apresentada na Equação 4.7:

$$custo = numVertices * numCoresSolucao + min \quad (4.7)$$

onde o $numVertices$ é o número de vértices que o grafo possui, $numCoresSolucao$ é o número de cores utilizadas na solução e min é o número de vezes em que a cor que foi menos atribuída foi utilizada.

Para exemplificar, considere duas colorações s_1 e s_2 para um mesmo grafo qualquer e considere que essas soluções possuem o mesmo tamanho, ou seja, o mesmo número de cores. Considere também que na solução s_1 a cor com a menor utilização é a cor c_1 e ela é atribuída a apenas um vértice. Já na solução s_2 a cor com utilizada menos vezes é a cor c_2 e essa cor é atribuída a cinco vértices. Nesse caso, a função objetivo de s_1 seria menor que a função objetivo de s_2 e, portanto, s_1 é considerada uma solução melhor em comparação com s_2 . Isso acontece pois, no desenvolvimento desse trabalho, foi considerado que uma coloração onde uma das cores é utilizada poucas vezes tem mais chances de ser otimizada,

uma vez que remover essa cor é mais fácil, e assim obter uma nova coloração com um menor número de cores. A implementação foi feita como apresentado no Algoritmo 6:

Algoritmo 6: Função Objetivo

Entrada: $graph, s, flag$

Saida: $funcaoObj$

inicio

$funcaoObj \leftarrow 0;$

se $flag = 1$ **então**

$funcaoObj \leftarrow$ número de cores utilizadas em s ;

fim

senão

$\alpha \leftarrow$ número de vértices do grafo;

$cores \leftarrow$ cores utilizadas em s ;

$numCores \leftarrow$ número de cores utilizadas em s ;

$i \leftarrow 0;$

$m \leftarrow 0;$

enquanto $(!cores.isEmpty())$ **fazer**

$cor \leftarrow cores[i];$

para $j = 1$ **até** tamanho da solução s **fazer**

se $cor = s[j]$ **então**

$aux \leftarrow aux + 1;$

fim

fim

$quantUsos[m] \leftarrow aux;$

 remover i de $cores$;

$m \leftarrow m + 1;$

$aux \leftarrow 0;$

fim

$min \leftarrow \infty;$

para $k = 1$ **até** tamanho de $quantUsos$ **fazer**

se $quantUsos[k] < min$ **então**

$min \leftarrow quantUsos[k];$

fim

fim

$funcaoObj = \alpha * numCores + min;$

fim

retorna $funcaoObj$;

fim

A forma para o cálculo da função objetivo é um parâmetro passado junto a solução para a qual se deseja obter o custo. Dessa forma, ele pode ser alterado de acordo com as

necessidades do programador. Nesse trabalho a função objetivo utilizada [SA](#) foi a segunda forma, buscando uma maior otimização na solução final encontrada pelo algoritmo.

4.3.3 Heurística Matemática

A heurística matemática desenvolvida nesse trabalho tem o objetivo de apresentar melhores soluções para o [PCG](#). Para isso, a abordagem utilizada consiste em decompor o problema original, e fazer a otimização do modelo com o subproblema. Para isso, uma solução inicial e o problema modelados são carregados e, durante um tempo previamente determinado, a heurística torna fixa parte das cores presentes na solução inicial, atualiza o modelo e faz a otimização usando as cores não fixadas. Dessa forma, cria-se um subproblema onde as cores fixadas não podem ter seus vértices alterados e as cores não fixadas podem ser mudadas de vértices com o objetivo de diminuir o número de cores necessárias para colorir o grafo. A porcentagem de cores a serem fixadas é previamente definida e as cores a serem fixadas são escolhidas randomicamente. A cada iteração a nova solução é totalmente fixada e novas cores sorteadas para dar novo início ao processo.

A *mipHeuristic* foi implementada como apresentado no Algoritmo 7:

Algoritmo 7: Heurística *mipHeuristic*

Entrada: *numColors, time, timeIter*

início

Carregar Modelo

Carregar Solução Inicial

Fixar Variáveis

Definir um limite de tempo para o solver passando timeIter

$t0 \leftarrow$ *tempo atual do sistema em milissegundos*

enquanto $((t0 + time \times 1000) > \text{tempo atual do sistema em milissegundos})$

fazer

$randColors \leftarrow$ *cores escolhidas randomicamente*

Desafixar randColors

Atualizar modelo

Otimizar modelo

$numColors \leftarrow$ *numero de cores ajustado*

Fixar todas as variáveis

fim

fim

5 Experimentos Computacionais

5.1 Ambiente Computacional

A máquina utilizada para o desenvolvimento desse trabalho foi um *Notebook PC HP ENVY 17* com arquitetura x64, ou seja, 64 bits. Seu processador é o *Intel(R) Core(TM) i7-4710HQ CPU @ 2.50GHZ*, 2501 Mhz, 4 *cores* e 8 processadores lógicos. Sua memória RAM possui 12GB. O sistema operacional utilizado é o *Microsoft Windows 10 Home*. O Ambiente de Desenvolvimento Integrado ([IDE](#)) utilizado para implementação dos algoritmos e realização dos testes foi o *NetBeans IDE 8.2*. Todos os algoritmos foram implementados na linguagem de programação *Java*. Para resolver o modelo matemático e a heurística matemática implementada, foi utilizado o *Gurobi Optimizer* que é um resolvidor voltado para programações matemáticas ([OPTIMIZATION, 2017](#)).

5.2 Caracterização das Instâncias

Para avaliar os algoritmos propostos, foram utilizadas instâncias já conhecidas na literatura, do *DIMACS Implementation Challenge* ([DIMACS, 1995](#)). A [Tabela 1](#) e a [Tabela 2](#) caracterizam esses grafos, onde $|V|$ é o número de vértices, $|E|$ é o número de arestas e ρ é a densidade do grafo. Os grafos da [Tabela 2](#) são os considerados grafos difíceis, pois são maiores e mais desafiadores.

Tabela 1 – Caracterização das instâncias (Grupo 1)

Grafo	V	E	ρ
anna	138	493	0,0522
david	87	406	0,1085
huck	74	301	0,1114
2-Insertions_3	37	72	0,1081
3-Insertions_3	56	110	0,0714
jean	80	254	0,0804
queen5_5	25	160	0,5333
queen6_6	36	290	0,4603
queen7_7	49	476	0,4048
queen8_8	64	728	0,3611
queen9_9	81	1056	0,3259
miles250	128	387	0,0476
miles500	128	1170	0,1439
games120	120	638	0,0894
mug88-1	88	146	0,0381
mug88-25	88	146	0,0381
mug100_1	100	166	0,0335
mug100_25	100	166	0,0335
myciel3	11	20	0,3636
myciel4	23	71	0,2806
myciel5	47	236	0,2183
myciel6	95	755	0,1691
myciel7	191	2360	0,1301
DSJC125.1	125	736	0,0950
DSJC125.5	125	3891	0,5021
DSJC125.9	125	6961	0,8982
DSJC250.1	250	3218	0,1034
DSJC250.9	250	27897	0,8963
fpsol2.i.1	496	11654	0,0949
fpsol2.i.2	451	8691	0,0856
fpsol2.i.3	425	8688	0,0964
zéroin.i.1	211	4110	0,1855
zéroin.i.2	211	3541	0,1598
zéroin.i.3	206	3540	0,1677
mulsol.i.1	197	3925	0,2033
mulsol.i.2	188	3885	0,2210
mulsol.i.3	184	3916	0,2326
mulsol.i.4	185	3946	0,2318
mulsol.i.5	186	3973	0,2309

Tabela 2 – Caracterização das instâncias (Grupo 2)

Grafo	V	E	ρ
DSJC250.5	250	15668	0,5034
DSJC500.1	500	12458	0,0999
DSJC500.5	500	62624	0,5020
DSJC1000.1	1000	49629	0,0994
DSJR500.1c	500	121275	0,9721
DSJR500.5	500	58862	0,4718
le450_15c	450	16680	0,1651
le450_15d	450	16750	0,1658
le450_25c	450	17343	0,1717
le450_25d	450	17425	0,1725

5.3 Planejamento Experimental

Existem alguns parâmetros a serem definidos e passados para os algoritmos antes de sua execução. Nesse trabalho, todas as heurísticas de refinamento necessitam da definição de um tempo de execução. Devido a falta de padronização na literatura sobre esse parâmetro, os tempos utilizados foram pensados de acordo com as características dos grafos testados com o objetivo de obter o melhor desempenho. Assim, foi definido para o grupo 1 que as heurísticas de refinamento seriam executadas durante 1000 segundos e, para o grupo 2, 100 segundos.

A única definição necessária para a heurística *recol* é esse tempo de execução. A meta-heurística *Simulated Annealing* necessita outros parâmetros, sendo eles, a taxa de resfriamento, o número máximo de iterações por temperatura, e a temperatura inicial. Já a heurística matemática *mipHeuristic* necessita, além do tempo de execução, da porcentagem cores da solução inicial que devem ser desafixadas.

Para definir esses parâmetros de forma a obter o melhor desempenho dos algoritmos, foram feitos alguns experimentos. Todos eles foram feitos seguindo a mesma metodologia e com os mesmos grafos, a fim de garantir que os melhores resultados. Os grafos utilizados para esses experimentos foram todos os do grupo 1, e alguns do grupo 2 os quais sua coloração ótima não havia sido encontrada já na solução inicial.

5.3.1 Definição dos parâmetros para a meta-heurística *Simulated Annealing*

A definição dos parâmetros foi feita com base em fatores e níveis. A fim de determinar parâmetros adequados ao problema e que levariam a melhores soluções foram feitos alguns testes. Os fatores utilizados e os níveis para cada um foram definidos conforme a indicado na [Tabela 3](#):

Tabela 3 – Fatores e Níveis - SA

Fatores	Níveis
α	{0,1; 0,4; 0,8}
T_0	{5; 10; 50}
SAmax	{50; 100; 500}

Dessa forma, a combinação desses valores resultou em 27 experimentos mostrados na [Tabela 4](#), onde é possível ver os valores testados para os parâmetros. A tabela também mostra o valor do *GAP* médio de cada experimento.

O *GAP* médio foi a forma de classificar quais parâmetros obtiveram mais sucesso e, dessa foram, serão os parâmetros utilizados. Esse valor é calculado considerando o tamanho da coloração obtida para cada grafo durante os experimentos e o tamanho da coloração ótima ou a melhor coloração encontrada dos respectivos grafos. Para obter o *GAP*, esses valores são relacionados conforme a equação [Equação 5.1](#):

$$GAP = \frac{s' - s^*}{s'} \quad (5.1)$$

onde s' é o tamanho da solução obtida no experimento e s^* é o tamanho da solução ótima, ou da melhor solução encontrada. Logo após, foi feita a média aritmética dos valores obtidos para cada grafo em cada experimento.

Tabela 4 – Experimentos - SA

Experimento	alpha	T_0	SAmix	GAP Médio
exp1	0,1	5	50	0,1491
exp2	0,1	5	100	0,1559
exp3	0,1	5	500	0,1499
exp4	0,1	10	50	0,1530
exp5	0,1	10	100	0,1518
exp6	0,1	10	500	0,1508
exp7	0,1	50	50	0,1553
exp8	0,1	50	100	0,1573
exp9	0,1	50	500	0,1492
exp10	0,4	5	50	0,1607
exp11	0,4	5	100	0,1553
exp12	0,4	5	500	0,1654
exp13	0,4	10	50	0,1508
exp14	0,4	10	100	0,1564
exp15	0,4	10	500	0,1611
exp16	0,4	50	50	0,1459
exp17	0,4	50	100	0,1514
exp18	0,4	50	500	0,1489
exp19	0,8	5	50	0,1581
exp20	0,8	5	100	0,1549
exp21	0,8	5	500	0,1567
exp22	0,8	10	50	0,1527
exp23	0,8	10	100	0,1488
exp24	0,8	10	500	0,1507
exp25	0,8	50	50	0,1522
exp26	0,8	50	100	0,1564
exp27	0,8	50	500	0,1534

É possível perceber que o menor valor de *GAP* médio obtido foi de 0.1459. Assim, utilizando o critério acima descrito, os parâmetros selecionados foram o do exp16, sendo esses $\alpha = 0,4$, $T_0 = 50$ e $SAmix = 50$.

5.3.2 Definição dos parâmetros para a heurística matemática *mipHeuristic*

A mesma metodologia utilizada na escolha de parâmetros para a meta-heurística SA foi utilizada na escolha do parâmetro da *mipHeuristic*. Para essa heurística, o único parâmetro a ser definido é a porcentagem de cores da solução inicial a ser desafixada, *color_percent*. A Tabela 5 mostra os níveis testados:

Tabela 5 – Fator e Níveis - *mipHeuristic*

Fator	Níveis
<i>color_percent</i>	{0,2; 0,5; 0,8}

Sendo assim, foram feitos 3 experimentos, como apresentados na [Tabela 6](#). Essa Tabela também apresenta os valores testados em cada experimento e o *GAP* médio obtido, assim como nos experimentos feitos para a meta-heurística [SA](#).

Tabela 6 – Experimentos *mipHeuristic*

Experimento	<i>color_percent</i>	<i>GAP</i> Médio
exp1	0,2	0,1370
exp2	0,5	0,1486
exp3	0,8	0,1428

O menor valor encontrado para o *GAP* Médio foi de 0.1370, e, sendo assim, o valor de parâmetro escolhido foi o do exp1, e *color_percent* = 0.2.

5.4 Resultados Obtidos

Durante a realização desse trabalho foram feitos diversos testes com o objetivo de que esses resultados juntos levassem ao melhor resultado possível para as instâncias caracterizadas acima. A heurística matemática proposta nesse trabalho recebe como parâmetro uma solução inicial, uma vez que ela é uma heurística de refinamento. Portanto, o primeiro passo é definir uma boa solução inicial, levando em consideração que quanto melhor essa solução mais próximo está da solução ótima. Com esse intuito, foram realizados testes comparando os dois algoritmos implementados para geração da solução inicial, o algoritmo guloso e o *D-Satur*, e, a partir desses resultados, definiu-se o melhor algoritmo a ser utilizado na geração da solução. O segundo passo foi passar a solução para as heurísticas de refinamento e, em seguida, fazer os testes para comparar os resultados dos três algoritmos de refinamento implementados, sendo eles, a heurística *recol*, a meta-heurística *Simulated Annealing* e a heurística matemática proposta neste trabalho, a *mipHeuristic*.

5.4.1 Geração da Solução Inicial

A [Tabela 7](#) e a [Tabela 8](#) apresentam uma comparação da solução inicial obtida utilizando os algoritmo guloso e o algoritmo *D-Satur*. Também apresentam o tempo gasto por cada um dos algoritmos em milissegundos.

Tabela 7 – Geração da Solução Inicial (Grupo 1)

Grafo	Algoritmo Guloso	$T_{\text{Algoritmo Guloso}}$ (ms)	$D\text{-Satur}$	$T_{D\text{-Satur}}$ (ms)
anna	12	20	11	74
david	12	13	11	56
huck	11	10	11	55
2-Insertions_3	4	2	4	36
3-Insertions_3	4	3	4	51
jean	10	10	10	56
queen5_5	8	0	5	28
queen6_6	11	6	9	54
queen7_7	10	10	11	56
queen8_8	13	24	12	61
queen9_9	16	17	13	67
miles250	9	4	8	57
miles500	22	24	20	75
games120	9	19	9	63
mug88_1	4	14	4	37
mug88_25	4	24	4	56
mug100_1	4	9	4	52
mug100_25	4	9	4	55
myciel3	4	2	4	33
myciel4	5	1	5	48
myciel5	6	10	6	50
myciel6	7	29	7	71
myciel7	8	20	8	76
DSJC125.1	8	28	6	65
DSJC125.5	26	36	22	87
DSJC125.9	56	42	51	121
DSJC250.1	13	79	10	85
DSJC250.9	99	149	92	336
fpsol2.i.1	65	169	65	219
fpsol2.i.2	30	141	30	208
fpsol2.i.3	30	121	30	167
zeroin.i.1	49	65	49	112
zeroin.i.2	30	56	30	81
zeroin.i.3	30	57	30	95
multsol.i.1	49	78	49	110
multsol.i.2	31	64	31	82
multsol.i.3	31	58	31	112
multsol.i.4	31	56	31	97
multsol.i.5	31	59	31	98

Analisando os resultados obtidos, é possível perceber que o $D\text{-Satur}$ apresenta um melhor desempenho em relação ao algoritmo guloso para a maioria das instâncias do primeiro grupo e para todas as instâncias do segundo grupo. Dessa maneira, conclui-se que, dentre as heurísticas construtivas avaliadas e considerando as instâncias utilizadas nesse

Tabela 8 – Geração da Solução Inicial (Grupo 2)

Grafo	Algoritmo Guloso	$T_{\text{Algoritmo Guloso}}$ (ms)	$D\text{-Satur}$	$T_{D\text{-Satur}}$ (ms)
DSJC250.5	43	107	37	173
DSJC500.1	20	139	16	190
DSJC500.5	72	571	65	897
DSJC1000.1	31	810	27	894
DSJR500.1c	109	923	90	1463
DSJR500.5	143	515	130	894
le450_15c	30	163	23	289
le450_15d	31	168	24	223
le450_25c	37	162	29	240
le450_25d	35	167	28	245

trabalho, o $D\text{-Satur}$ é o melhor algoritmo para geração da solução inicial a ser utilizada nas heurísticas de refinamento.

5.4.2 Heurística Matemática - *mipHeuristic*

Os resultados obtidos pela heurística matemática proposta estão apresentados abaixo nas [Tabela 9](#) e [Tabela 10](#). A coluna s^* mostra o melhor número cromático conhecido na literatura para cada instância. Valores marcados com * representam soluções ótimas¹. As colunas *Repli1*, *Repli2*, *Repli3*, *Repli4*, *Repli5* mostram os resultados das cinco replicações realizadas, s_{min} a coloração mínima obtida, s_{max} coloração máxima obtida, e o *GAP* da menor coloração obtida nas replicações. É importante notar que o cálculo do *GAP* apresentado foi feito de acordo com a [Equação 5.1](#). As instâncias do grupo 1 em que a coloração ótima foi encontrada já na solução inicial foram ocultadas para melhor visualização dos resultados obtidos pela *mipHeuristic*.

Tabela 9 – Resultados *mipHeuristic* (Grupo 1)

Grafo	s^*	Repli1	Repli2	Repli3	Repli4	Repli5	s_{min}	s_{max}	$s_{média}$	<i>GAP</i>
queen6_6	7*	7	7	7	7	7	7	7	7,00	0,00
queen7_7	7*	7	7	7	7	7	7	7	7,00	0,00
queen8_8	9*	10	9	9	9	10	9	10	9,40	0,00
queen9_9	10*	11	11	11	11	10	10	11	10,80	0,00
DSJC125.1	5*	5	5	5	5	5	5	5	5,00	0,00
DSJC125.5	17*	20	21	21	21	20	20	21	20,60	0,15
DSJC125.9	44*	48	47	49	49	47	47	49	48,00	0,06
DSJC250.1	8*	9	9	9	10	9	9	10	9,20	0,11
DSJC250.9	72*	91	92	92	90	92	90	92	91,40	0,20

¹ Valores obtidos em [Douiri e Elberoussi \(2015\)](#)

Tabela 10 – Resultados *mipHeuristic* (Grupo 2)

Grafo	s^*	Repli1	Repli2	Repli3	Repli4	Repli5	s_{min}	s_{max}	$s_{média}$	GAP
DSJC250.5	28	37	36	36	36	36	36	37	36,20	0,22
DSJC500.1	12	15	15	15	15	15	15	15	15,00	0,20
DSJC500.5	48	64	64	64	64	64	64	64	64,00	0,25
DSJC1000.1	20	26	26	26	26	26	26	26	26,00	0,23
DSJR500.1c	85	89	90	90	90	89	89	90	89,60	0,04
DSJR500.5	122	128	130	127	129	127	127	130	128,20	0,04
le450_15c	15	23	23	23	23	23	23	23	23,00	0,35
le450_15d	15	23	23	23	23	23	23	23	23,00	0,35
le450_25c	25	28	28	28	28	28	28	28	28,00	0,11
le450_25d	25	28	28	28	28	28	28	28	28,00	0,11

5.5 Análise de Resultados

Para fazer a análise dos resultados obtidos pela heurística matemática *mipHeuristic*, a ideia é comparar os resultados acima apresentados com os resultados obtidos pelas outras heurísticas de refinamento já consagradas na literatura e que também foram implementadas. Uma outra forma utilizada para avaliar os resultados foi a comparação com o trabalho dos autores [Douiri e Elbernoussi \(2015\)](#) intitulado *Solving the Graph Coloring Problem via Hybrid Genetic Algorithms*, onde ele apresenta os resultados obtidos com o algoritmo proposto pelos mesmos.

5.5.1 Comparação das Heurísticas de Refinamento

A [Tabela 11](#) e a [Tabela 12](#) mostram os resultados das comparações entre as heurísticas. Com o objetivo de fazer uma comparação justa entre elas, e como todas utilizam valores aleatórios, todos os testes seguiram a mesma metodologia e foram feitos como o apresentado para a *mipHeuristic*. Os resultados individuais para a heurística *recol* e a meta-heurística *Simulated Annealing* contendo a melhor coloração, a pior coloração, a média dos valores e o GAP encontra-se nos [Apêndice A](#) e [Apêndice B](#), e abaixo encontram-se os melhores resultados obtidos nos testes. Como feito anteriormente, as instâncias comparadas são aquelas em que a solução ótima não foi encontrada já na solução inicial pois, dessa forma, é possível comparar os resultados obtidos por cada algoritmo e o GAP novamente foi calculado como apresentado na [Equação 5.1](#).

Tabela 11 – Heurísticas de Refinamento (Grupo 1)

Grafo	s^*	$recol$	GAP_{recol}	SA	GAP_{SA}	$mipHeuristic$	$GAP_{mipHeuristic}$
queen6_6	7*	8	0,13	8	0,13	7	0,00
queen7_7	7*	9	0,22	9	0,22	7	0,00
queen8_8	9*	10	0,10	10	0,10	9	0,00
queen9_9	10*	11	0,09	11	0,09	10	0,00
DSJC125.1	5*	6	0,17	6	0,17	5	0,00
DSJC125.5	17*	18	0,06	19	0,11	20	0,15
DSJC125.9	44*	45	0,02	45	0,02	47	0,06
DSJC250.1	8*	9	0,11	9	0,11	9	0,11
DSJC250.9	72*	75	0,04	85	0,15	90	0,20

Os resultados obtidos para as instâncias do grupo 1 mostram que a *mipHeuristic* obteve a coloração ótima para a cinco das instâncias utilizadas nesse trabalho, sendo elas, **queen6_6**, **queen7_7**, **queen8_8**, **queen9_9** e **DSJC125.1**. As demais instâncias não tiveram sua coloração ótima encontrada e, em relação a elas, é possível perceber que os três algoritmos apresentaram soluções semelhantes para a **DSJC250.1**, e para as outras a *recol* e a *Simulated Annealing* apresentaram melhores soluções. Entretanto, as heurísticas *recol* e *Simulated Annealing* não encontraram a coloração ótima para nenhuma das instâncias do grupo 1. Dessa forma, considerando as instâncias utilizadas nesse trabalho e neste grupo e o contexto experimental estabelecido, a heurística matemática proposta, *mipheuristic*, apresentou um melhor desempenho em relação aos demais algoritmos avaliados, uma vez que apresentou a coloração ótima ou a melhor coloração encontrada em 66,67% das instâncias desse grupo.

Tabela 12 – Heurísticas de Refinamento (Grupo 2)

Grafo	s^*	$recol$	GAP_{recol}	SA	GAP_{SA}	$mipHeuristic$	$GAP_{mipHeuristic}$
DSJC250.5	28	31	0,10	31	0,10	36	0,22
DSJC500.1	12	14	0,14	14	0,14	15	0,20
DSJC500.5	48	54	0,11	60	0,20	64	0,25
DSJC1000.1	20	22	0,09	24	0,17	26	0,23
DSJR500.1c	85	85	0,00	89	0,04	89	0,04
DSJR500.5	122	128	0,05	129	0,05	127	0,04
le450_15c	15	20	0,25	23	0,35	23	0,35
le450_15d	15	21	0,29	23	0,35	23	0,35
le450_25c	25	27	0,07	28	0,11	28	0,11
le450_25d	25	27	0,07	27	0,07	28	0,11

Para o grupo 2, o qual possui os grafos mais desafiadores, a *mipHeuristic* apresenta um resultado de destaque, sendo esse, a melhor coloração obtida para a instância **DSJR500.5**. Também apresenta quatro colorações semelhantes a meta-heurística SA para

os grafos **DSJR500.1c** , **le450_15c**, **le450_15d**, **le450_25c**. Considerando as instâncias desse grupo os resultados sugerem que em 50% dos grafos testados, a *mipHeuristic* mostra resultados competitivos, que, no contexto desse trabalho, podem ser considerados bons resultados.

5.5.2 Comparação com a Literatura

A [Tabela 13](#) e a [Tabela 14](#) mostram uma comparação entre os valores encontrados na literatura e a heurística matemática proposta nesse trabalho, *mipHeuristic*. A coluna s^* mostra o melhor número cromático conhecido na literatura para cada instância. Valores marcados com * representam soluções ótimas², a coluna $s_{DourirElbernoussi}$ mostra a coloração obtida pelo algoritmo proposto pelos autores [Douiri e Elbernoussi \(2015\)](#), a coluna $s_{mipHeuristic}$ apresenta a coloração obtida pela *mipHeuristic* e a coluna GAP mostra o GAP calculado através da [Equação 5.1](#) entre a coloração obtida pela *mipHeuristic* e a coloração ótima s^* .

Tabela 13 – Comparação com a Literatura (Grupo 1)

Grafo	s^*	$s_{DourirElbernoussi}$	GAP_{DeE}	$s_{mipHeuristic}$	$GAP_{mipHeuristic}$
queen6_6	7*	7	0,00	7	0,00
queen7_7	7*	7	0,00	7	0,00
queen8_8	9*	9	0,00	9	0,00
queen9_9	10*	10	0,00	10	0,00
DSJC125.1	5*	6	0,17	5	0,00
DSJC125.5	17*	17	0,00	20	0,15
DSJC125.9	44*	44	0,00	47	0,06
DSJC250.1	8*	8	0,00	9	0,11
DSJC250.9	72*	72	0,00	90	0,20

Tabela 14 – Comparação com a Literatura (Grupo 2)

Grafo	s^*	$s_{DourirElbernoussi}$	GAP_{DeE}	$s_{mipHeuristic}$	$GAP_{mipHeuristic}$
DSJC250.5	28	28	0,00	36	0,22
DSJC500.1	12	12	0,00	15	0,20
DSJC500.5	48	48	0,00	64	0,25
DSJC1000.1	20	20	0,00	26	0,23
DSJR500.1c	85	85	0,00	89	0,04
DSJR500.5	122	124	0,02	127	0,04
le450_15c	15	15	0,00	23	0,35
le450_15d	15	15	0,00	23	0,35
le450_25c	25	25	0,00	28	0,11
le450_25d	25	25	0,00	28	0,11

² Valores obtidos em [Douiri e Elbernoussi \(2015\)](#)

Para comparar os resultados é importante entender como esses resultados foram obtidos. Douiri e Elbernoussi (2015) apresentaram um novo algoritmo genético híbrido baseado em uma heurística local chamada DBG (DOURI; ELBERNOUSSI, 2011). Já a *mipHeuristic* é uma heurística matemática de refinamento baseada em decomposição que utiliza uma solução inicial produzida pela heurística construtiva *D-Satur*.

Analizando os resultados obtidos pela *mipHeuristic* e os apresentados por Douiri e Elbernoussi (2015) é possível perceber que, para o grupo 1, o algoritmo proposto por estes autores encontra a coloração ótima para oito das nove instâncias utilizadas nesse trabalho, enquanto a *mipHeuristic* encontra a coloração ótima para uma das nove instâncias. Para o grupo 2, o algoritmo genético híbrido encontra a melhor coloração conhecida para nove das dez instâncias. Entretanto, é possível perceber também, que, no grupo 1, para o único grafo em que o algoritmo genético híbrido não encontra a coloração ótima, a *mipHeuristic* encontra essa solução (**DSJC125.1**).

5.5.3 Discussão de Resultados

O problema da coloração de grafos é um problema de muita importância na comunidade acadêmica. Sendo um problema NP-difícil, não existe uma solução exata em tempo polinomial para resolver esse problema. Dessa forma, muitos esforços são feitos com o objetivo de encontrar algoritmos capazes de fornecer as melhores soluções possíveis. A heurística matemática *recol* apresentada nesse trabalho, tem o objetivo de funcionar como uma heurística de refinamento, ou seja, melhorar as soluções iniciais obtidas por meio de heurísticas construtivas. Com os resultados apresentados, considerando as instâncias utilizadas nesse trabalho e o contexto em que os experimentos foram feitos, nota-se que a heurística matemática *recol* apresenta uma melhora para as colorações iniciais geradas pelo algoritmo *D-Satur*, que já produz bons resultados. Dessa forma, o refinamento feito pela heurística matemática proposta permite alcançar resultados ainda melhores.

É possível perceber que, considerando a forma em que esse trabalho foi desenvolvido, as instâncias utilizadas nos testes, os tempos pré-definidos e os demais fatores específicos, que a heurística apresenta melhores resultados para os grafos menores do grupo 1. Isso não quer dizer que a heurística não possa apresentar bom desempenho para grafos maiores, uma vez que parâmetros podem ser redefinidos e também podem ocorrer modificações na forma de decisão sobre as cores a serem desafixadas. Sendo assim, apesar da comparação com o estado da arte não ter obtido melhores resultados que os já obtidos por outros autores, é importante destacar que isso não exclui a utilização da heurística matemática proposta nesse artigo, apenas mostra que muitos estudos ainda podem ser feitos nessa área.

6 Conclusão

Este trabalho popôs a utilização de uma heurística matemática baseada em decomposição aplicada ao Problema da Coloração de Grafos. Sendo o Problema da Coloração de Grafos de elevada relevância na comunidade científica nas áreas de computação e matemática, foi fundamental entender sua importância, aplicação e parte dos estudos já feitos com o objetivo de apresentar melhores soluções para esse problema. Dessa forma, foi feito um levantamento bibliográfico dos trabalhos mais relevantes relacionados e algoritmos importantes foram implementados. Também foi feita a programação matemática do problema e a implementação da heurística proposta. Para avaliar os resultados obtidos foram feitas comparações com outras heurísticas de refinamento e com a literatura. Foi possível perceber que, apesar dos primeiros relatos da aplicação dessa estratégia serem recentes, e ainda existir muitos estudos a serem feitos, essa abordagem apresenta resultados preliminares promissores, que podem ser aprimorados em trabalhos futuros.

Como trabalhos futuros, uma proposta interessante para a *mipHeuristic*, seria utilizá-la para refinar, ainda mais, os resultados de outras heurísticas de refinamento, de forma que a saída de uma das heurísticas de refinamento fosse a entrada da heurística matemática. Outro ponto, seria explorar outras estratégias de decompor o problema na heurística matemática.

Analisando todos os aspectos apresentados, pode-se concluir que a utilização de uma heurística matemática baseada em decomposição para o Problema da Coloração de Grafos apresenta resultados satisfatórios, e que, dada a importância do problema estudado, possui grandes possibilidades de crescimento dentro da área.

Referências

- AL-OMARI, H.; SABRI, K. E. New graph coloring algorithms. *American Journal of Mathematics and Statistics*, v. 2, n. 4, p. 739–741, 2006. Citado na página 26.
- ALVES, R.; DELGADO, C. Programação linear inteira. 1997. Citado na página 23.
- ARROYO, J. E. C. et al. Heurísticas e metaheurísticas para otimização combinatória multiobjetivo. [sn], 2002. Citado na página 23.
- BALL, M.; MAGAZINE, M. The design and analysis of heuristics. *Networks*, Wiley Online Library, v. 11, n. 2, p. 215–219, 1981. Citado na página 25.
- BARNIER, N.; BRISSET, P. Graph coloring for air traffic flow management. *Annals of operations research*, Springer, v. 130, n. 1-4, p. 163–178, 2004. Citado na página 20.
- BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM computing surveys (CSUR)*, ACM, v. 35, n. 3, p. 268–308, 2003. Citado na página 24.
- BRÉLAZ, D. New methods to color the vertices of a graph. *Communications of the ACM*, ACM, v. 22, n. 4, p. 251–256, 1979. Citado 2 vezes nas páginas 19 e 31.
- BUENO, F. Métodos heurísticos. *Teoria e implementações. Araranguá: IFSC*, 2009. Citado 3 vezes nas páginas 23, 32 e 33.
- CAMPOS, V. C. d. S. Simulated annealing. *Notas de Aula, Universidade Federal de Ouro Preto*, 2018. Citado na página 34.
- CHAITIN, G. J. Register allocation & spilling via graph coloring. In: ACM. *ACM Sigplan Notices*. [S.l.], 1982. v. 17, n. 6, p. 98–105. Citado na página 20.
- CHARTRAND, G.; ZHANG, P. *Chromatic graph theory*. [S.l.]: CRC press, 2008. Citado na página 22.
- CHOW, F. C.; HENNESSY, J. L. The priority-based coloring approach to register allocation. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, ACM, v. 12, n. 4, p. 501–536, 1990. Citado na página 20.
- COLIN, E. C. *Pesquisa Operacional: 170 aplicações em estratégia, finanças, logística, produção, marketing e vendas*. [S.l.]: Livros Técnicos e Científicos, 2007. Citado na página 32.
- CORDENONSI, A. Z. Ambientes, objetos e dialogicidade: uma estratégia de ensino superior em heurísticas e metaheurísticas. 2008. Citado na página 23.
- DALLAQUA, C. *Origem da Teoria dos Grafos – As 7 Pontes de Königsberg*. 2016. Disponível em: <<https://universoracionalista.org/origem-da-teoria-dos-grafos-as-7-pontes-de-konigsberg/>>. Citado na página 18.

- DIMACS. *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*. 1995. Disponível em: <<http://dimacs.rutgers.edu/Challenges/>>. Citado 2 vezes nas páginas 20 e 39.
- DOUIRI, S. M.; ELBERNOUSSI, S. A new heuristic for the sum coloring problem. *Applied Mathematical Sciences*, v. 5, n. 63, p. 3121–3129, 2011. Citado na página 50.
- DOUIRI, S. M.; ELBERNOUSSI, S. Solving the graph coloring problem via hybrid genetic algorithms. *Journal of King Saud University-Engineering Sciences*, Elsevier, v. 27, n. 1, p. 114–118, 2015. Citado 7 vezes nas páginas 27, 46, 47, 49, 50, 56 e 57.
- DREXL, A.; KNUST, S. Sports league scheduling: Graph-and resource-based models. *Omega*, Elsevier, v. 35, n. 5, p. 465–471, 2007. Citado na página 20.
- DU, T.-C. December 2, 2013. 2013. Citado na página 31.
- FONSECA, G. H.; SANTOS, H. G.; CARRANO, E. G. Integrating matheuristics and metaheuristics for timetabling. *Computers & Operations Research*, Elsevier, v. 74, p. 108–117, 2016. Citado na página 25.
- FREITAS, F. G. de et al. Otimização em teste de software com aplicação de metaheurísticas. *Revista de Sistemas de Informação da FSMA nº 5, pag*, p. 3–13, 2010. Citado na página 24.
- GAREY, M. R.; JOHNSON, D. S. *Computers and intractability*. [S.l.]: wh freeman New York, 2002. v. 29. Citado na página 20.
- GLOVER, F. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, Elsevier, v. 13, n. 5, p. 533–549, 1986. Citado na página 24.
- GLOVER, F. *Kochenberger, Gary-Handbook of Metaheuristics (International Series in Operations Research & Management Science)*. [S.l.]: Springer, 2003. Citado na página 24.
- JENSEN, T. R.; TOFT, B. *Graph coloring problems*. [S.l.]: John Wiley & Sons, 2011. v. 39. Citado na página 18.
- KIRKPATRICK, S.; JR, C. G.; VECCHI, M. Optimization by simulated annealing. *SCIENCE*, v. 220, n. 4598, 1983. Citado 2 vezes nas páginas 19 e 33.
- KOUIDER, A. et al. Mixed integer linear programs and tabu search approach to solve mixed graph coloring for unit-time job shop scheduling. In: IEEE. *Automation Science and Engineering (CASE), 2015 IEEE International Conference on*. [S.l.], 2015. p. 1177–1181. Citado na página 20.
- LIMA, M. P. et al. Using evolutionary algorithms for channel assignment in 802.11 networks. In: IEEE. *Computational Intelligence for Communication Systems and Networks (CICOMMS), 2014 IEEE Symposium on*. [S.l.], 2014. p. 1–8. Citado na página 20.
- LOZANO, D. Modelagem matemática e aplicações do problema de coloração em grafos. Universidade Estadual Paulista (UNESP), 2007. Citado na página 27.
- LUIZ, A. G. Coloração de grafos e suas aplicações. *V Workshop de Tecnologia da Informação do Sertão Central - Universidade Federal do Ceará – Campus Quixadá*, 2015. Citado na página 19.

- MACEDO, H. R.; LIMA, H. H. d. C.; TOLENTINO, C. H. C. Desenvolvimento de meta-heurística baseada em simulated annealing aplicado a coloração de grafos para otimização de processos. *Sistema Eletrônico de Administração de Conferências, 8ª JICE - Jornada de Iniciação Científica e Extensão*, 2017. Citado na página 28.
- MARAPPAN, R.; SETHUMADHAVAN, G.; SRIHARI, R. New approximation algorithms for solving graph coloring problem—an experimental approach. *Perspectives in Science*, Elsevier, v. 8, p. 384–387, 2016. Citado na página 28.
- NETO, A. S. A.; GOMES, M. J. N. Problema e algoritmos de coloração em grafos-exatos e heurísticos. *Revista de Sistemas e Computação-RSC*, v. 4, n. 2, 2015. Citado na página 31.
- NEUFELD, G.; TARTAR, J. Graph coloring conditions for the existence of solutions to the timetable problem. *Communications of the ACM*, ACM, v. 17, n. 8, p. 450–453, 1974. Citado na página 20.
- OPTIMIZATION, G. Gurobi optimizer 7.5.2. *Gurobi*: <http://www.gurobi.com>, 2017. Citado na página 39.
- OSTROSKI, A.; MENONCINI, L. Aplicações práticas da teoria dos grafos. *Pato Branco: XIII ERMAC*, 2009. Citado 2 vezes nas páginas 18 e 19.
- PAIVA, S. M. d. A. *A programação linear no ensino secundário*. Dissertação (Mestrado), 2008. Citado na página 22.
- PUREZA, V. Tópicos em gerência da produção (métodos heurísticos). *Notas de Aula, Universidade Federal de São Carlos*, 2010. Citado na página 25.
- REGO, M. F.; SANTOS, H. G. Algoritmos para o problema de coloração de grafos. *Computational optimization and applications*, v. 19, n. 2, p. 165–178, 2001. Citado 2 vezes nas páginas 22 e 26.
- ROCHA, A.; DORINI, L. B. Algoritmos gulosos: definições e aplicações. *Campinas, SP*, p. 42, 2004. Citado na página 30.
- SANTOS, H. G. et al. Integer programming techniques for the nurse rostering problem. *Annals of Operations Research*, Springer, v. 239, n. 1, p. 225–251, 2016. Citado na página 25.
- SOUZA, M. J. F. Inteligência computacional para otimização. *Notas de aula, Departamento de Computação, Universidade Federal de Ouro Preto, disponível em <http://www.decom.ufop.br/prof/marcone>*, 2008. Citado 2 vezes nas páginas 19 e 32.
- ZUFFEREY, N. Graph coloring tabu search for project scheduling. In: *Advanced Computational Methods for Knowledge Engineering*. [S.l.]: Springer, 2015. p. 107–118. Citado na página 20.

Apêndices

APÊNDICE A – Experimentos *Recol*

A.1 Testes Heurística de Refinamento - *recol*

As [Tabela 15](#) e [Tabela 16](#) mostram os resultados dos testes feitos com heurística *recol* a fim de fazer a comparação entre as heurísticas apresentadas nas tabelas [11](#) e [12](#). Os experimentos foram repetidos cinco vezes cada devido a aleatoriedade presente no algoritmo. Dessa forma, é possível identificar na tabela a coloração obtida em cada experimento, a coloração mínima, a coloração máxima, o tamanho médio das colorações e o GAP em comparação com a coloração ótima ou a melhor possível encontrada na literatura¹, utilizando a formula descrita na [Equação 5.1](#). Para melhor visualização, como foi feito nos demais testes, as intâncias cuja a solução ótima foi encontrada já na geração da solução inicial foram omitidas.

Tabela 15 – Experimentos *recol* (Grupo 1)

Grafo	s*	Repli1	Repli2	Repli3	Repli4	Repli5	s_{min}	s_{max}	$s_{média}$	GAP
queen6_6	7*	8	8	8	8	8	8	8	8,00	0,13
queen7_7	7*	9	9	9	9	9	9	9	9,00	0,22
queen8_8	9*	11	10	10	10	10	10	11	10,20	0,10
queen9_9	10*	12	11	11	11	11	11	12	11,20	0,09
DSJC125.1	5*	6	6	6	6	6	6	6	6,00	0,17
DSJC125.5	17*	20	18	19	19	19	18	20	19,00	0,06
DSJC125.9	44*	45	46	45	45	45	45	46	45,20	0,02
DSJC250.1	8*	10	9	9	9	9	9	10	9,20	0,11
DSJC250.9	72*	77	75	75	77	75	75	77	75,80	0,04

Tabela 16 – Experimentos *recol* (Grupo 2)

Grafo	s*	Repli1	Repli2	Repli3	Repli4	Repli5	s_{min}	s_{max}	$s_{média}$	GAP
DSJC250.5	28	32	31	31	31	31	31	32	31,20	0,10
DSJC500.1	12	14	14	14	14	14	14	14	14,00	0,14
DSJC500.5	48	54	54	54	54	54	54	54	54,00	0,11
DSJC1000.1	20	22	22	22	22	22	22	22	22,00	0,09
DSJR500.1c	85	85	85	85	85	85	85	85	85,00	0,00
DSJR500.5	122	129	129	129	128	129	128	129	128,80	0,05
le450_15c	15	22	20	21	22	21	20	22	21,20	0,25
le450_15d	15	22	22	21	21	21	21	22	21,40	0,29
le450_25c	25	28	29	27	28	28	27	29	28,00	0,07
le450_25d	25	27	27	27	27	27	27	27	27,00	0,07

¹ Valores obtidos em ([DOURI; ELBERNOUSSI, 2015](#))

APÊNDICE B – Experimentos *Simulated Annealing* (SA)

B.1 Testes Meta-Heurística - *Simulated Annealing* (SA)

As [Tabela 17](#) e [Tabela 18](#) mostram os resultados dos testes feitos com meta-heurística *Simulated Annealing* a fim de fazer a comparação entre as heurísticas apresentadas nas tabelas [11](#) e [12](#). Baseado nos experimentos anteriores, a mesma metodologia foi utilizada, ou seja, os experimentos foram repetidos cinco vezes cada devido a aleatoriedade presente no algoritmo. Nas tabelas abaixo é possível identificar a coloração obtida em cada experimento, a coloração mínima, a coloração máxima, o tamanho médio das colorações e o GAP em comparação com a coloração ótima ou a melhor possível encontrada na literatura¹, utilizando a formula descrita na [Equação 5.1](#). Novamente, para melhor visualização as instâncias cuja a solução ótima foi encontrada já na geração da solução inicial foram omitidas.

Tabela 17 – Experimentos SA (Grupo 1)

Grafo	s*	Repli1	Repli2	Repli3	Repli4	Repli5	s_{min}	s_{max}	$s_{média}$	GAP
queen6_6	7*	8	8	8	8	8	8	8	8,00	0,13
queen7_7	7*	9	9	9	9	9	9	9	9,00	0,22
queen8_8	9*	10	10	10	10	10	10	10	10,00	0,10
queen9_9	10*	12	11	12	11	11	11	12	11,40	0,09
DSJC125.1	5*	6	6	6	6	6	6	6	6,00	0,17
DSJC125.5	17*	19	19	20	19	19	19	20	19,20	0,11
DSJC125.9	44*	45	46	45	46	46	45	46	45,60	0,02
DSJC250.1	8*	9	9	9	9	9	9	9	9,00	0,11
DSJC250.9	72*	85	89	88	87	88	85	89	87,40	0,15

¹ Valores obtidos em ([DOUIRI; ELBERNOUSSI, 2015](#))

Tabela 18 – Experimentos SA (Grupo 2)

Grafo	s*	Repli1	Repli2	Repli3	Repli4	Repli5	s_{min}	s_{max}	$s_{média}$	GAP
DSJC250.5	28	32	31	32	33	32	31	33	32,00	0,10
DSJC500.1	12	14	14	15	14	14	14	15	14,20	0,14
DSJC500.5	48	64	60	64	62	61	60	64	62,20	0,20
DSJC1000.1	20	26	25	25	25	24	24	26	25,00	0,17
DSJR500.1c	85	90	89	90	90	90	89	90	89,80	0,04
DSJR500.5	122	129	129	129	129	129	129	129	129,00	0,05
le450_15c	15	23	23	23	23	23	23	23	23,00	0,35
le450_15d	15	23	23	23	23	23	23	23	23,00	0,35
le450_25c	25	28	28	29	28	28	28	29	28,20	0,11
le450_25d	25	28	27	28	27	28	27	28	27,60	0,07