UNIVERSIDADE FEDERAL FLUMINENSE

VICTOR MIRANDA RANGEL SILVA

Abordagens Exatas e Heurísticas para o Problema da Coloração Robusta

NITERÓI

2014

UNIVERSIDADE FEDERAL FLUMINENSE

VICTOR MIRANDA RANGEL SILVA

Abordagens Exatas e Heurísticas para o Problema da Coloração Robusta

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Algoritmos e Otimização.

Orientador:
Yuri Abitbol de Menezes Frota

Co-orientador:
Luidi Gelabert Simonetti

NITERÓI

2014

Victor Miranda Rangel Silva

Abordagens Exatas e Heurísticas para o Problema da Coloração Robusta

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Algoritmos e Otimização.

Aprovada em 09 de abril de 2014.

BANCA EXAMINADORA

Prof. D.Sc. Yuri Abitbol de Menezes Frota / IC-UFF / Orientador

---

Prof. D.Sc. Luidi Gelabert Simonetti / IC-UFF / Co-Orientador

---

Prof. D.Sc. Luiz Satoru Ochi / IC-UFF

Prof. D.Sc. Laura Silvia Bahiense da Silva Leite / COPPE/UFRJ

Niterói

2014

À minha família e amigos.

Agradecimentos

Agradeço a Deus, pelas dificuldades superadas e objetivos alcançados. À minha família e aos meus amigos, pelo apoio e paciência que tiveram comigo ao longo deste trabalho. Aos professores Yuri e Luidi, pela orientação, paciência e confiança. Aos professores Satoru, Plastino e Laura, pelos conselhos e sugestões. Aos demais professores do IC-UFF pelos ensinamentos. À UFF, pela oportunidade proporcionada.

Resumo

Este trabalho busca estudar uma variação do clássico Problema da Coloração Mínima (PCM), chamado Problema da Coloração Robusta (PCR). Este problema, relativamente pouco explorado na literatura, busca encontrar uma k-coloração em um grafo G, ou seja, uma coloração de G que utilize no máximo k cores, ponderando toda a topologia de G. Cada coloração encontrada possui um *nível de rigidez* R, sendo R definido pela soma das penalidades das anti-arestas de G cujos vértices possuam a mesma cor. Algumas das aplicações do PCR envolvem: alocação de horários, clusterização e quaisquer outros problemas que possam ser modelados como problemas de coloração, mas com uma quantidade máxima de recursos (cores) definida. Neste trabalho é proposto um algoritmo baseado na metaheurística ILS, utilizando os métodos *Variable Neighborhood Descent* (VND) e *Random Variable Neighborhood Descent* (RVND). Também é proposta uma nova formulação matemática para o PCR. Os resultados obtidos mostram que o algoritmo proposto é competitivo tanto em relação aos algoritmos heurísticos quanto aos algoritmos exatos já existentes na literatura para resolver o PCR.

Palavras-chave: Problema da Coloração Robusta, *Iterated Local Search*, *Variable Neighborhood Descent*.

Abstract

This work deals with the Robust Coloring Problem (RCP). This problem is a variation of the classical Minimum (Vertex) Coloring Problem (MCP), since each coloring has a rigity level, defined by the sum of the penalties of the graph complementary edges, whose vertices has the same color. The RCP problem is NP-hard and occurs in areas of applications such as scheduling, clusterization and problems that can be modeled as coloring problems, with a fixed amount of resources (colors). Due to the combinatorial complexity of the problem its resolution by pure exact methods is, in many cases, computationally impractical. This fact motivates the development of heuristic algorithms, which are usually faster but do not guarantee the best solution for the problem. This work proposes an algorithm based in the ILS metaheuristic, using Variable Neighborhood Descent methods and a representative model for the RCP problem. The computational results demonstrate that the proposed approach is quite competitive against known algorithms that address the same problem.

Keywords: Robust Coloring Problem, *Iterated Local Search*, *Variable Neighborhood Descent*.

Sumário

Lista de Abreviaturas e Siglas

PCR	:	Problema da Coloraçao Robusta;
ILS	:	*Iterated Local Search*;
VND	:	*Variable Neighborhood Descent*;
RVND	:	*Random Variable Neighborhood Descent*;
RCP	:	*Robust Coloring Problem*;

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Alguns problemas computacionais podem ser modelados como problemas de coloração de vértices em grafos (e.g. alocação de frequências de celulares, alocação de horários, roteamento e atribuição de frequências em redes óticas e etc). Nestes problemas, é necessário encontrar uma coloração própria, ou seja, uma atribuição de cores aos vértices de um grafo, de modo que dois vértices adjacentes não possuam a mesma cor. Um exemplo do que pode ser modelado como problema de coloração é a alocação de horários de aulas em uma faculdade, de modo que um aluno matriculado em duas aulas consiga frequentar ambas, ou seja, essas duas aulas não podem ser alocadas no mesmo horário. Neste exemplo, os vértices do grafo representam as aulas, cada par de aulas incompatíveis é ligado por uma aresta, e cada cor representaria um horário. No Problema da Coloração Mínima, o objetivo é encontrar uma coloração própria que minimize a quantidade de cores distintas atribuidas aos vértices. Essa quantidade mínima de cores também é chamada de número cromático de um grafo.

Porém, em alguns casos, este modelo pode ser muito restritivo. Considerando o exemplo acima, podemos supor que a quantidade de horários disponíveis seja fixa (k cores), o que tornaria o número de cores (horários) uma constante e não um resultado a ser otimizado. Desta maneira, tem-se um novo objetivo: encontrar uma coloração que utilize até k cores de modo a otimizar algum outro critério.

Tendo isso como base, um novo problema de coloração de grafos foi proposto por Yañez e Ramírez [30]. Neste problema, vértices adjacentes não podem possuir a mesma cor, assim como no problema de coloração original, porém agora as arestas complementares ou anti-arestas (i.e. arestas formadas entre vértices não adjacentes) possuem pesos que são levados em consideração. O objetivo do novo problema agora é, utilizando no máximo k cores, encontrar uma coloração que minimize a soma dos pesos das arestas complementares

entre vértices que possuem a mesma cor. Com isso, a ideia é minimizar a probabilidade de se incluir no grafo uma aresta com os dois extremos possuindo a mesma cor. A esse problema foi dado o nome de *Problema de Coloração Robusta* (PCR).

1.1 Motivação

O PCR ainda é um problema relativamente novo e pouco explorado na literatura, porém com a capacidade de dar uma nova perspectiva ao Problema da Coloração Mínima, um dos mais clássicos problemas de otimização, abrindo espaço para a abordagem de novas aplicações práticas (ver Seção 2.2). Neste contexto, a proposta de mais uma heurística para resolver este problema teve como meta o desenvolvimento de um *Iterated Local Search* (ILS) híbrido e a demonstração de que ele é competitivo frente às abordagens já existentes.

1.2 Objetivos do Trabalho

O presente trabalho tem como objetivo a criação de um algoritmo eficiente para a resolução do Problema de Coloraçao Robusta, utilizando a metaheurística Iterated Local Search e os métodos *Variable Neighborhood Descent* (VND) e *Random Variable Neighborhood Descent* (RVND).

Os objetivos específicos a serem alcançados incluem:

- efetuar uma revisão bibliográfica detalhada e atualizada sobre o PCR, além de uma revisão das metodologias utilizadas para esse problema;
- coletar, na literatura, problemas-teste para o PCR;
- estudar as vizinhanças disponíveis em pesquisas anteriores;
- avaliar o desempenho dos algoritmos desenvolvidos, comparando com os melhores resultados em trabalhos anteriores;

1.3 Organização do Texto

Este trabalho se organiza em cinco capítulos, incluindo esta introdução. No Capítulo 2 define-se a notação utilizada durante o texto e faz-se uma revisão da literatura. Neste capítulo também é feita a exposição de formulações matemáticas para o problema, bem

como a apresentação e a discussão de uma nova modelagem matemática. No Capítulo 3 são discutidas a metaheurística e as vizinhanças empregadas. Também são expostas, de forma detalhada, as características de implementação dos algoritmos ILS-RVND e ILS-VND, assim como o procedimento para a construção de uma solução inicial. No Capítulo 4 são descritos os experimentos computacionais realizados, os quais são avaliados e discutidos. Por fim, o Capítulo 5 apresenta as considerações finais e ideias para trabalhos futuros.

Capítulo 2

Problema de Coloração Robusta

O Problema de Coloração Robusta é uma variação do Problema de Coloração Mínima (PCM). No PCM busca-se encontrar uma coloração para um grafo utilizando a menor quantidade possível de cores. Porém, segundo [30], este pode ser um modelo muito restritivo para alguns problemas em que a quantidade de recursos que podem ser utilizados já está definida (e.g. problemas de alocação de horários [30]). Neste cenário, o Problema de Coloração Robusta foi estabelecido com o objetivo de ponderar, na função objetivo, toda a topologia do grafo e não apenas o número de cores utilizadas: uma coloração do grafo deve ser construída com um número fixo de cores, nos quais vértices adjacentes não podem receber a mesma cor e as arestas complementares (i.e. arestas que não pertencem ao grafo) de vértices de mesma cor serão penalizadas na função objetivo. Apesar de ser bastante semelhante ao PCM, o PCR não pode ser considerado uma generalização do problema de coloração em grafos [25].

Um exemplo de aplicação do PCR é a alocação de tripulações, onde cada cor representa uma tripulação, cada vértice representa um voo, cada aresta representa uma sobreposição de horários entre os voos e cada aresta complementar possui um custo, indicando a possibilidade de haver um atraso que cause a sobreposição de horários.

Entre suas aplicações possiveis estão os problemas de alocação de horários, clusterização, coloração de mapas [30], alocação de tripulação [28], alocação de frequências (*Frequency Assignment Problem - FAP*) [1] e quaisquer outros que possam ser modelados como problemas de coloração, mas com uma quantidade máxima de recursos a ser utilizado já definida.

2.1 Definição do Problema

Seja um grafo não-direcionado $G = (V, E)$, onde $V = \{1, 2, ..., n\}$ é o conjunto de vertices, sendo $|V| = n$, e E é o conjunto de arestas, com $|E| = m$. Seja também $c > 0$ um número inteiro, e $N(c) = \{1, 2, ..., c\}$ um conjunto de cores. Cada vértice deve possuir uma, e apenas uma, cor dentre as c cores disponíveis. Assim, uma coloração válida para o PCM pode ser descrita como um mapeamento $C : V \rightarrow N(c)$, de tal forma que dois vértices adjacentes i e j tenham cores diferentes, ou seja, $C(i) \neq C(j)$, $\forall i, j \in E$. Um grafo é passivel de coloração se o número de cores permitida for maior ou igual ao número cromático do grafo G, ou seja, $c \geq \chi(G)$.

O *nível de rigidez* $R(C)$ [30] de uma coloração C indica o quanto esta coloração pode ser considerada *robusta*, no sentido de que as arestas complementares, com vértices que possuem cores iguais, serão penalizadas e, consequentemente, esta coloração se tornaria inválida se essa aresta fosse adicionada. Portanto, um nível baixo de rigidez torna a coloração C mais robusta. No caso extremo de $c = n$, obtem-se o nível máximo de robustez, já que é possível colocar cada vértice com uma cor diferente, e assim atingir um nível de rigidez $R(C) = 0$.

Seja $\bar{E}$ o conjunto de arestas complementares, de tal forma que $i \neq j$ e $(i, j) \in \bar{E} \Leftrightarrow (i, j) \notin E$, e $\bar{G} = (V, \bar{E})$, o grafo complementar de G. Sejam dois vértices i e j, de tal forma que $(i, j) \in \bar{E}$ e i e j tenham a mesma cor. Com isso, tem-se uma penalidade que é denotada por $p_{ij} \geq 0$. Esta penalidade pode ser definida, por exemplo, como a probabilidade que a aresta (i, j) tem de ser adicionada no grafo G [30].

A partir disso, pode-se definir como a melhor coloração possivel aquela que é menos *rígida*, ou seja, a que possui o menor nível de penalidade: $R(C_R) = \min_{C} R(C)$, com $R(C) = \sum_{(i,j) \in \bar{E} | C(i) = C(j)} p_{ij}$. Assim, tem-se uma coloração C para o grafo G, de forma que, caso uma nova aresta $e \in \bar{E}$ seja incluída, a possibilidade de se mudar a coloração obtida anteriormente será mínima. Logo, o PCR é definido como o problema de determinar uma $c-$coloração viável do grafo (i.e. uma coloração que utiliza no máximo c cores) que minimiza o nível de rigidez. A Figura 2.1 ilustra um exemplo em que uma coloração mínima de cores não necessariamente produz a menor rigidez.

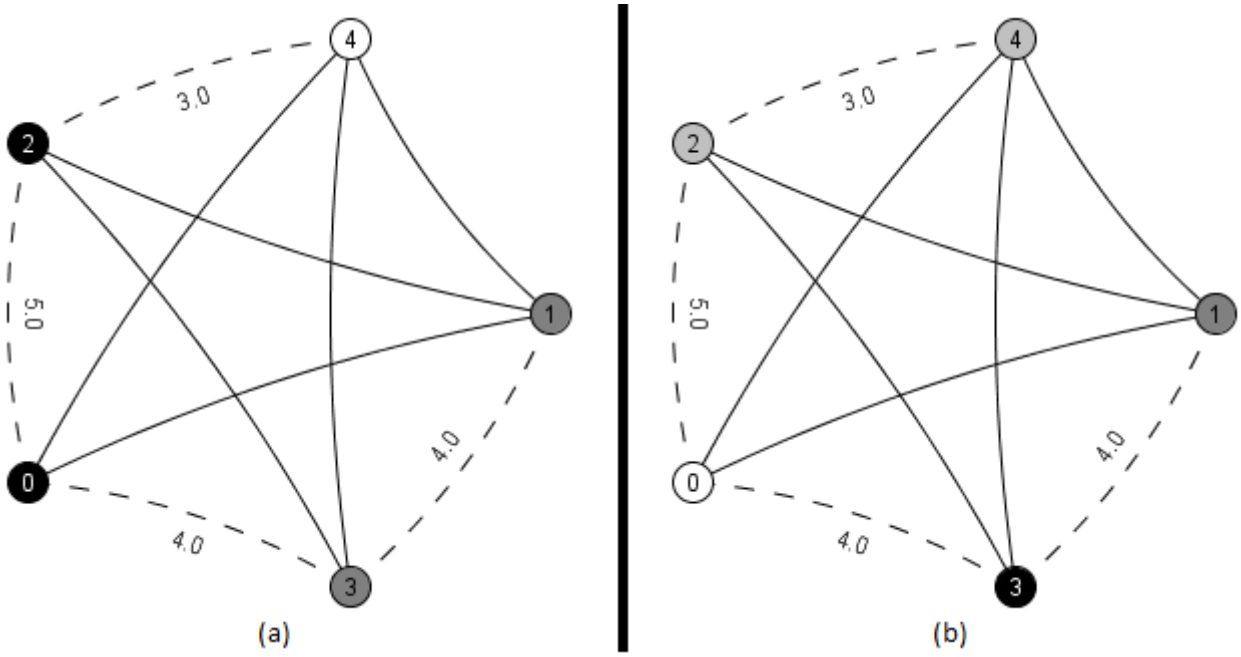

Figura 2.1: Exemplo de PCR. Na figura (a) uma coloração de 3 cores válida, c=4, com $R(C) = 9$. Na figura (b), a coloração robusta ótima do grafo utilizando 4 cores, c=4, com $R(C*) = 3$.

2.2 Revisão de Literatura

Como dito na Seção 1.1, o PCR é um problema relativamente novo. Ele foi proposto por Yañez e Ramírez [30] em 2003, tendo como motivação o Problema de Coloração Mínima (PCM). Segundo os autores, o PCM era muito restritivo para alguns problemas em que a quantidade de recursos disponíveis a serem compartilhados era conhecida (e.g. clusterização) ou em que havia a possibilidade de inclusão de novas arestas no grafo (e.g. alocação de recursos). Neste trabalho, os autores primeiro exibem a dificuldade do PCR, mostrando que este pertence a classe dos problemas $NP-$Completos. A seguir, um conjunto de aplicações é exibido para diferenciar bem o PCM do PCR, já que os exemplos apresentados não visam buscar uma quantidade mínima de cores, e sim utilizar uma quantidade definida de cores (recursos) na resolução do problema. Os autores propõem duas abordagens de solução: a primeira via um modelo exato para o problema e a segunda via um algoritmo genético [16]. Experimentos computacionais indicaram que o modelo exato proposto não é adequado para resolver grandes instâncias.

Em 2005, o PCR voltou a ser estudado por Lim e Wang [20]. Em seu artigo eles utilizam o PCR em uma aplicação com instâncias reais para o problema de alocação de aeronaves. Um dos motivos desta aplicação, segundo os autores, é a necessidade de se colocar a incerteza de atrasos de voos no processo de tomada de decisão. Além de

apresentarem este caso real, os autores definiram um espaço de busca, introduzindo duas maneiras de se representar uma solução do PCR: uma baseada em ordem de coloração e outra baseada em partições de cores. Também definiram duas estruturas de vizinhanças, entre as quais se destaca a estrutura *Improvement Graph Based k-exchange*, e dois métodos de geração de soluções iniciais. Por fim, fizeram uma comparação de quatro abordagens heurísticas: uma baseada em busca local, um *Simulated Annealing* [18], uma Busca Tabu [12] e um método híbrido, sequenciando a busca tabu com a busca local.

Em 2004, Lara-Velázquez et al. [29] publicaram um trabalho utilizando a técnica *Scatter Search* [11] para resolver o PCR, enquanto que em 2007, Gutiérrez-Andrade et al. [13] apresentaram um algoritmo baseado em *Simulated Annealing* para uma nova aplicação prática do PCR conhecida como *Alocação de Frequências no Espectro Eletromagnético* (AFEE). Em 2010, Gutiérrez-Andrade et al. [14] combinaram as duas abordagens apresentadas em [29] e [13], e propuseram dois novos algoritmos: um baseado em GRASP [5] e outro baseado em Busca Tabu. Em 2011, Laureano-Cruces et al. [19] apresentaram um caso de estudo para design e operação de sistemas manufaturas flexíveis (*Flexible Manufacturing Systems* (FMS)), além de um algoritmo de colônia de formigas para solucionar o problema [8].

Em 2011, Wang e Xu [28] publicaram um trabalho estendendo o trabalho anterior [20], no qual ampliam seus estudos sobre metaheurísticas para o PCR, apresentando algoritmos baseados em *Hill Climbing*, *Simulated Annealing*, Busca Tabu, Algoritmo Genético e um método híbrido, combinando a Busca Tabu com o *Hill Climbing* utilizando a vizinhança *Improvement Graph Based k-exchange*. Além disso, são apresentadas novas estruturas de vizinhança e um modelo de programação quadrática para o problema. Os autores também fizeram testes com instâncias de vários tamanhos, a fim de comparar as metaheurísticas propostas.

Em 2013, Archetti et al. [2] propuseram o primeiro algoritmo exato (*Branch-and-Price*) para resolver o PCR. Os autores também propuseram uma nova formulação matemática, de forma a contornar a dificuldade da solução de grandes instâncias, apontado por Yañez [30] na formulação original do problema. Além disso, este trabalho propõe um método de conversão de instâncias de coloração do DIMACS [17] para o PCR, além de disponibilizar um conjunto artificial de instâncias própria para o PCR.

2.3 Formulações Matemáticas Anteriores

Nesta seção, são apresentados duas formulações matemáticas. A formulação original, proposta por [30], e uma segunda formulação, mais robusta, proposta por [2].

2.3.1 Formulação Original

Com o objetivo de solucionar exatamente o PCR, o seguinte modelo de programação binária foi apresentado em [30]. Seja $x_{ik} \in \{0,1\}$ uma variável de decisão definida para todo índice $\{i \mid v_i \in V\}$ e para toda cor $k \in N(c)$ onde:

$$x_{ik} = \begin{cases} 1 & \text{caso } C(\text{i}) = k \\ 0 & \text{caso contrário} \end{cases}$$

Seja também a variável auxiliar $y_{ij} \in \{0,1\}$ definida para todo par de índices $\{i,j \mid v_i, v_j \in V \text{ e } (v_i, v_j) \in \bar{E}\}$ onde:

$$y_{ij} = \begin{cases} 1 & \text{se existir } k \in \{1,...,c\} \text{ tal que } x_{ik} = x_{jk} = 1 \\ 0 & \text{caso contrário} \end{cases}$$

Com isso, o modelo matemático para o PCR como um problema de programação binária é apresentado a seguir:

$$PCR = \min \sum_{(i,j)\in\bar{E}} p_{ij} y_{ij} \tag{2.1}$$

$$s.a. \quad \sum_{k=1}^{c} x_{ik} = 1, \forall\ i \in V \tag{2.2}$$

$$x_{ik} + x_{jk} \leq 1, \forall\ (i,j) \in E, \forall\ k \in \{1,...,c\} \tag{2.3}$$

$$x_{ik} + x_{jk} - 1 \leq y_{ij}, \forall\ (i,j) \in \bar{E}, \forall\ k \in \{1,...,c\} \tag{2.4}$$

$$x \in \{0,1\}^{|V|} \text{ e } y \in \{0,1\}^{|\bar{E}|} \tag{2.5}$$

Nesta formulação, a função objetivo 2.1 pondera as arestas complementares contidas em cada *classe de cor* (i.e. conjunto de vértices com a mesma cor). A equação 2.2 assegura que todos os vértices recebem exatamente uma cor, enquanto a equação 2.3 garante que dois vértices adjacentes não possam receber a mesma cor. A equação 2.4 assegura o valor

$y_{ij} = 1$ para quaisquer pares de vértices pertencentes a uma aresta complementar, desde que estes possuam a mesma cor. Por fim, as equações 2.5 impõe as restrições binárias às variáveis do problema.

Os autores mencionam que essa formulação consegue resolver apenas instâncias de pequeno porte (com até 15 vértices), pois o número de varáveis e restrições cresce proporcionalmente ao tamanho da instância (mais detalhes ver [30]). Para instâncias maiores (com mais de 15 vértices), é sugerida a utilização de heurísticas.

2.3.2 Formulação por Cobertura de Conjuntos

Para suprir o problema da formulação anterior, foi proposta em [2] uma nova formulação baseada em cobertura de conjuntos. Com o objetivo de apresentar este modelo, um *conjunto estável* foi definido como um conjunto de vértices onde seus elementos são dois a dois não-adjacentes (i.e. um conjunto s de vértices é estável se não existe aresta com ambas as pontas em s). Além disso, S foi definido como o conjunto de todos os conjuntos estáveis de G. Vale ressaltar que uma $c-$coloração viável de um grafo é definida por c conjuntos estáveis (classes de cores).

Seja então $\sigma_s \in \{0, 1\}$ uma variável de decisão definida para todo conjunto estável $s \in S$ onde:

$$\sigma_s = \begin{cases} 1 & \text{se } s \text{ for escolhida como uma classe de cor na coloração} \\ 0 & \text{caso contrário} \end{cases}$$

O PCR, então, irá procurar a solução ótima do seguinte modelo de cobertura de conjuntos:

$$PCR_{cc} = \min \sum_{s \in S} p_s \sigma_s \tag{2.6}$$

$$s.a. \sum_{s \in S : u \in s} \sigma_s \geq 1, \forall u \in V \tag{2.7}$$

$$\sum_{s \in S} \sigma_s = c \tag{2.8}$$

$$\sigma_s \in \{0, 1\}, \forall s \in S \tag{2.9}$$

onde $p_s = \sum_{u,v \in s} p_{uv}$.

Nesta nova formulação, a função objetivo 2.6 minimiza a rigidez da coloração. A restrição 2.7 afirma que cada vértice deve pertencer a pelo menos uma classe de cor,

enquanto a restrição 2.8 impõe o uso de exatamente c classes de cores. É importante notar que o número de conjuntos estáveis em S, normalmente, é exponencial em relação ao número de vértices. Logo, o modelo acima é resolvido utilizando um algoritmo de *branch-and-price* [3, 7] onde, a cada nó da árvore, a relaxação linear de 2.6 a 2.9, acrescida de restrições de *branching*, é resolvida por geração de colunas [23].

De forma resumida, o procedimento de geração de colunas começa com a identificação de um subconjunto restrito de conjuntos estáveis $S' \subseteq S$, que cubra os vértices de G. A partir desta solução inicial, o problema relaxado restrito a S' é solucionado. Isto fornecerá uma solução ótima para o problema relaxado, assim como sua solução dual que servirá para guiar a procura por novos conjuntos estáveis (variáveis) que possam melhorar a função objetivo através da expanção do conjunto S'. Se um conjunto estável s for encontrado dessa forma, o conjunto $S' = \{S' \cup s\}$ é atualizado, e o processo é reiniciado. Esta tarefa é feita resolvendo iterativamente um subproblema chamado *problema de pricing*.

O problema de *pricing* consiste em encontrar um conjunto estável s com o maior *custo reduzido negativo* [23]. Para isso, são consideradas as seguintes variáveis binárias:

- $x_u = 1$ se o vértice $u \in V$ pertence ao conjunto estável s. Caso contrário, 0.
- $y_{uv} = 1$ se os vértices $u, v \in V$ pertencem ao conjunto estável s. Caso contrário, 0.

Pode-se então definir o problema de *pricing* de PCR_{cc} como:

$$PCR_{cc}^{aux} = \min \sum_{uv \in \bar{E}} p_{uv} y_{uv} - \sum_{u \in V} \pi_u x_u - \beta \tag{2.10}$$

$$s.a. \quad x_u + x_v \leq 1, \forall\ (u,v) \in E \tag{2.11}$$

$$x_u + x_v - 1 \leq y_{uv}, \forall\ (u,v) \in \bar{E} \tag{2.12}$$

$$x_u \in \{0,1\}, \forall\ u \in V \tag{2.13}$$

$$y_{uv} \in \{0,1\}, \forall\ (u,v) \in \bar{E} \tag{2.14}$$

onde:

- π_u é o custo dual associado à restrição 2.7 para o vértice u;
- β é o custo dual associado à restrição 2.8.

Vê-se então que a restrição 2.11 impõe que não podem existir arestas entre os vértices escolhidos para compor o conjunto estável, enquanto a restrição 2.12 associa as variáveis

y com as variáveis x. É interessante notar que as restrições 2.11 e 2.12 tem a mesma função que as restrições 2.3 e 2.4 do modelo original apresentado anteriormente.

2.4 Uma Nova Formulação para o PCR

Nesta seção é apresentada uma nova formulação para PCR baseada no modelo dos representantes apresentada por Campêlo et al. [6], que procura construir uma formulação tão compacta quanto o modelo PCR. O modelo é apresentado a seguir.

Uma coloração robusta de G pode ser vista como uma família $S_1, ..., S_c$ de c conjuntos estáveis de G, em que cada conjunto está associada a uma cor (ou classe de cor). Suponha que, para cada cor i, seja escolhido um vértice para ser o representante da correspondente classe de cor S_i. Então, cada vértice pode estar em um destes dois estados: ou ele representa uma determinada classe de cor ou existe outro vértice que representa sua cor. Para descrever esta situação, definem-se as variáveis binárias w_{uv}, para todo $u \in V$ e v pertencente ao conjunto de *anti-vizinhos* de v, definido por $A(v)$ (i.e. vértices ligados por arestas complementares a v). Este conjunto de variáveis binárias terá a seguinte interpretação: $w_{uv} = 1$ se e somente se u representar a cor de v. Além disso, definem-se também as variáveis binárias auxiliares y_{uv}, para todo $u, v \in \bar{E}$, onde $y_{uv} = 1$ se e somente se os vértices u e v possuírem a mesma cor. Os vetores (w, y), formados por todas estas variáveis binárias, são vetores incidentes a uma coloração robusta ótima de G se existe uma solução ótima do seguinte modelo de programação inteira:

$$PCR_{rep} = \min \sum_{uv \in \bar{E}} y_{uv}.p_{uv} \tag{2.15}$$

$$s.a. \quad \sum_{v \in A'(u)} w_{vu} = 1 \quad \forall u \in V \tag{2.16}$$

$$\sum_{v \in V} w_{vv} = k \tag{2.17}$$

$$w_{uv} + w_{ul} \leq w_{uu} \quad \forall u \in V, \quad \forall (v,l) \in E \text{ com } v, l \in A(u) \tag{2.18}$$

$$w_{uv} \leq w_{uu} \quad \forall u \in V, \quad \forall v \in A(u) \text{ onde } v \text{ é isolado em } A(u) \tag{2.19}$$

$$w_{uv} \leq y_{uv} \quad \forall u \in V, \quad \forall v \in A(u) \tag{2.20}$$

$$w_{uv} + w_{ul} + w_{vl} \leq w_{uu} + y_{vl}, \quad \forall u \in V, \forall (v,l) \in \bar{E} \text{ com } v, l \in A(u) \tag{2.21}$$

$$w \in \{0,1\}^{|V|+|\bar{E}|} \text{ e } y \in \{0,1\}^{|\bar{E}|} \tag{2.22}$$

onde $A'(u) = A(u) \cup \{u\}$.

O primeiro grupo de restrições indica que cada vértice $u \in V$ precisa ser representado ou por si mesmo ou por outro vértice em sua antivizinhança. Isto significa que esta restrição força que exatamente uma cor seja atribuída para u. A restrição força o modelo a possuir k representantes, um para cada cor. Como os vértices extremos de cada aresta precisam receber cores diferentes, as restrições 2.18 asseguram que eles tenham representantes diferentes. As restrições 2.18 e 2.19 garantem que um vértice v possa ser representado apenas por um vértice u que seja representante. Por fim, as restrições 2.20 e 2.21 asseguram a interpretação correta da variável y enquanto que a restrição 2.22 impõe restrições binárias a w e y.

Capítulo 3

Metodologia Proposta

Nesse capítulo são expostos: o modo como uma solução do PCR é representada; os procedimentos para a construção de uma solução inicial; as estruturas de vizinhança empregadas; os métodos de busca (*Random Variable Neighborhood Descent* e *Variable Neighborhood Descent*) e um algoritmo *Iterated Local Search* (ILS) proposto para solucionar o problema.

3.1 Representação de uma Solução

No presente trabalho, uma solução é representada como uma matriz binária nxc sendo n a quantidade de vértices do grafo e c a quantidade de cores. Neste caso, cada linha i conterá exatamente uma coluna j com valor 1, indicando que o vértice i possui a cor j. No exemplo da Figura 2.1 , para o grafo (a) há a representação $x_{0,1} = x_{1,2} = x_{2,1} = x_{3,2} = x_{4,3} = 1$ (conforme mostra a Tabela 3.1), para $c = 4$. Já no caso do grafo (b) da mesma figura, há a representação $x_{0,3} = x_{1,2} = x_{2,4} = x_{3,1} = x_{4,4} = 1$, também para $c = 4$.

	$\mathbf{1}(\boldsymbol{preto})$	$\mathbf{2}(\boldsymbol{cinza})$	$\mathbf{3}(\boldsymbol{branco})$	$\mathbf{4}(\boldsymbol{cinzaclaro})$
$\boldsymbol{v_0}$	1	0	0	0
$\boldsymbol{v_1}$	0	1	0	0
$\boldsymbol{v_2}$	1	0	0	0
$\boldsymbol{v_3}$	0	1	0	0
$\boldsymbol{v_4}$	0	0	1	0

Tabela 3.1: Tabela de representação de solução para o grafo da Figura 2.1(a)

Esta representação de solução pode ser usada para denotar um tipo de esquema de coloração chamado de *Partition Based Encoding* (PBE), proposto por [9] para o problema clássico de coloração. Neste esquema, para cada coloração que usa exatamente c cores, a

solução pode ser representada como uma partição de V que contém c subconjuntos não-vazios disjuntos, denotados por $V_1, V_2, ..., V_c$, onde V_t para $1 \leq t \leq c$ indica o conjunto de vértices que contém a cor t. Por exemplo, o grafo (a) da figura 2.1, com $c = 4$ poderia ser codificado no PBE como $\{\{0, 2\}, \{1, 3\}, \{4\}, \{\}\}$. Já o grafo (b) da mesma figura poderia ser codificado como $\{\{3\}, \{1\}, \{0\}, \{2, 4\}\}$.

3.2 Estruturas de vizinhança

Neste trabalho, foram utilizadas 4 Estruturas de vizinhança básicas: SVR (*Single Vertex Recoloring*), MVR (*Multi Vertex Recoloring*), IG-K*-cycle (*Improvement Graph Based K*-cycle*) e ΘK-cycle. Cada uma dessas estruturas denota operadores para transformar uma coloração C em uma nova coloração, que é definida como vizinha de C. As vizinhanças MVR e ΘK-cycle são propostas neste trabalho, enquanto SVR foi proposta por [28] e IG-K*-cycle foi proposta por [20]. Algo importante a se notar é que as vizinhanças SVR e MVR permitem a inclusão de novas cores no gráfico, enquanto que as vizinhanças IG-K*-cycle e ΘK-cycle apenas mudam a configuração das cores já existentes na coloração.

3.2.1 Single Vertex Recoloring - SVR

Essa estrutura de vizinhança, proposta por [28] e apresentada no Algoritmo 1, consiste basicamente no processo de escolher um único vértice $v \in V$ e lhe dar uma nova cor t (diferente da anterior) onde $1 \leq t \leq c$. Para isso, todas as cores são testadas, até achar o movimento $C(v, t)$ (i.e. v recebe a cor t) que mais minimiza a função objetivo do problema.

Algoritmo 1 SVR(v, C)

```
1: p_melhor = ∞
2: C* = C
3: para t = 1 até c faça
4:    Gera nova coloração C(v, t)
5:    p = penalidade(C(v, t))
6:    se p < p_melhor então
7:       C* = C(v, t)
8:       p_melhor = p
9:    fim se
10: fim para
11: return C*
```

Com a utilização da representação PBE, é possível recalcular o custo de uma solução

vizinha (gerada pela mudança de cor em um vértice) em $O(n)$. Logo, o Algoritmo 1 possui complexidade $O(cn)$.

Em adição a esse operador básico, os autores em [28] expandem o processo original para um novo operador definido como *Random Single Vertex Recoloring* (RSVR) e que tem seu pseudocódigo representado no Algoritmo 2. Neste novo operador, adicionamos um parâmetro inteiro r, onde $1 \leq r \leq n$. Com isso, o RSVR vai escolher r vértices aleatoriamente para formar um subconjunto V_r de vértices. Posteriormente, para cada vértice $v \in V_r$ e para cada cor t onde $1 \leq t \leq c$, é escolhido o movimento $C(v,t)$ que proporcionar a menor penalidade.

Algoritmo 2 RSVR(r,C)

```
1: Inicializa aleatoriamente uma lista L com r vértices de V
2: p_melhor = ∞
3: C* = C
4: para j = 1 até r faça
5:    v = L[j]
6:    para t = 1 até c faça
7:       Gera nova coloração C(v,t)
8:       p = penalidade(C(v,t))
9:       se p < p_melhor então
10:         C* = C(v,t)
11:         p_melhor = p
12:      fim se
13:   fim para
14: fim para
15: return C*
```

No caso especial em que $r = n$, tem-se $V_r = V$, e assim todos os vértices em V são verificados. Este caso especial é chamado de *Enumerative Single Vertex Recoloring* (ESVR), onde a complexidade é definida como $O(n^2c)$. Caso contrário, a complexidade do operador RSVR é limitado por $O(rcn)$.

No exemplo da Figura 3.1, a vizinhança RSVR é aplicada. A lista de vértices é definida por $L = \{2, 0, 3, 1\}$. Para cada um dos vértices de L, há uma tentativa de modificar a cor para cada uma das c cores disponíveis para obter uma nova coloração. Neste exemplo, a mudança de cor do vértice 2 para a cor 4 (cinza claro) foi escolhida como a melhor mudança. A nova coloração, codificada pelo esquema PBE, é a seguinte: $C = \{\{0\}, \{1, 3\}, \{4\}, \{2\}\}$. Outras combinações também poderiam nos dar a mesma redução, como por exemplo, mudar a cor do vértice 0 para a cor 4 (cinza claro), ou reduções menores, como mudar o vértice 2 para 3 (branco). Outras poderiam nos dar colorações inválidas, como modificar a cor do vértice 3 para 3 (branco) ou 1 (preto).

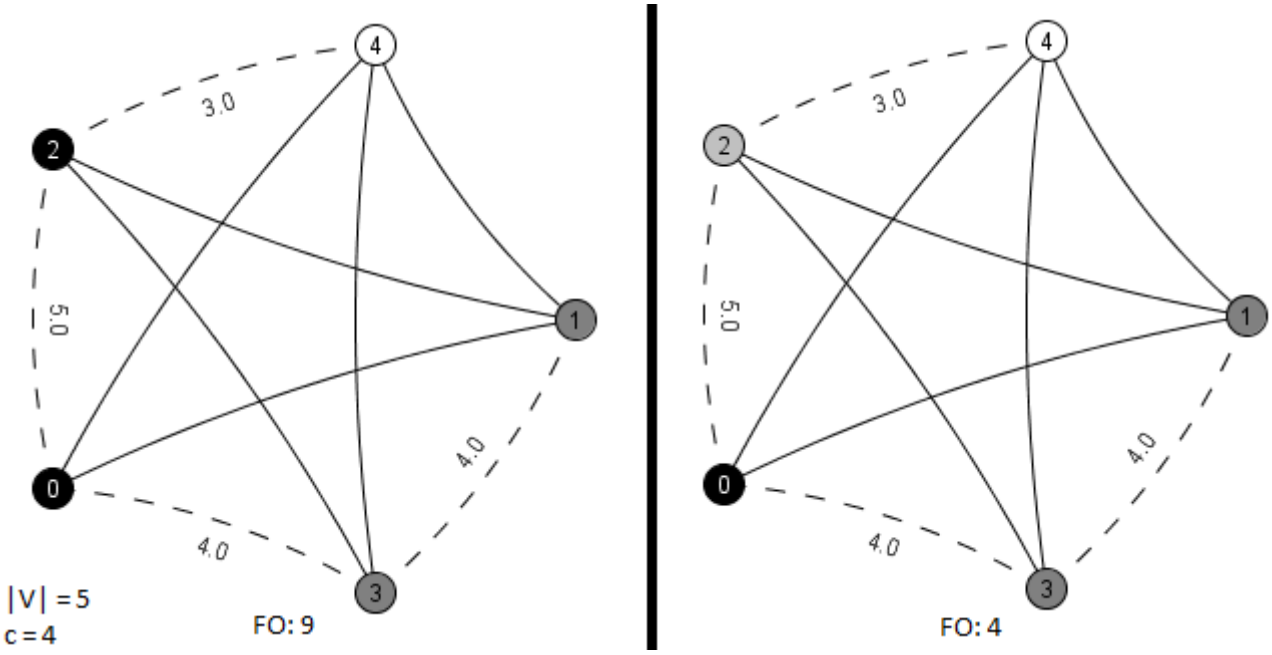

Figura 3.1: Exemplo de RSVR

3.2.2 Multi Vertex Recoloring - MVR

No presente trabalho foi desenvolvida uma nova estrutura de vizinhança, baseada na vizinhança SVR. Ela se concentra no fato de que não apenas uma cor pode ser modificada por vez, mas que todas as cores de um conjunto de α vértices podem ser modificadas ao mesmo tempo. A nova coloração escolhida é aquela que tiver a menor penalidade. O Algoritmo 3 é apresentado a seguir.

Algoritmo 3 MVR(α, C)

Inicializa aleatoriamente uma lista L com α vértices de V
Seja $\mathbb{C}$ a família de colorações geradas a partir de C com todas as combinações de cores possíveis dos vértices em L.
$p_{melhor} = \infty$
$C* = C$
para todo C_i em $\mathbb{C}$ **faça**
    $p = penalidade(C_i)$
    se $p < p_{melhor}$ **então**
        $C* = C_i$
        $p_{melhor} = p$
    fim se
fim para
return $C*$
end MVR

Sabendo-se que o número de colorações viáveis geradas por todas as combinações de c cores dos α vértices está limitada por c^α (i.e. $|\mathbb{C}| \leq c^\alpha$), e que é possível recalcular o custo

de uma solução vizinha gerada pela mudança de cor de α vértices em $O(\alpha n)$, pode-se definir a complexidade do Algoritmo 3 por $O(c^{\alpha}.\alpha n)$.

No exemplo da Figura 3.2, a vizinhança MVR é aplicada. A lista de vértices é definida por $L = \{2, 0, 3, 4\}$. Aqui, são analisadas todas as combinações de cores possíveis entre esses 4 vértices, e a configuração que nos dá a maior redução possível é escolhida. Neste caso, todos os vértices mudaram de cor, e a nova coloração ficou $C = \{\{3\}, \{1\}, \{0\}, \{2, 4\}\}$.

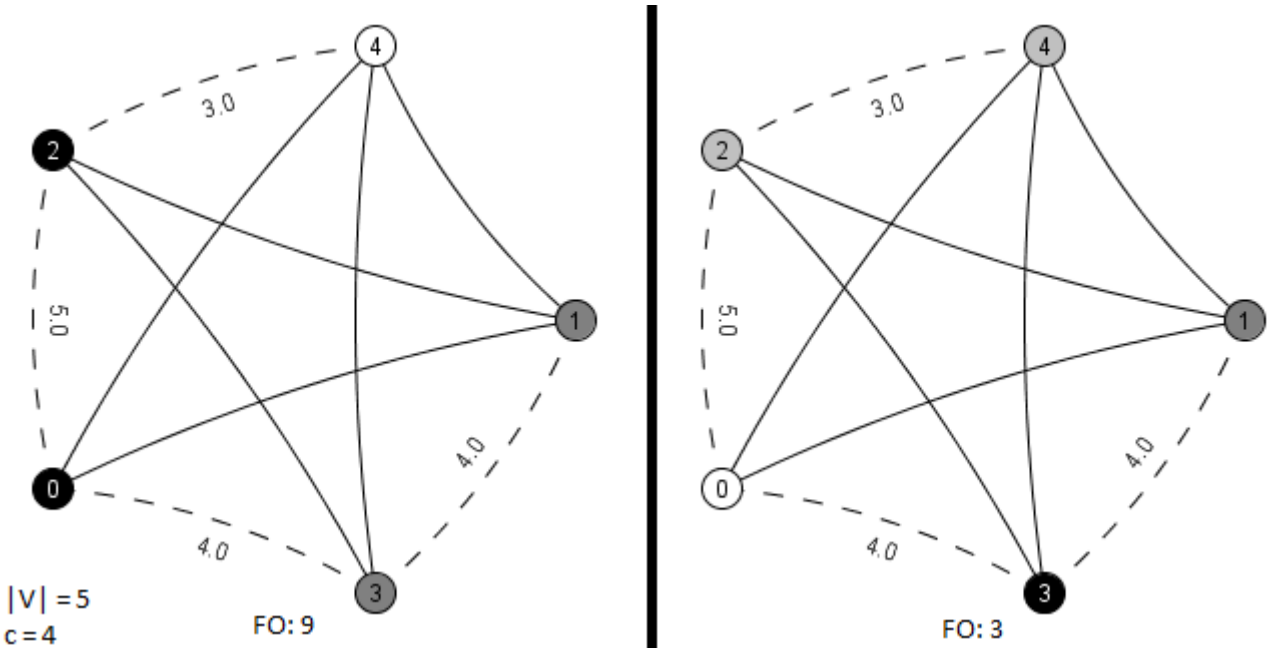

Figura 3.2: Exemplo de MVR

3.2.3 Improvement Graph Based K*-cycle - IG-K*-cycle

Este operador, desenvolvido em [20], é baseado na criação de um *grafo de melhoramento* direcionado $G' = (V', A)$ onde:

1. $V' = V$
2. $(i, j) \in A$ se e somente se $C(i) \neq C(j)$
3. O peso do arco (i, j), denotado por $w(i, j)$, é definido como a variação (positiva ou negativa) do *nível de rigidez* $R(C)$ quando há um esvaziamento da cor do vértice i (i.e. o vértice i fica descolorido e, consequentemente, fora da solução) enquanto ocorre a mudança de cor do vertice j de $C(j)$ para $C(i)$.

A partir deste novo grafo G', é criado um *ciclo de k-trocas*. Esse ciclo, denominado $ciclo(P_k)$, é uma sequência de k vértices distintos $\langle v_1, v_2, ..., v_k, v_{k+1} \rangle$ onde $v_{k+1} = v_1$ e

$(v_i, v_{i+1}) \in A \ \forall \ 1 \leq i \leq k$. Definiu-se $P_k = \{v_1, v_2, ..., v_k\}$ como o caminho (*path*) de k vértices do ciclo. Definiu-se também $A_c(P_k) = \{(v_1, v_2), (v_2, v_3), ..., (v_{k-1}, v_k), (v_k, v_1)\}$ como conjunto de arcos do ciclo. Além disso, se tem

$$w(ciclo(P_k)) = \sum_{(v_i, v_{i+1}) \in A_c(P_k)} w(v_i, v_{i+1})$$

como o custo do ciclo formado.

Realizando trocas das cores dos vértices que estão no ciclo, uma nova coloração C' é alcançada da seguinte forma: cada vértice j pertencente ao ciclo recebe a cor do vértice anterior i no ciclo (i.e. $C'(j) = C(i), \forall (i, j) \in A_c(P_k)$). Nota-se que a cor do vértice inícial do ciclo é igual a cor do vertice final (i.e. $C'(1) = C(k)$). Ao aplicar essa ciclagem de cores, vê-se que a variação (positiva ou negativa) em $R(C)$ é igual ao custo do ciclo $w(ciclo(P_k))$.

A Figura 3.3 dá um exemplo de um grafo G', criado a partir do grafo G à esquerda. O ciclo em vermelho é o melhor ciclo de troca.

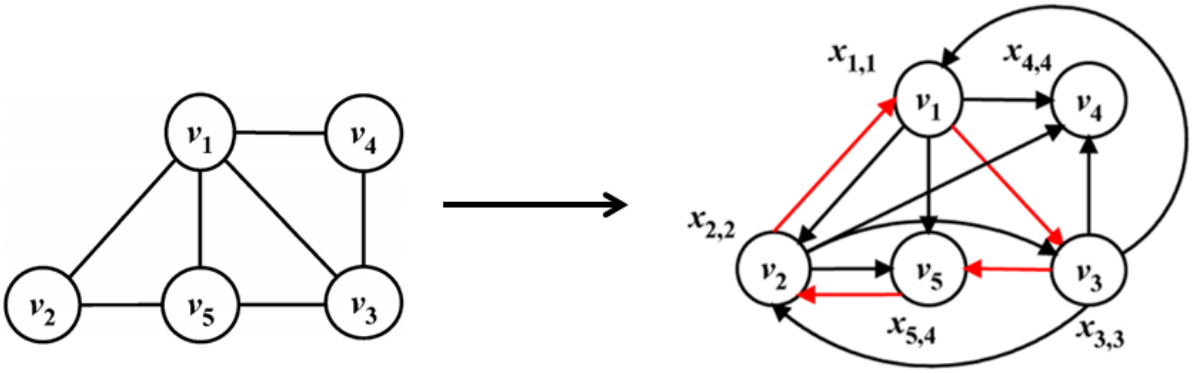

Figura 3.3: Exemplo de ciclo de trocas

Como há muitos ciclos k-troca possiveis num grafo de melhorias, foi desenvolvido um algoritmo de Programação Dinâmica (PD) [4] em tempo polinomial, de forma a encontrar o ciclo de redução ótimo [20]. Porém, antes, é preciso definir algumas notações:

- $Tamanho(P_k)$: O tamanho do caminho P_k
- $s(P_k)$: o primeiro vértice do caminho P_k
- $e(P_k)$: o último vértice do caminho P_k
- $P_k + v$: novo caminho com um vértice v adicionado ao final do caminho P_k

A Fórmula 3.1 de PD para obter o melhor ciclo de k-trocas é definida abaixo:

$$\begin{aligned} & P_{k+1} = min_{P_k+v}\{w(ciclo(P_k+v))\} \\ & \text{sujeito a} \\ & \quad \forall v \in V(G') \text{ onde } (e(P_k), v) \in E(G') \text{ e} \\ & \quad C(v) \neq C(v_j),\ \forall v_j \in P_k \end{aligned} \tag{3.1}$$

O algoritmo para buscar o melhor ciclo de melhora de k-trocas ($k \geq 2$) está definido no Algoritmo 4. Nele, o ciclo de melhora de k-trocas é definido como sendo o melhor ciclo contendo entre 2 e k vértices. Primeiramente, todas as arestas com peso negativo são adicionadas em uma lista de candidatos ($Lista$). Desta forma são consideradas todos os melhores caminhos possíveis com $k = 2$. Então, os melhores caminhos com tamanho 3 até k são obtidos através da fórmula de PD 3.1. Durante as iterações, o melhor caminho P_{melhor} com a maior redução de w_{melhor} é atualizado. Finalmente, o melhor ciclo de k-trocas é determinado.

Algoritmo 4 Algoritmo PD para melhor ciclo de k-trocas

1: $Lista = \emptyset$
2: $w_{melhor} = \infty$
3: $P_{melhor} = \emptyset$
4: **para todo** $(v_i, v_j) \in E'$ e $w(v_i, v_j) < 0$) **faça**
5: $Add(Lista, (v_i, v_j))$
6: **se** $w(ciclo(v_i, v_j)) < w_{melhor}$ **então**
7: $w_{melhor} = w(ciclo(v_i, v_j))$
8: $P_{melhor} = (v_i, v_j)$
9: **fim se**
10: **fim para**
11: **para** $l = 3$ até k **faça**
12: $P = Pop(Lista)$
13: **para todo** v tal que $(e(P), v) \in E', w(P + v) < 0$ e $C(v_j) \neq C(v),\ \forall v_j \in P$ **faça**
14: $P' = P + v$
15: **se** $(v, s(P)) \in E'$ e $w(ciclo(P')) < w_{melhor}$ **então**
16: $w_{melhor} = w(ciclo(P'))$
17: $P_{melhor} = P'$
18: **fim se**
19: **se** $Tamanho(P') \leq k$ **então**
20: $Add(Lista, P')$
21: **fim se**
22: **fim para**
23: **fim para**

Como o p̈roblema do caminho mais longoë NP-completo [10], a complexidade computacional é exponencial quando o Algoritmo 4 percorre todo o espaço de busca. Para balancear a acurácia da solução de acordo com o tempo de execução, uma lista de gerenciamento de candidados é criada no Algoritmo 5. Esta *Lista* é uma lista duplamente encadeada onde os elementos representam os caminhos candidatos em ordem crescente de w. A cada vez que um novo candidato é encontrado, ele é inserido em *Lista* de forma ordenada. Para controlar o tamanho da lista, há um parâmetro chamado $TamanhoMaxLista$ (ou θ). Quando o tamanho da *Lista* excede o tamanho máximo estabelecido, o último elemento da lista (ou seja, o que teoricamente tem menor possibilidade de ser o melhor caminho) é removido. Se $TamanhoMaxLista$ for grande o suficiente, a PD garante encontrar a solução ótima para o ciclo de k-troca. Por outro lado, um valor pequeno de $TamanhoMaxLista$ pode não conseguir a solução ótima, mas reduz o espaço de busca eficientemente. A complexidade do operador k-trocas é de $O(kn^2\theta)$ [20].

Algoritmo 5 Algoritmo para gerenciamento da lista de candidatos - $Add(Lista, P)$

1: Insira P na *Lista* em ordem crescente de $w(ciclo(P))$
2: **se** $Tamanho(Lista) > TamanhoMaxLista$ **então**
3: Retira o último item de *Lista*
4: **fim se**

No exemplo da Figura **??**fig:igkciclo, a vizinhança IG-K*-cycle é aplicada. O ciclo de mudança definido foi o correspondente aos vértices 0 e 4, formando a nova coloração $C = \{\{2,4\},\{1,3\},\{0\},\{\}\}$. Outro ciclo de redução possível, que produziria a mesma redução na penalidade, seria o ciclo dos vértices 0, 4 e 1, produzindo a nova coloração $C = \{\{2,4\},\{0,3\},\{1\},\{\}\}$.

3.2.4 ΘK-cycle

Esta estrutura de vizinhança é similar à *IG-K*-cycle* apresentada anteriormente. Porém, nesta nova vizinhança, não existe a restrição de colocar no ciclo apenas vértices com cores diferentes, qualquer subconjunto de vértices podem compor um ciclo de troca. O Algoritmo 6 ilustra o método ΘK*-cycle*.

O processo é iniciado com a seleção de k vértices aleatórios (linha 4). A seguir, são gerados ciclos com tamanhos de 2 a k observando as seguintes regras: **(i)** cria-se um ciclo com o primeiro vértice da lista (linha 5). **(ii)** para um $j \in \{2,..,k\}$, adiciona-se o vértice $vertices[j]$ no melhor ciclo de tamanho $j-1$ em cada uma das j posições possíveis (linhas 7 a 20). **(iii)** calcula-se a penalidade total de cada ciclo (linha 10) **(iv)** seleciona-se o melhor

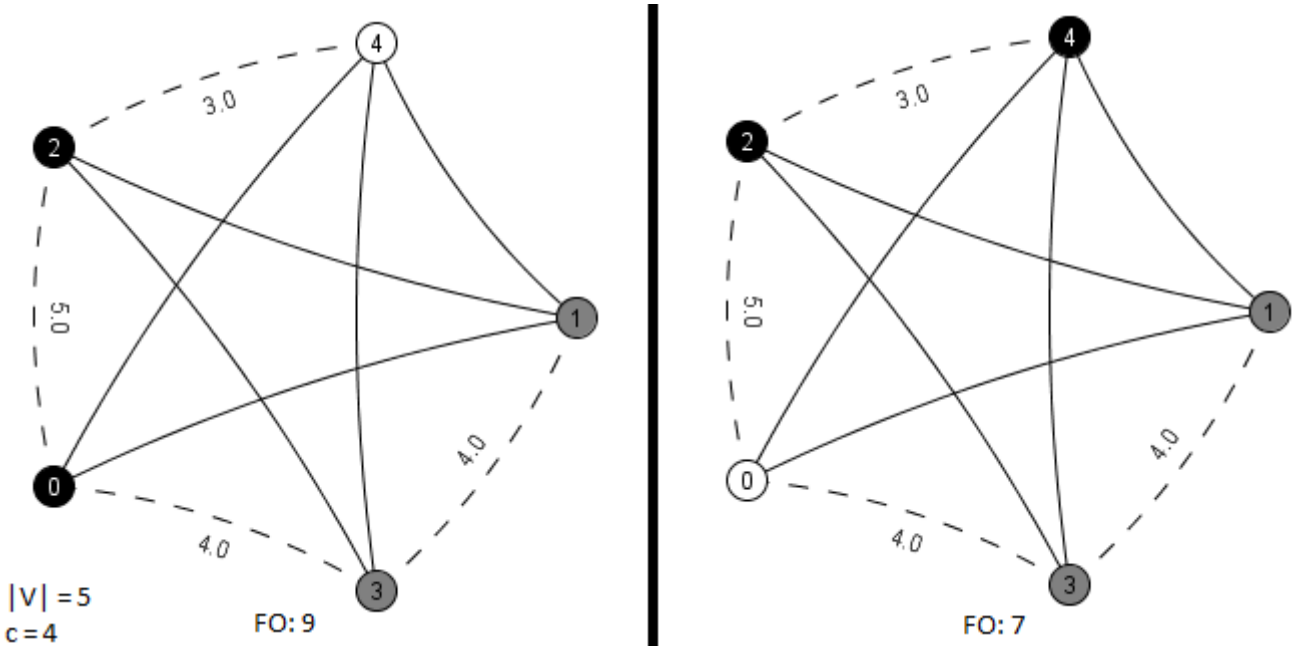

Figura 3.4: Exemplo de IG-K*-cycle

ciclo de tamanho j dentre esses ciclos criados (linhas 11 a 13) e o processo é reiniciado a partir de (ii); caso este seja o melhor ciclo encontrado até então, também é atualizado (linhas 14 a 17). Esse processo é repetido uma quantidade Θ de vezes, produzindo assim vários conjuntos de vértices aleatórios iniciais.

No exemplo da Figura 3.5, a vizinhança ΘK*-cycle é aplicada. O ciclo de mudança definido foi o correspondente aos vértices 3, 4, 0, 1 e 2, formando a nova coloração $C = \{\{1,3\},\{2,4\},\{0\},\{\}\}$. Outro ciclo de redução possível, que produziria a mesma redução na penalidade, seria o ciclo dos vértices 4, 0, 1, 2 e 3, produzindo a nova coloração $C = \{\{1,3\},\{2,4\},\{0\},\{\}\}$, a mesma coloração anterior. É preciso reparar que, neste ciclo, quase não há mudança nas cores, já que os ciclos de vértices fazem com que alguns vértices recebam novas cores iguais às que possuíam anteriormente. Ainda é possível produzir um ciclo com os vértices 2, 4 e 0, que produzirá a coloração $C = \{\{2,4\},\{1,3\},\{0\},\{\}\}$, que terá a mesma redução de penalidade, porém movendo um número menor de vértices.

Vizinhança	Complexidade
SVR	$O(cn)$
RSVR	$O(rcn)$
ESVR	$O(cn^2)$
MVR	$O(c^{\alpha}\alpha n)$
IG-K*-cycle	$O(kn^2\theta)$
ΘK-cycle	$O(kn^2\Theta)$

Tabela 3.2: Tabela de complexidade das vizinhanças

Algoritmo 6 Algoritmo da vizinhança ΘK-cycle

```
1: ciclo_melhor = ∅
2: P_melhor = ∞
3: para i = 1 até Θ faça
4:    vertices = lista de k vertices aleatórios
5:    ciclo_tamanho[1] = vertices[1]
6:    p_tamanho = ∞
7:    para j = 2 até k faça
8:       Ciclos = todos os ciclos colocando o vértice vertice[j] em cada posição {1, .., j}
         no ciclo_tamanho[j − 1]
9:       para todo ciclo(P_o) em Ciclos faça
10:         penal = w(ciclo(P_o))
11:         se penal < p_tamanho então
12:            p_tamanho = penal
13:            ciclo_tamanho[j] = ciclo(P_o)
14:            se penal < p_melhor então
15:               p_melhor = penal
16:               ciclo_melhor[j] = ciclo(P_o)
17:            fim se
18:         fim se
19:      fim para
20:   fim para
21: fim para
```

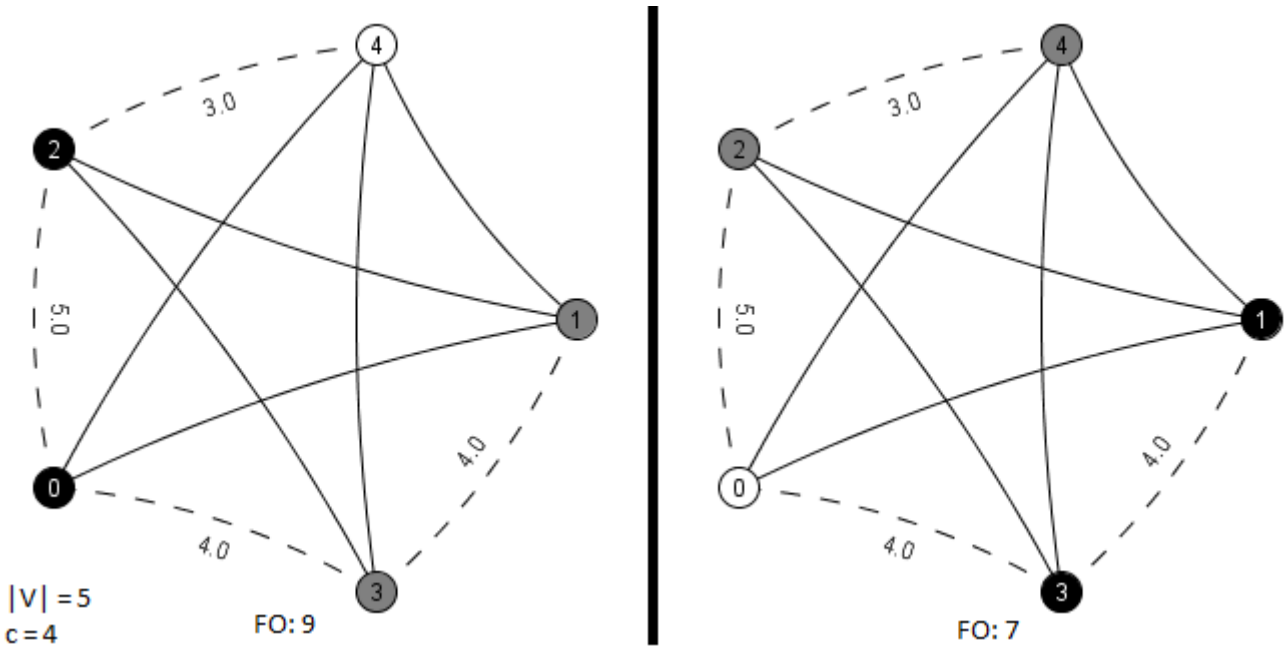

Figura 3.5: Exemplo de ΘK*-cycle

Na Tabela 3.2.4 ilustramos a complexidade de cada estrutura de vizinhança aqui apresentada.

3.3 Função de Avaliação

Em alguns momentos durante a execução do algoritmo desenvolvido, torna-se difícil caminhar por soluções viáveis devido à complexidade do problema. Assim, com o objetivo de fornecer mais liberdade ao método, resolveu-se permitir que o algoritmo caminhe por regiões inviáveis do espaço de busca. Esta é uma estratégia muito adotada em metaheurísticas. Desta forma, uma coloração C é avaliada por uma nova função de rigidez $\bar{R}$, que deve ser minimizada, e é apresentada na Equação 3.2.

$$\bar{R}(C) = \sum_{(i,j)\in\bar{E}} p_{ij}y_{ij} + \sum_{(i,j)\in E} \mu\bar{y}_{ij} \tag{3.2}$$

Onde:

- y_{ij} é a variável binária que recebe valor 1 se os vértices i e j, onde $(i,j) \in \bar{E}$, tiverem a mesma cor e 0 caso contrário,
- p_{ij} é o custo da aresta complementar $(i,j) \in \bar{E}$,
- $\bar{y}_{ij}$ é a variável binária que recebe valor 1 se os vértices i e j, onde $(i,j) \in E$, tiverem a mesma cor e 0 caso contrário,
- μ é o valor da penalidade caso um conflito de cores ocorra.

3.4 Construção da Solução Inicial

Com o objetivo de gerar soluções iniciais, é utilizado um método de inicialização aleatória, que se destina a alocar aleatoriamente uma cor para cada um dos vértices do grafo. Este método de alocação aleatória pode gerar colorações inviáveis que são aceitas, e seu custo é calculado conforme apresentado na Equação 3.2. A partir desta coloração inicial C, uma ordem aleatória dos vértices é estabelecida e armazenada em uma lista. Para cada um dos vértices dessa lista, busca-se refinar a solução atual aplicando a vizinhança SVR de forma a escolher, para cada vértice, uma nova cor que reduza ao máximo a rigidez da solução. Este método é chamado de *Solução Inicial Refinada* (SIR), e está descrito no Algoritmo 7.

Algoritmo 7 Algoritmo de Construção da Solução Inicial Refinada

1: $C = \emptyset$
2: **para todo** $v \in V$ **faça**
3: C(v) = cor aleatória dentre as c possíveis cores
4: **fim para**
5: $Lista$ = ordem aleatória dos vértices V
6: **para todo** $v \in Lista$ **faça**
7: C = SVR(C,v)
8: **fim para**

3.5 Algoritmo Proposto

Este trabalho propõe-se a criar um algoritmo baseado na metaheurística ILS para tratar o PCR, chamado ILS-PCR. Na metaheuristica ILS (*Iterated Local Search*) [21], a cada iteração a solução corrente passa por uma etapa de diversificação, caracterizada pela aplicação de um mecanismo de peturbação, e uma etapa de intensificação, na qual a solução peturbada é melhorada por meio de uma busca local. Os passos de perturbação e intensificação podem fazer com que esta metaheurística explore bem o espaço de soluções, de forma a convergir para um ótimo global. O comportamento da metaheurística é ilustrado na Figura 3.6.

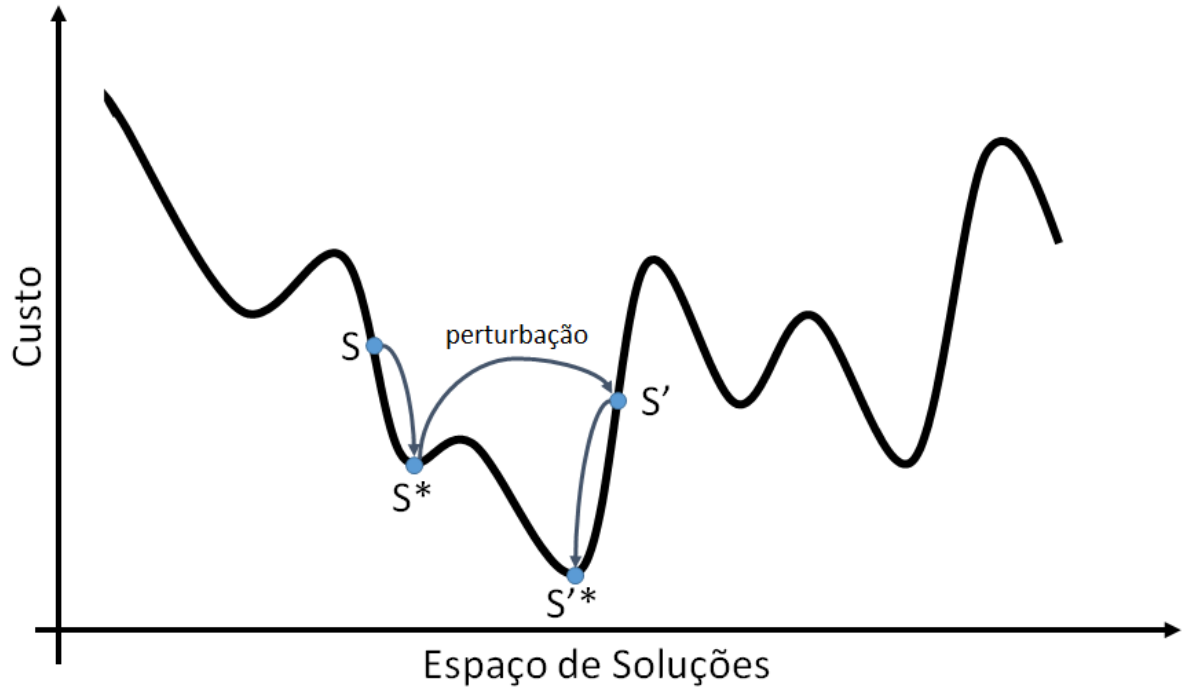

Figura 3.6: Mecanismo de perturbação (Adaptado de [21])

Em linhas gerais, o ILS, ilustrado pelo Algoritmo 8 e pela Figura 3.6, pode ser definido

da seguinte forma: uma solução inicial s_0 é gerada e possivelmente aprimorada pela aplicação de uma busca local, obtendo-se a solução s. A seguir, os seguintes passos são repetidos de forma iterativa: **(i)** perturba-se a solução s obtendo uma solução s', e **(ii)** aplica-se uma busca local na solução s' obtendo uma solução s'', um ótimo local perante as vizinhanças utilizadas. Caso s'' seja melhor que a solução corrente s (segundo um critério de aceitação que, neste trabalho, foi definido como uma redução na função de avaliação definida pela Equação 3.2), a solução s'' se torna a nova solução corrente. Esse processo é repetido até que um critério de parada seja satisfeito (normalmente um número máximo de iterações).

Algoritmo 8 Algoritmo Geral do ILS

```
1: s_0 = Gera solução inicial
2: s = BuscaLocal(s_0)
3: para Iter ≤ MaxIter faça
4:     s' = Perturbação(s)
5:     s'' = BuscaLocal (s')
6:     s = CriterioAceitacao(s'',s)
7: fim para
```

O mecanismo de perturbação tem um importante papel nessa metaheurística. As perturbações não podem ser demasiadamente pequenas ou grandes. Se são muito pequenas o processo de diversificação da solução é penalizado, ou seja, não consegue escapar do atual ótimo local. Do contrário, quando são muito grandes, a solução torna-se completamente aleatória.

Para este trabalho, foram feitas algumas modificações no algoritmo geral do ILS, conforme mostra o Algoritmo 9. Primeiramente, foi utilizado um mecanismo de *multi-start*, de forma a gerar uma certa quantidade de soluções iniciais, fazendo com que o algoritmo parta de soluções iniciais diferentes. Além disso, foi utilizado um mecanismo de perturbações por nível. A cada nível, tem-se uma certa quantidade $maxIter$ de iterações. Caso a solução não melhore naquele nível de perturbação em $maxIter$ iterações, passamos para um próximo nível, até chegar a um nível máximo. Cada pertubação consiste em escolher aleatoriamente uma cor para $2 + nvel$ vértices, conforme definido nas linhas 10 a 14 do Algoritmo 9. Por fim, tem-se um valor μ de penalidade para soluções inválidas, conforme definido na Equação 3.2. Esse valor de penalidade μ é definido dinamicamente da seguinte forma (linhas 16 a 20): caso a nova solução s'', seja viável, o valor de μ é decrescido em $\delta\%$ de seu valor original. Caso contrário, é aumentado em $\delta\%$. Essa penalidade dinâmica tem a função de fazer com que sejam priorizados as soluções válidas, colocando nelas um custo menor do que nas soluções inválidas.

Algoritmo 9 Algoritmo ILS-PCR

1: $s_{melhor} = \emptyset$
2: $\mu_{original} = 100000$
3: **para** $start = 1$ até $maxMultiStart$ **faça**
4: $\mu = \mu_{original}$
5: s_0 = Solução Inicial Refinada (SIR)
6: s = BuscaLocal(s_0)
7: **enquanto** $nvel = 1$ até $maxNivel$ **faça**
8: **enquanto** $iter = 1$ até $maxIter$ **faça**
9: $s' = s$
10: **para** $i = 1$ até $2 + nvel$ **faça**
11: selecione aleatoriamente um vértice v que ainda não foi escolhido
12: coloque uma nova cor em v definida aleatoriamente
13: atualize s'
14: **fim para**
15: s'' = BuscaLocal (s')
16: **se** Viável(s'') **então**
17: $\mu = \mu - \delta * \mu_{original}$
18: **senão**
19: $\mu = \mu + \delta * \mu_{original}$
20: **fim se**
21: **se** CriterioAceitacao(s'',s) **então**
22: $s = s''$
23: $iter = 0$
24: $nvel = 0$
25: **senão**
26: $iter++$
27: **fim se**
28: **fim enquanto**
29: $nvel++$
30: **fim enquanto**
31: **se** CriterioAceitacao(s,s_{melhor}) **então**
32: $s_{melhor} = s$
33: **fim se**
34: **fim para**

3.5.1 Variable Neighborhood Descent (VND)

O procedimento de busca local *Variable Neighborhood Descent* (VND), desenvolvido por Mladenović e Hansen [15], consiste em explorar exaustivamente o espaço de soluções por meio de trocas sistemáticas de estruturas de vizinhança, partindo de uma solução inicial C. No decorrer da busca, somente são aceitas as soluções de melhora em relação à solução C corrente. Na versão clássica, o VND efetua buscas por ótimos locais seguindo uma ordem pré-estabelecida de estruturas de vizinhança. Quando uma solução melhor é encontrada, o método retoma sua busca partindo da primeira estrutura de vizinhança.

Este procedimento baseia-se em três princípios fundamentais:

1. um ótimo local, com relação a uma dada estrutura de vizinhança, não corresponde necessariamente a um ótimo local com relação a uma outra estrutura de vizinhança;
2. o ótimo global corresponde a melhor solução passível de ser encontrada para todas as estruturas de vizinhança;
3. para muitos problemas, ótimos locais com relação a uma ou mais estruturas de vizinhança são relativamente próximas.

Os princípios destacados sugerem o uso de várias estruturas de vizinhança para resolução dos problemas de otimização. O objetivo agora é determinar o conjunto de estruturas que são utilizadas e a forma que são aplicadas. Neste trabalho, as estruturas de vizinhança demonstradas anteriormente são aplicadas na seguinte ordem: RSVR, MVR, ΘK-cycle e IG-K*-cycle. Além disso, para este trabalho, ao final da escolha do melhor vizinho é aplicada a vizinhança ESVR, caso o melhor vizinho melhore a solução atual, de forma a tentar melhorar ainda mais a solução encontrada. O VND é apresentado em forma de pseudo-código no Algoritmo 10.

3.5.2 Random Variable Neighborhood Descent (RVND)

O Algoritmo 11 apresenta o pseudocódigo do procedimento RVND [24], que consiste em uma adaptação do *Variable Neighborhood Descent* visto anteriormente. A diferença é que, agora, a lista de vizinhanças não precisa ter uma ordem pré-estabelecida. Ao invés disso, a ordem de visitação das vizinhanças é escolhida aleatoriamente, na inicialização do RNVD e em cada atualização da solução. Além disso, foi mostrado através de testes empíricos

Algoritmo 10 *Variable Neigborhood Descent* - VND

1: $N_l \leftarrow$ lista com as vizinhanças, com $l \in \{1, .., l_{max}\}$
2: $C \leftarrow$ Solução inicial
3: **enquanto** $ocorre - melhora$ **faça**
4: $l \leftarrow 1$
5: **enquanto** $l \leq l_{max}$ **faça**
6: $C' \leftarrow$ melhor vizinho da vizinhança N_l obtido a partir da solução C
7: **se** $\bar{R}(C') < \bar{R}(C)$ **então**
8: $C' \leftarrow$ ESVR(C')
9: $C \leftarrow C'$
10: $l \leftarrow 1$
11: **senão**
12: $l \leftarrow l + 1$
13: **fim se**
14: **fim enquanto**
15: **fim enquanto**

que, para alguns problemas, o RNVD é capaz de produzir melhores resultados quando comparado com o VND, utilizando uma abordagem baseado no ILS [24].

Algoritmo 11 *Random Variable Neigborhood Descent* - RVND

1: $N_l =$ lista aleatória com as vizinhanças, com $l \in \{1, .., l_{max}\}$
2: $C \leftarrow$ Solução inicial
3: **enquanto** $ocorre - melhora$ **faça**
4: $l \leftarrow 1$
5: **enquanto** $l \leq l_{max}$ **faça**
6: $C' \leftarrow$ melhor vizinho da vizinhança N_l obtido a partir da solução C
7: **se** $\bar{R}(C') < \bar{R}(C)$ **então**
8: $C' \leftarrow$ ESVR(C')
9: $C \leftarrow C'$
10: $l \leftarrow 1$
11: $N_l \leftarrow$ lista aleatória com as vizinhanças, com $l \in \{1, .., l_{max}\}$
12: **senão**
13: $l \leftarrow l + 1$
14: **fim se**
15: **fim enquanto**
16: **fim enquanto**

Capítulo 4

Experimentos Computacionais

O algoritmo proposto ILS-PCR foi desenvolvido na linguagem de programaçao C++, e todos os testes foram executados em um computador Intel Core 2 Duo E7500, 2.93GHz com 1,98 GB de RAM.

4.1 Descrição das Instâncias

Os experimentos computacionais foram realizados em um conjunto de instâncias apresentadas na Tabela 4.1. Essas instâncias foram obtidas de acordo com o artigo de Archetti et. al. [2] e divididas em cinco conjuntos de instâncias (57 instâncias no total): três criadas utilizando a técnica descrita em [30], com a densidade do grafo (i.e. a probabilidade de existir uma aresta entre dois vértices aleatórios) de 0.5, e duas adaptadas do conjunto de instâncias do DIMACS [17], criadas de acordo com o procedimento publicado em [2]. A Tabela 4.1 apresenta a configuração das instâncias utilizadas neste artigo. A coluna v informa o intervalo da quantidade de vértices, a coluna k informa o intervalo da quantidade de cores de cada conjunto de instâncias e a coluna *quantidade* indica quantas instâncias há no grupo. A lista completa de todas as instâncias está na Tabela A.1, com seus respectivos nomes, quantidade de vértices v, e quantidade máxima de cores k, bem como o valor do melhor limite inferior conhecido $\underline{z}$, e o tempo para provar sua otimalidade t, conforme visto em [2]. Segundo os autores, o tempo t somente é registrado caso o algoritmo prove que o limite inferior encontrado é ótimo dentro do limite de tempo de 2 horas (7200 segundos). Caso não consiga, um '-' é colocado no lugar do tempo, indicando que o resultado ótimo não foi encontrado, por superar o tempo limite ou por estouro de memória.

#Conjunto	Conjunto	v	k	#Instâncias
1	$DIMACS1$	{11..128}	{6..110}	14
2	$DIMACS2$	{11..128}	{8..84}	18
3	$Archetti40$	{40}	{14, 15}	5
4	$Archetti90$	{90}	{30, 31}	10
5	$Archetti120$	{120}	{40, 41}	10

Tabela 4.1: Tabela de instâncias

4.2 Resultados Detalhados

Foram realizadas 5 execuções do algoritmo heurístico para cada uma das 57 instâncias. O algoritmo ILS-PCR necessita de um conjunto de parâmetros, conforme descrito no Capítulo 3. Estes parâmetros foram determinados empiricamente conforme a Tabela 4.2.

Parâmetro	Valor	Descrição
r	3	Tamanho da lista da vizinhança RSVR
α	2	Tamanho da lista da vizinhança MVR
k	5	Quantidade máxima de vértices no ciclo da vizinhança IG-K*-cycle
θ	100	Tamanho máximo da lista de candidatos da vizinhança IG-K*-cycle
K	5	Tamanho da lista de vértices da vizinhança θK-cycle
Θ	10	Quantidade de ciclos produzidos na vizinhança θK-cycle
$maxMultiStart$	5	Parâmetro multi-start para o ILS-PCR
$maxNivel$	3	Quantidade de níveis de pertubação para o ILS-PCR
$maxIter$	30	Quantidade de iterações por nível para o ILS-PCR
μ	100000	Valor de penalidade para soluções inválidas
δ	20%	Valor de ajuste da penalidade μ

Tabela 4.2: Parâmetros do ILS-PCR

Definidos esses parâmetros, primeiramente foi realizada uma comparação das buscas locais RVND e VND, descritas nesse trabalho, de forma a escolhar um algoritmo de referência para as demais comparações. Em seguida, o ILS-PCR de referência foi comparado com o algoritmo Busca Tabu de Wang e Chu [28] e com o algoritmo *Branch-and-price* proposto por Archetti et. al [2]. Em seguida, foi feito um estudo comparando o comportamento do ILS-PCR, para diferentes valores de $maxNivel$, alterando assim a quantidade de peturbações feitas por iteração do algoritmo.

4.2.1 Comparação entre RVND e VND

Para comparar a eficiência entre as duas buscas locais descritas neste trabalho, foram executados os algoritmos ILS-PCR-RVND (correspondente ao ILS-PCR usando a busca

local RVND) e ILS-PCR-VND (correspondente ao ILS-PCR usando a busca local VND), com os parâmetros definidos na Tabela 4.2. Os resultados detalhados estão na Tabela A.3, onde $\bar{z}$ representa o valor médio da função objetivo, σ_z o desvio padrão da função objetivo, $C_v\%$ o coeficiente de variação obtido através da razão entre $\bar{z}$ e σ_z, $GAP\%$ o *gap* entre $\bar{z}$ e o limite inferior $\underline{z}$, calculado utilizando a fórmula $GAP\% = 100 * \frac{\bar{z}-\underline{z}}{\underline{z}}$, $\bar{t}$ o tempo médio e σ_t o desvio padrão do tempo. O tempo limite foi de 7200 segundos. Os testes que chegaram nesse limite de tempo estão registrados com *. Não é conhecido o valor ótimo da instância *myciel*6 do conjunto 2, portanto seu $GAP\%$ também foi registrado como *. Todos os resultados encontrados foram soluções viáveis.

De posse dos dados da Tabela A.3, foram utilizados alguns testes estatísticos utilizando o software R [26] para verificar se algum dos algoritmos apresentava desempenho superior. Primeiramente, foi executado o teste de Shapiro-Wilk [27] para verificar se os dados seguiam distribuição normal. O resultado do teste está na Tabela 4.3.

Tabela 4.3: Teste de Shapiro-Wilk para normalidade

Algoritmo	Estatística de Teste	P-valor
ILS-PCR-RVND	W = 0,5146	0,000
ILS-PCR-VND	W = 0,5145	0,000

De acordo com os dados apresentados na Tabela 4.3 pode-se concluir, ao nível de 5% de significância, que as respostas médias de ambos os algoritmos não seguiam distribuição normal. Portanto, a comparação entre os mesmos foi feita com a utilização de um teste não paramétrico: o teste de Mann-Whitney [22]. O objetivo deste teste foi verificar se os dois algoritmos pertenciam a um mesmo grupo (população). Caso não pertençam, pode-se concluir que as médias em estudo não são estatisticamente iguais. O resultado deste teste está na Tabela 4.4.

Tabela 4.4: Teste de Mann-Whitney para resposta média: ILS-PCR-RVND e ILS-PCR-VND

Algoritmos	Estatística de Teste	P-valor
ILS-PCR-RVND & ILS-PCR-VND	W = 1627,5	0,9887

Pelos resultados apresentados na Tabela 4.4 pode-se concluir, ao nível de 5% de signficância, que as respostas médias de ambos algoritmos são estatisticamente iguais, portanto, não existem razões estatísticas para afirmar que um dos algoritmos é mais eficiente no fornecimento de respostas médias. Assim sendo, foi refeito o teste de Mann-Whitney para o tempo médio de execução do algoritmo. O resultado deste teste está na Tabela 4.5.

Tabela 4.5: Teste de Mann-Whitney para tempo médio: ILS-PCR-RVND e ILS-PCR-VND

Algoritmos	Estatística de Teste	P-valor
ILS-PCR-RVND & ILS-PCR-VND	W = 1612,5	0,7979

Pelos resultados apresentados na Tabela 4.5 pode-se concluir, ao nível de 5% de signficância, que o tempo médio de ambos algoritmos é estatisticamente igual, portanto, não existem razões estatísticas para afirmar que um dos algoritmos é superior ao outro em relação ao tempo médio decorrido para encontrar a solução em todos os casos. Portanto, com base nos resultados apresentados nas Tabelas 4.4 e 4.5, não há nenhuma evidência estatística que comprove que uma busca local seja superior a outra.

Desse modo, para escolher o algoritmo de referência, foi investigado qual possuía um menor coeficiente de variação na maioria das vezes. Para isso, foi realizada a diferença entre o $C_v\%$ do ILS-PCR-VND e o $C_v\%$ do ILS-PCR-RVND para cada instância. Foi verificado que o ILS-PCR-VND tem um maior coeficiente de variação em 22 instâncias, enquanto que o ILS-PCR-RVND tem menor coeficiente em 19 instâncias (ocorreu empate em 16 instâncias). Portanto, na maioria das vezes, os resultados encontrados pelo ILS-PCR-RVND variam menos ou de forma igual aos resultados encontrados pelo ILS-PCR-VND.

De posse das análises realizadas, o algoritmo de referência a ser utilizado nas demais comparações foi o ILS-PCR-RVND.

4.2.2 Comparação com Busca Tabu

Para medir a eficiência do ILS-PCR-RVND, foi utilizada a metaheurística híbrida *Busca Tabu* (TS) desenvolvida por [28], por ter sido a que apresenta os melhores resultados na literatura. Porém, não foi possível obter as instâncias utilizadas pelos autores pois, segundo eles, se tratavam de instâncias baseadas em dados restritos (funcionamento do *Hong Kong International Airport*). Os autores de [28] também se negaram a fornecer a implementação do método TS, e a possibilidade de executarem as instâncias aqui obtidas para uma comparação de resultados. Com isso, foi necessário realizar a implementação do algoritmo conforme descrito em [28].

Os parâmetros utilizados foram os mesmos propostos pelos autores em seu artigo: $maxTenure = 6$, $tMax = 5000$, $r = 5$, $K = 4$, $\theta = 100$. Foram realizadas 5 execuções do algoritmo para cada uma das 57 instâncias. Os resultados detalhados estão na Tabela

A.2, onde $\bar{z}$ representa o valor médio da função objetivo, σ_z o desvio padrão da função objetivo, $C_v\%$ o coeficiente de variação, $GAP\%$ o *gap* entre $\bar{z}$ e o limite inferior $\underline{z}$, $\bar{t}$ o tempo médio e σ_t o desvio padrão do tempo. Todas as soluções encontradas foram viáveis.

Analisando os dados das Tabelas A.3 e A.2, pode-se perceber que o ILS-PCR-RVND possuía uma melhor solução média em 56 das 57 instâncias, sendo que apenas em 1 instância (myciel3) apresentam resultados iguais. Além disso, como é possível ver nos gráficos das Figuras 4.2 e 4.3, o ILS-PCR-RVND apresentou uma menor variação nos resultados médios e, exceto nas instâncias DSJC125.5 e queen9_9 do conjunto de instâncias DIMACS2, o ILS-PCR-RVND apresentou um *gap* menor. Um fato interessante é que, tanto o *gap* quanto o coeficiente de variação, para o conjunto de instâncias de Archetti, é muito maior no Busca Tabu do que no ILS-PCR-RVND.

Como a diferença de tempo entre os algoritmos Busca Tabu e ILS-PCR-RVND foi substancial em várias instâncias, foram realizados novos testes com limites de tempo mais comparáveis. Os novos limites de tempo foram de 15, 30 e 60 segundos, e os algoritmos serão chamados neste trabalho de ILS-PCR-RVND-15, ILS-PCR-RVND-30 e ILS-PCR-RVND-60, respectivamente. Os resultados detalhados estão representados nas Tabelas A.4, A.5 e A.6, respectivamente. Nestas tabelas percebe-se que, mesmo limitando o tempo para um valor muito baixo (15 segundos), o algoritmo ILS-PCR-RNVD consegue encontrar soluções melhores que a do Busca Tabu, exceto para a instância #1_*myciel*3, que apresentou um resultado médio igual. Isso fica mais claro no gráfico da Figura 4.2, que mostra o GAP entre a solução média encontrada nos algoritmos Busca Tabu, ILS-PCR-RVND, ILS-PCR-RVND-15, ILS-PCR-RVND-30 e ILS-PCR-RVND-60. Com relação ao coeficiente de variação, o que se nota é uma grande variação entre as soluções do grupo 5 de instâncias, conforme indica o gráfico da Figura 4.3, porém ainda muito inferior às apresentadas pelo Busca Tabu.

4.2.3 Comparação com *Branch-and-Price*

Para este trabalho, foram utilizados para comparação os resultados do algoritmo *Branch-and-Price*, proposto por Archetti et al. [2]. Em seu artigo, os autores disseram que o algoritmo foi implementado em C++, e todos os experimentos foram executados em um processador AMD Athlon 64 X2 Dual Core 5600+, 2,89GH com 3,37GB de RAM. O tempo máximo de execução foi de 2 horas (7200 segundos). Os resultados descritos no artigo de Archetti et al. [2], conforme dito anteriormente, foram utilizados como referência neste trabalho.

Pelos resultados apresentados na Tabela A.3, foi alcançado um *gap* médio de 3,33% para o método ILS-PCR-RVND, desconsiderando os casos em que o *gap* foi igual a zero (i.e. a solução ótima foi encontrada). Além disso, é possível notar que em todas as instâncias em que o *Branch-and-Price* não conseguiu encontrar uma solução ótima por estouro de tempo e/ou memória, o ILS-PCR-RVND conseguiu encontrar uma boa solução, com um *gap* baixo em comparação à solução de Archetti et al [2]. Os maiores *gaps* alcançados pelo método ILS-PCR-RVND foram para o conjunto de instâncias 4 e 5, porém com um tempo médio de execução muito inferior. Em poucas instâncias (11) o algoritmo *Branch-and-Price* encontrou uma solução com um tempo inferior ao ILS-PCR-RVND.

Portanto, pode-se dizer que o algoritmo ILS-PCR-RVND encontra soluções ótimas (ou quase-ótimas) em um tempo muito inferior ao método *Branch-and-Price*. Essa afirmação pode ser visualizada pelo *boxplot* da Figura 4.1, que ilustra uma comparação entre resposta média e tempo médio de execução de cada algoritmo.

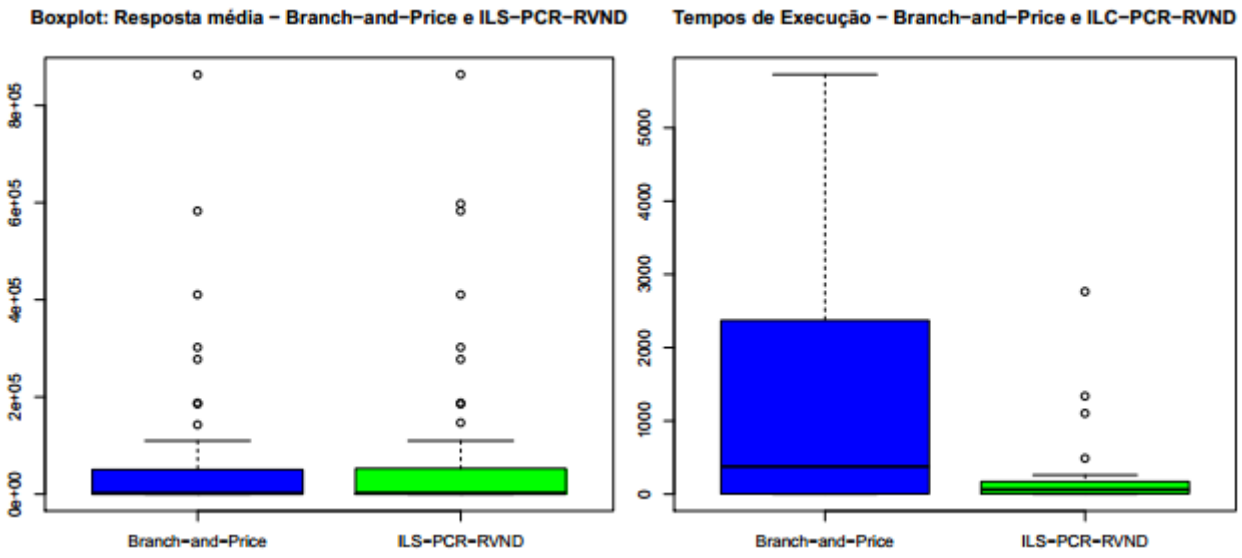

Figura 4.1: Boxplot: ILS-PCR-RVND e *Branch-and-Price*

Pelo *boxplot* ilustrado na Figura 4.1, podemos levantar a hipótese de que os dois algoritmos são estatisticamente iguais em relação a $\bar{z}$. Com base nos dados apresentados nas Tabelas A.3 e A.1, os dois algoritmos foram comparados utilizando o teste de Mann-Whitney, de forma a confirmar esta hipótese. O resultado do teste está descrito na Tabela 4.6.

Tabela 4.6: Teste de Mann-Whitney: $\bar{z}$ - Branch and Price e ILS-PCR-RVND

Algorítimos	Estatística de Teste	P-valor
ILS-PCR-RVND & Branch and Price	W = 1492	0,55230

De acordo com o resultado do teste (Tabela 4.6), ao nível de 5% de significância, pode-se concluir que não existem diferenças estatisticamente significativas entre as respostas médias dos algoritmos, o que confirma a hipótese levantada de que os dois algoritmos são estatisticamente iguais em relação à $\bar{z}$.

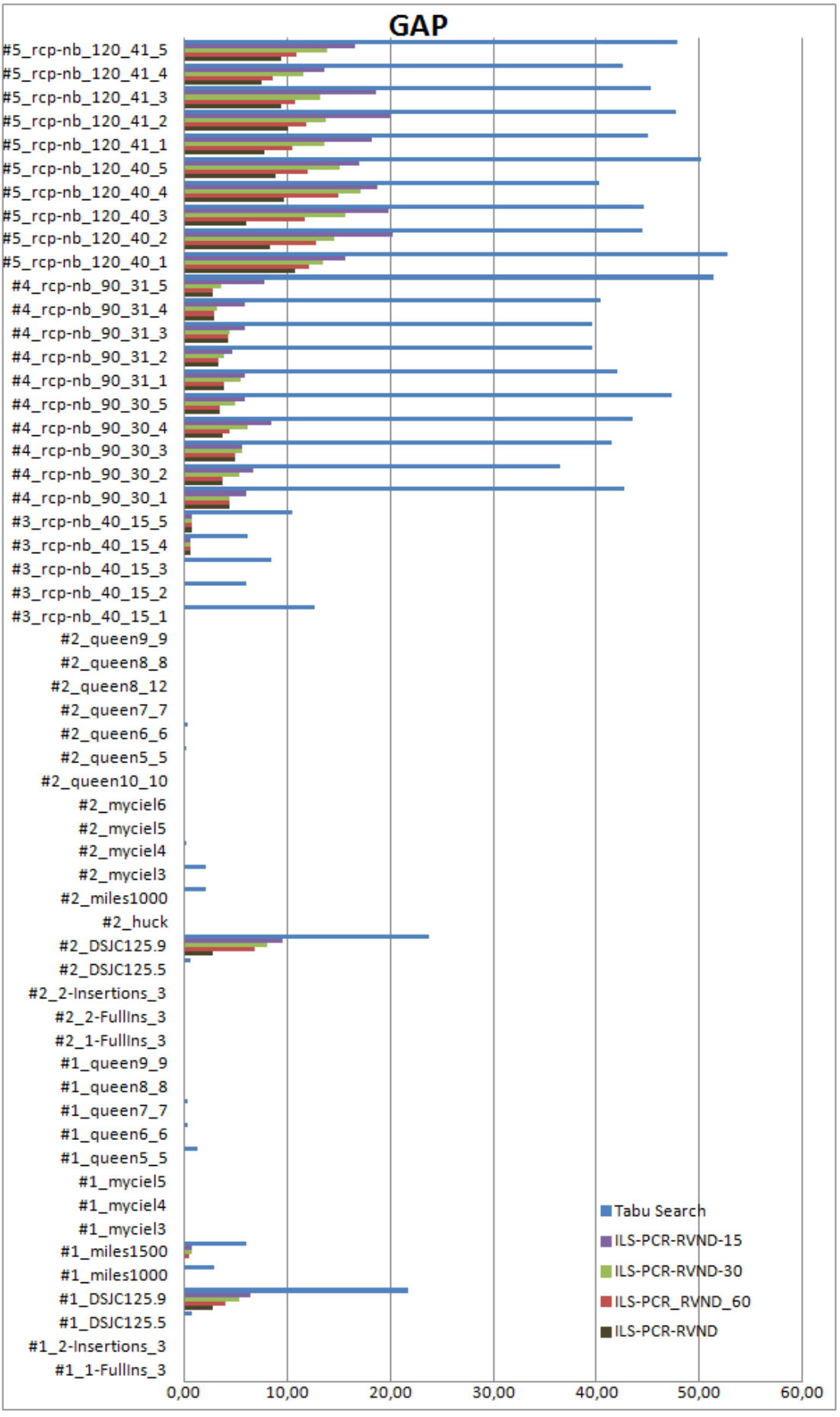

Figura 4.2: GAPs do Busca Tabu e os ILS-PCR-RVND

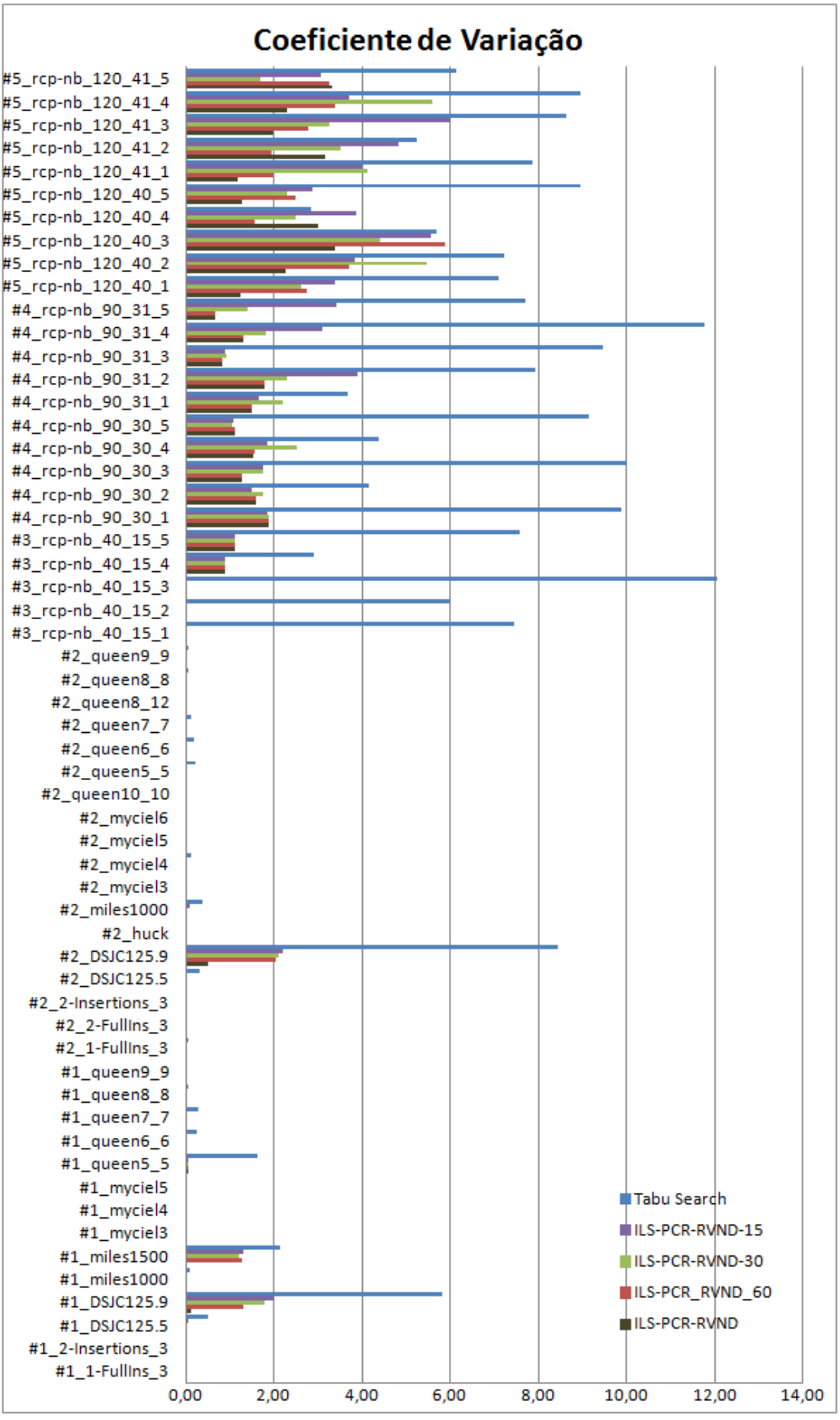

Figura 4.3: Coeficientes de variação do Busca Tabu e os ILS-PCR-RVND

Capítulo 5

Conclusão

O Problema da Coloração Robusta (PCR) é recente e ainda relativamente pouco explorado na literatura, porém com poder de proporcionar uma nova perspectiva ao clássico Problema da Coloração Mínima, abrindo espaço para a abordagem de novas aplicações práticas, devido ao fato de considerar toda a topologia do grafo e não apenas a quantidade de cores utilizada.

Neste trabalho foi proposta uma nova formulação matemática para o problema PCR, baseada no modelo dos representantes, e um algoritmo baseado na metaheurística *Iterated Local Search* (ILS) utilizando dois tipos diferentes de buscas locais: a *Random Variable Neighborhood Search* (RNVD) e a *Variable Neighborhood Search* (VND). Também foram propostas duas novas vizinhanças, baseando-se nas já existentes na literatura: a *Multi-Vertex Recoloring* e a ΘK*-Cycle*. Testes mostraram que as duas buscas locais são estatisticamente iguais, mas com uma ligeira vantagem para a busca local RVND.

O algoritmo ILS-PCR-RVND mostrou resultados eficientes e robustos para solucionar o PCR, mesmo com um baixo limite de tempo. Isto pôde ser verificado quando comparado com soluções já existentes na literatura, sendo comparável estatísticamente com um algoritmo exato, porém apresentando um desempenho muito superior para a maioria dos testes realizados. Mesmo quando limitado em seu tempo de execução, o ILS-PCR-RVND conseguiu resultados competitivos.

5.1 Trabalhos Futuros

Como trabalhos futuros propõe-se a criação de um algoritmo híbrido, incorporando o algoritmo ILS-PCR em um modelo de programação matemática utilizando as formulações

apresentadas neste trabalho, de forma a acelerar a obtenção de limitantes superiores e melhorar os resultados para o problema.

Além disso, sugere-se uma maior investigação dos parâmetros das vizinhanças e do ILS-PCR, de forma a averiguar o impacto de cada parâmetro no algoritmo. Também pretende-se expandir o uso das vizinhanças geradas para a resolução do PCR em outros problemas de coloração, como o clássico Problema da Coloração Mínima.

Referências

[1] Aardal, K. I.; Hoesel, S. P. M. V.; Koster, A. M. C. A.; Mannino, C.; Sassano, A. Models and solution techniques for frequency assignment problems. In *Annals of Operations Research* (2007), vol. 153, pp. 79–129.

[2] Archetti, C.; Bianchessi, N.; Hertz, A. A branch-and-price algorithm for the robust graph coloring problem. *Bibliographie du Québec 45*, 3 (Março 2012).

[3] Barnhart, C.; Johnson, E. L.; Nemhauser, G. L.; Savelsbergh, M. W. P.; Vance, P. H. Branch-and-price: Column generation for solving huge integer programs. *Operations Research 46*, 3 (1998), pp. 316–329.

[4] Bellman, R. *Dynamic Programming*. Dover Publications, Mar. 2003.

[5] Blum, C.; Roli, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv. 35*, 3 (sep 2003), 268–308.

[6] Corrêa, R. C.; Campelô, M.; Frota, Y. A. M. Cliques, holes and the vertex coloring polytope. *Information Processing Letters 89* (2004), 159–164.

[7] Desaulniers, G.; Desrosiers, J.; Solomon, M. *Column Generation*. GERAD 25th anniversary series. Springer, 2006.

[8] Dorigo, M. *Optimization, Learning and Natural Algorithms*. Tese de Doutorado, Politecnico di Milano, Italy, 1992.

[9] Galinier, P.; Hao, J.-K. Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization 3*, 4 (1999), 379–397.

[10] Garey, M. R.; Johnson, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[11] Glover, F. Heuristics for integer programming using surrogate constraints. *Decision Sciences 8*, 1 (1977), 156–166.

[12] Glover, F. Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res. 13*, 5 (May 1986), 533–549.

[13] Gutierrez-Andrade M.A., Lara-Velazquez P., d. l. C. S.-S. A new simulated annealing algorithm for the robust coloring problem. *Journal of Industrial Engineering International 3*, 5 (2007), 27–32.

[14] Gutierrez-Andrade M.A., Lara-Velazquez P., L.-B. R. R.-R. J. Heuristic for the robust coloring problem. *Revista de Matemática: Teoría y Aplicaciones 18*, 1, 137–147.

[15] Hansen, P.; Mladenovic, N. *Handbook of Metaheuristics*. Kluwer Academic Publishers, 2002, ch. Variable Neighborhood Search, pp. 145–184.

[16] Holland, J. H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992.

[17] Johnson, D. J.; Trick, M. A., Eds. *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, Workshop, October 11-13, 1993*. American Mathematical Society, Boston, MA, USA, 1996.

[18] Kirkpatrick, S.; Gelatt, C. D.; Vecchi, M. P. Optimization by simulated annealing. *Science 220*, 4598 (1983), 671–680.

[19] Laureano-Cruces, A. L.; Ramírez-Rodríguez, J.; Hernández-González, D. E.; Méndez-Gurrola, I. I. An ant algorithm for the robust coloring problem. In *ICGST Conference on Artificial Intelligence and Machine Learning, AIML-11 Dubai-11 Conference* (4 2011), ICGST, pp. 57–60.

[20] Lim, A.; Wang, F. Robust graph coloring for uncertain supply chain management. In *HICSS '05. Proceedings of the 38th Annual Hawaii International Conference on System Sciences, 2005.* (2005), IEEE Computer Society.

[21] Lourenço, H. R.; Martin, O. C.; Stützle, T. *Handbook of Metaheuristics*. Kluwer Academic Publishers, Norwell, MA, 2002, ch. Iterated Local Search, pp. 321–353.

[22] Mann, H. B.; Whitney, D. R. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics 18*, 1 (03 1947), 50–60.

[23] Nemhauser, G. L.; Wolsey, L. A. *Integer and combinatorial optimization*. Wiley-Interscience, New York, NY, USA, 1988.

[24] Penna, P. H. V.; Subramanian, A.; Ochi, L. S. An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics 19*, 2 (2013), 201–232.

[25] P.M. Pardalos, T. Mavridou, J. X. The graph coloring problem: A bibliographic survey. *Handbook of Combinatorial Optimization 2* (1998), 331–395.

[26] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.

[27] Shapiro, S. S.; Wilk, M. B. An analysis of variance test for normality (complete samples). *Biometrika 52*, 3/4 (Dec. 1965), 591–611.

[28] Wang, F.; Xu, Z. Metaheuristics for robust graph coloring. *Journal of Heuristics* (July 2011), 1–20.

[29] y Miguel Angel Gutiérrez Andrade y Javier Ramírez Rodríguez y Rafael López Bracho, P. L. V. Un algoritmo evolutivo para resolver el problema de coloración robusta. *Revista de Matemática: Teoría y Aplicaciones 12*, 1-2 (2005).

[30] Yanez, J.; Ramírez, J. The robust coloring problem. *European Journal of Operational Research 148*, 3 (2003), 546–558.

APÊNDICE A – Resultados do ILS-PCR

Tabela A.1: Instâncias Detalhadas

Instância	Conjunto	V	K	z	t
1-FullIns_3	DIMACS1	30	6	13292	100,66
2-Insertions_3	DIMACS1	37	6	32397	-
DSJC125.5	DIMACS1	125	26	863172	-
DSJC125.9	DIMACS1	125	66	142721	212,78
miles1000	DIMACS1	128	63	187394	2597,28
miles1500	DIMACS1	128	110	4284	46,48
myciel3	DIMACS1	11	6	110	0,16
myciel4	DIMACS1	23	8	2600	2,08
myciel5	DIMACS1	47	9	52829	-
queen5_5	DIMACS1	25	8	3850	2,98
queen6_6	DIMACS1	36	11	12061	4,16
queen7_7	DIMACS1	49	11	48002	2785
queen8_8	DIMACS1	64	14	109796	-
queen9_9	DIMACS1	81	15	277416	-
1-FullIns_3	DIMACS2	30	8	8787	95,38
2-FullIns_3	DIMACS2	52	10	70830	-
2-Insertions_3	DIMACS2	37	8	22101	2075,91
DSJC125.5	DIMACS2	125	34	582561	-
DSJC125.9	DIMACS2	125	88	36351	114,16
huck	DIMACS2	74	22	106103	-
miles1000	DIMACS2	128	84	58768	1068,44
myciel3	DIMACS2	11	8	28	0,03
myciel4	DIMACS2	23	10	1648	0,84
myciel5	DIMACS2	47	12	35126	4066,59
myciel6	DIMACS2	95	14	-	-
queen10_10	DIMACS2	100	22	410430	-
queen5_5	DIMACS2	25	10	2521	0,99
queen6_6	DIMACS2	36	14	7741	3,28
queen7_7	DIMACS2	49	14	33383	98,50
queen8_12	DIMACS2	96	24	301864	-
queen8_8	DIMACS2	64	18	75460	-
queen9_9	DIMACS2	81	20	185506	-
rcp-nb_40_15_1	Archetti40	40	15	5,367	2,33
rcp-nb_40_15_2	Archetti40	40	15	4,819	3,58
rcp-nb_40_15_3	Archetti40	40	15	5,224	11,50
rcp-nb_40_15_4	Archetti40	40	15	5,544	10,74
rcp-nb_40_15_5	Archetti40	40	15	4,939	10,53
rcp-nb_90_30_1	Archetti90	90	30	10,785	-
rcp-nb_90_30_2	Archetti90	90	30	10,316	1208,86
rcp-nb_90_30_3	Archetti90	90	30	10,53	2641,53
rcp-nb_90_30_4	Archetti90	90	30	10,113	2134,30
rcp-nb_90_30_5	Archetti90	90	30	9,481	2252,52
rcp-nb_90_31_1	Archetti90	90	31	9,263	4183,66
rcp-nb_90_31_2	Archetti90	90	31	8,924	1695,75
rcp-nb_90_31_3	Archetti90	90	31	8,966	540,47
rcp-nb_90_31_4	Archetti90	90	31	8,949	1598,47
rcp-nb_90_31_5	Archetti90	90	31	7,667	1493,95
rcp-nb_120_40_1	Archetti120	120	40	10,343	5544,30
rcp-nb_120_40_2	Archetti120	120	40	11,078	-
rcp-nb_120_40_3	Archetti120	120	40	10,819	5727,91
rcp-nb_120_40_4	Archetti120	120	40	10,886	2490,19
rcp-nb_120_40_5	Archetti120	120	40	11,933	4553,19
rcp-nb_120_41_1	Archetti120	120	41	10,184	4780,66
rcp-nb_120_41_2	Archetti120	120	41	10,661	-
rcp-nb_120_41_3	Archetti120	120	41	9,430	-
rcp-nb_120_41_4	Archetti120	120	41	9,387	1431,95
rcp-nb_120_41_5	Archetti120	120	41	9,700	-

Tabela A.2: Resultados: Busca Tabu

#Conjunto	Instância	$\bar{z}$	σ_z	C_v%	GAP%	$\bar{t}$	σ_t
1	1-FullIns_3	13295,80	3,58	0,03	0,03	0,48	0,02
1	2-Insertions_3	32397,60	0,45	0,00	0,00	0,70	0,03
1	DSJC125.5	870188,20	4343,40	0,50	0,81	25,31	0,42
1	DSJC125.9	173832,80	10126,87	5,83	21,80	61,18	0,85
1	miles1000	192853,60	150,76	0,08	2,91	62,43	0,94
1	miles1500	4542,60	96,74	2,13	6,04	105,96	1,66
1	myciel3	110,00	0,00	0,00	0,00	0,10	0,01
1	myciel4	2600,80	0,45	0,02	0,03	0,40	0,02
1	myciel5	52834,00	5,86	0,01	0,01	1,50	0,04
1	queen5_5	3898,80	63,65	1,63	1,27	0,47	0,03
1	queen6_6	12099,80	28,48	0,24	0,32	1,15	0,07
1	queen7_7	48177,20	135,51	0,28	0,36	1,97	0,09
1	queen8_8	109920,80	75,32	0,07	0,11	3,93	0,07
1	queen9_9	277648,60	91,52	0,03	0,08	6,61	0,16
2	1-FullIns_3	8790,40	4,76	0,05	0,04	0,62	0,02
2	2-FullIns_3	70833,20	3,65	0,01	0,00	1,98	0,08
2	2-Insertions_3	22102,20	1,22	0,01	0,01	0,95	0,04
2	DSJC125.5	585956,60	1875,36	0,32	0,58	31,99	0,48
2	DSJC125.9	44997,40	3796,98	8,44	23,79	80,30	1,07
2	huck	106180,40	22,10	0,02	0,07	7,73	0,10
2	miles1000	60022,60	231,63	0,39	2,13	82,32	1,41
2	myciel3	28,60	0,00	0,00	2,14	0,13	0,01
2	myciel4	1650,80	1,87	0,11	0,17	0,46	0,02
2	myciel5	35172,80	5,76	0,02	0,05	1,90	0,03
2	myciel6	597492,80	21,30	0,00	*	8,31	0,14
2	queen10_10	410880,60	109,70	0,03	0,11	13,73	0,07
2	queen5_5	2527,80	5,63	0,22	0,27	0,55	0,03
2	queen6_6	7770,80	14,02	0,18	0,38	1,41	0,08
2	queen7_7	33436,20	36,60	0,11	0,16	2,37	0,10
2	queen8_12	302066,40	62,54	0,02	0,07	14,00	0,37
2	queen8_8	75550,20	30,17	0,04	0,12	4,87	0,10
2	queen9_9	185728,20	89,59	0,05	0,12	8,42	0,19
3	rcp-nb_40_15_1	6,047	0,450	7,45	12,66	1,736	0,078
3	rcp-nb_40_15_2	5,112	0,306	5,99	6,08	1,726	0,071
3	rcp-nb_40_15_3	5,665	0,683	12,06	8,43	1,788	0,079
3	rcp-nb_40_15_4	5,887	0,172	2,92	6,19	1,734	0,077
3	rcp-nb_40_15_5	5,459	0,413	7,56	10,52	1,746	0,068
4	rcp-nb_90_30_1	15,387	1,520	9,88	42,67	15,132	0,253
4	rcp-nb_90_30_2	14,077	0,583	4,14	36,46	15,154	0,206
4	rcp-nb_90_30_3	14,904	1,490	9,99	41,54	15,102	0,274
4	rcp-nb_90_30_4	14,512	0,633	4,36	43,50	15,076	0,304
4	rcp-nb_90_30_5	13,963	1,275	9,13	47,27	14,818	0,216
4	rcp-nb_90_31_1	13,156	0,484	3,68	42,02	15,420	0,193
4	rcp-nb_90_31_2	12,460	0,989	7,93	39,62	15,440	0,188
4	rcp-nb_90_31_3	12,520	1,184	9,46	39,64	15,412	0,240
4	rcp-nb_90_31_4	12,567	1,478	11,76	40,43	15,698	0,294
4	rcp-nb_90_31_5	11,607	0,894	7,70	51,39	15,672	0,282
5	rcp-nb_120_40_1	15,792	1,119	7,08	52,69	35,208	0,182
5	rcp-nb_120_40_2	15,998	1,156	7,23	44,41	35,602	0,708
5	rcp-nb_120_40_3	15,649	0,892	5,70	44,64	35,250	0,155
5	rcp-nb_120_40_4	15,270	0,436	2,85	40,27	35,156	0,356
5	rcp-nb_120_40_5	17,913	1,601	8,94	50,11	35,230	0,414
5	rcp-nb_120_41_1	14,772	1,161	7,86	45,05	36,082	0,701
5	rcp-nb_120_41_2	15,749	0,825	5,24	47,73	35,568	0,391
5	rcp-nb_120_41_3	13,700	1,181	8,62	45,28	36,078	0,711
5	rcp-nb_120_41_4	13,385	1,200	8,96	42,59	35,746	0,178
5	rcp-nb_120_41_5	14,336	0,881	6,14	47,79	35,530	0,438
	Médias C_v% & GAP			0	0		

Tabela A.3: Resultados: ILS-PCR com $maxNivel = 3$, $t_{max} = 7200$

#Conjunto	Instância	ILS-PCR-RVND						ILS-PCR-VND					
		$\bar{z}$	σ_z	$C_v\%$	$GAP\%$	$\bar{t}$	σ_t	$\bar{z}$	σ_z	$C_v\%$	$GAP\%$	$\bar{t}$	σ_t
1	1-FullIns_3	13292,00	0,00	0,00	0,00	2,41	0,11	13292,00	0,00	0,00	0,00	2,67	0,09
1	2-Insertions_3	32397,00	0,00	0,00	0,05	3,59	0,31	32397,00	0,00	0,00	0,00	3,36	0,04
1	DSJC125.5	863632,20	117,94	0,01	2,83	245,01	43,40	863578,00	119,90	0,01	0,05	284,40	23,55
1	DSJC125.9	146764,20	190,02	0,13	0,01	260,66	60,26	146936,80	161,39	0,11	2,95	289,71	23,86
1	miles1000	187412,60	7,44	0,00	0,00	1101,97	117,88	187425,40	5,81	0,00	0,02	1311,38	180,14
1	miles1500	4284,00	0,00	0,00	0,00	*	*	4284,00	0,00	0,00	0,00	*	*
1	myciel3	110,00	0,00	0,00	0,00	0,39	0,01	110,00	0,00	0,00	0,00	0,36	0,01
1	myciel4	2600,00	0,00	0,00	0,00	1,63	0,15	2600,00	0,00	0,00	0,00	1,90	0,14
1	myciel5	52829,00	0,00	0,00	0,02	11,16	1,07	52829,00	0,00	0,00	0,00	12,16	1,22
1	queen5_5	3850,80	1,79	0,05	0,01	2,07	0,24	3852,40	2,19	0,06	0,06	1,98	0,19
1	queen6_6	12062,60	0,89	0,01	0,02	7,51	1,17	12063,60	1,14	0,01	0,02	9,15	1,15
1	queen7_7	48013,60	4,51	0,01	0,00	14,96	2,60	48015,20	6,87	0,01	0,03	14,11	1,53
1	queen8_8	109801,20	2,17	0,00	0,00	41,65	7,53	109801,60	2,30	0,00	0,01	49,48	2,47
1	queen9_9	277425,40	2,88	0,00	0,00	81,83	9,42	277428,80	2,77	0,00	0,00	76,87	8,71
2	1-FullIns_3	8787,00	0,00	0,00	0,00	2,86	0,11	8787,00	0,00	0,00	0,00	3,28	0,21
2	2-FullIns_3	70830,00	0,00	0,00	0,00	11,34	0,30	70830,00	0,00	0,00	0,00	11,83	0,50
2	2-Insertions_3	22101,00	0,00	0,00	0,03	3,92	0,19	22101,00	0,00	0,00	0,00	4,13	0,09
2	DSJC125.5	582710,80	14,24	0,00	2,83	487,91	59,97	582690,80	17,37	0,00	0,02	585,13	68,84
2	DSJC125.9	37378,00	182,64	0,49	0,00	1337,22	358,65	37347,40	257,04	0,69	2,74	1389,89	319,79

Continua na próxima página

Tabela A.3: Resultados: ILS-PCR com $maxNivel = 3$, $t_{max} = 7200$

#Conjunto	Instância	ILS-PCR-RVND						ILS-PCR-VND					
		$\bar{z}$	σ_z	$C_v\%$	$GAP\%$	$\bar{t}$	σ_t	$\bar{z}$	σ_z	$C_v\%$	$GAP\%$	$\bar{t}$	σ_t
2	huck	106103,00	0,00	0,00	0,01	53,80	3,42	106103,00	0,00	0,00	0,00	61,92	7,65
2	miles1000	58772,20	3,49	0,01	0,00	2763,65	427,55	58770,60	2,30	0,00	0,00	2327,70	1952,87
2	myciel3	28,00	0,00	0,00	0,00	0,41	0,01	28,00	0,00	0,00	0,00	0,36	0,00
2	myciel4	1648,00	0,00	0,00	0,00	1,88	0,08	1648,00	0,00	0,00	0,00	2,31	0,07
2	myciel5	35156,80	0,45	0,00	*	13,20	0,67	35156,60	0,55	0,00	0,00	14,13	2,03
2	myciel6	597470,00	0,00	0,00	0,00	89,73	5,06	597470,00	0,00	0,00	*	90,99	6,75
2	queen10_10	410433,00	1,58	0,00	0,00	222,48	44,81	410433,40	1,14	0,00	0,00	265,08	46,16
2	queen5_5	2521,00	0,00	0,00	0,01	2,55	0,16	2521,00	0,00	0,00	0,00	2,79	0,22
2	queen6_6	7741,80	0,45	0,01	0,00	8,15	0,79	7741,80	0,84	0,01	0,01	9,43	1,37
2	queen7_7	33383,80	0,84	0,00	0,00	18,30	1,62	33383,60	0,55	0,00	0,00	19,75	1,01
2	queen8_12	301867,20	0,84	0,00	0,00	206,11	35,75	301867,00	0,71	0,00	0,00	212,87	17,27
2	queen8_8	75461,40	0,89	0,00	0,00	54,27	5,38	75462,00	1,00	0,00	0,00	51,88	6,53
2	queen9_9	185509,80	0,84	0,00	6,09	110,80	19,71	185509,60	0,89	0,00	0,00	131,47	19,09
3	rcp-nb_40_15_1	5,367	0,000	0,00	0,000	6,040	1,236	5,367	0,000	0,00	0,000	6,176	0,665
3	rcp-nb_40_15_2	4,819	0,000	0,00	0,635	4,746	0,465	4,835	0,036	0,75	0,336	5,262	0,499
3	rcp-nb_40_15_3	5,224	0,000	0,00	0,000	6,176	1,115	5,224	0,000	0,00	0,000	6,354	0,722
3	rcp-nb_40_15_4	5,579	0,049	0,87	0,761	5,366	1,190	5,604	0,058	1,04	1,086	5,934	1,042
3	rcp-nb_40_15_5	4,977	0,055	1,10	5,001	5,970	0,689	4,965	0,059	1,18	0,530	5,670	0,720
4	rcp-nb_90_30_1	11,268	0,210	1,86	3,736	54,672	9,066	11,126	0,164	1,48	3,166	65,630	7,092

Continua na próxima página

Tabela A.3: Resultados: ILS-PCR com $maxNivel = 3$, $t_{max} = 7200$

#Conjunto	Instância	ILS-PCR-RVND						ILS-PCR-VND					
		$\bar{z}$	σ_z	$C_v\%$	$GAP\%$	$\bar{t}$	σ_t	$\bar{z}$	σ_z	$C_v\%$	$GAP\%$	$\bar{t}$	σ_t
4	rcp-nb_90_30_2	10,701	0,169	1,58	3,734	65,846	19,998	10,691	0,139	1,30	3,639	66,556	9,722
4	rcp-nb_90_30_3	11,057	0,140	1,27	4,475	71,082	13,217	11,149	0,074	0,66	5,882	68,034	13,460
4	rcp-nb_90_30_4	10,491	0,160	1,52	3,481	68,488	3,197	10,330	0,087	0,84	2,148	66,128	13,832
4	rcp-nb_90_30_5	9,811	0,108	1,10	4,999	63,208	10,403	9,962	0,159	1,60	5,071	59,576	10,736
4	rcp-nb_90_31_1	9,626	0,144	1,50	3,411	66,646	12,150	9,697	0,129	1,33	4,690	69,962	10,685
4	rcp-nb_90_31_2	9,228	0,165	1,79	2,955	68,218	13,244	9,116	0,093	1,02	2,156	66,728	12,450
4	rcp-nb_90_31_3	9,414	0,116	1,23	3,917	61,848	10,511	9,438	0,070	0,74	5,260	60,478	2,370
4	rcp-nb_90_31_4	9,353	0,077	0,83	4,316	61,398	10,372	9,357	0,132	1,41	4,37	64,742	5,106
4	rcp-nb_90_31_5	7,883	0,053	0,67	0,028	64,034	10,379	8,017	0,121	1,51	4,565	76,228	9,568
5	rcp-nb_120_40_1	11,461	0,141	1,23	8,417	150,480	23,606	11,254	0,083	0,73	8,808	185,632	36,305
5	rcp-nb_120_40_2	12,010	0,272	2,26	9,655	173,394	16,813	12,083	0,346	2,87	9,074	191,024	25,899
5	rcp-nb_120_40_3	11,478	0,388	3,38	10,813	193,254	27,387	11,532	0,296	2,57	6,594	179,044	20,409
5	rcp-nb_120_40_4	11,937	0,360	3,01	8,932	167,294	16,432	12,253	0,405	3,30	12,556	167,746	23,146
5	rcp-nb_120_40_5	12,999	0,163	1,26	9,423	175,112	11,064	13,148	0,240	1,83	10,178	174,894	18,538
5	rcp-nb_120_41_1	10,972	0,128	1,17	10,072	170,742	14,801	10,835	0,211	1,95	6,394	215,690	46,971
5	rcp-nb_120_41_2	11,735	0,370	3,16	7,555	182,548	47,638	11,580	0,260	2,25	8,620	191,204	46,879
5	rcp-nb_120_41_3	10,319	0,203	1,96	7,742	170,964	11,559	10,324	0,253	2,45	9,483	183,828	24,871
5	rcp-nb_120_41_4	10,096	0,230	2,28	9,499	189,112	9,486	10,163	0,189	1,86	8,262	177,162	22,565
5	rcp-nb_120_41_5	10,621	0,352	3,32	0,000	170,192	38,377	10,500	0,339	3,22	8,243	169,720	16,478

Tabela A.4: Resultados: ILS-PCR-RVND-15

#Conjunto	Instância	$\bar{z}$	σ_z	C_v%	GAP%
1	1-FullIns_3	13292	0	0,00	0,00
1	2-Insertions_3	32397	0	0,00	0,00
1	DSJC125.5	864090,6	371,251	0,04	0,11
1	DSJC125.9	151861,4	3031,304	2,00	6,40
1	miles1000	187556,4	24,6536	0,01	0,09
1	miles1500	4317,6	56,76971	1,31	0,78
1	myciel3	110	0	0,00	0,00
1	myciel4	2600	0	0,00	0,00
1	myciel5	52829	0	0,00	0,00
1	queen5_5	3850,8	1,788854	0,05	0,02
1	queen6_6	12062,6	0,894427	0,01	0,01
1	queen7_7	48013,6	4,505552	0,01	0,02
1	queen8_8	109802,4	1,516575	0,00	0,01
1	queen9_9	277439,4	17,88295	0,01	0,01
2	1-FullIns_3	8787	0	0,00	0,00
2	2-FullIns_3	70830	0	0,00	0,00
2	2-Insertions_3	22101	0	0,00	0,00
2	DSJC125.5	582856,2	90,54391	0,02	0,05
2	DSJC125.9	39845	880,0148	2,21	9,61
2	huck	106103,2	0,447214	0,00	0,00
2	miles1000	58829	46,43813	0,08	0,10
2	myciel3	28	0	0,00	0,00
2	myciel4	1648	0	0,00	0,00
2	myciel5	35156,8	0,447214	0,00	0,00
2	myciel6	597470	0	0,00	*
2	queen10_10	410439,6	2,408319	0,00	0,00
2	queen5_5	2521	0	0,00	0,00
2	queen6_6	7741,8	0,447214	0,01	0,01
2	queen7_7	33383,8	0,83666	0,00	0,00
2	queen8_12	301869,8	1,48324	0,00	0,00
2	queen8_8	75462,6	0,894427	0,00	0,00
2	queen9_9	185511,8	1,48324	0,00	0,00
3	rcp-nb_40_15_1	5,367	0	0,00	0,00
3	rcp-nb_40_15_2	4,819	0	0,00	0,00
3	rcp-nb_40_15_3	5,224	0	0,00	0,00
3	rcp-nb_40_15_4	5,5792	0,048618	0,87	0,63
3	rcp-nb_40_15_5	4,9766	0,05467	1,10	0,76
4	rcp-nb_90_30_1	11,4394	0,21051	1,84	6,07
4	rcp-nb_90_30_2	11,0068	0,16512	1,50	6,70
4	rcp-nb_90_30_3	11,1214	0,194028	1,74	5,62
4	rcp-nb_90_30_4	10,9652	0,202057	1,84	8,43
4	rcp-nb_90_30_5	10,0464	0,108661	1,08	5,96
4	rcp-nb_90_31_1	9,8126	0,162708	1,66	5,93
4	rcp-nb_90_31_2	9,3482	0,362827	3,88	4,75
4	rcp-nb_90_31_3	9,4914	0,08426	0,89	5,86
4	rcp-nb_90_31_4	9,4794	0,293313	3,09	5,93
4	rcp-nb_90_31_5	8,27	0,281866	3,41	7,86
5	rcp-nb_120_40_1	11,9576	0,404342	3,38	15,61
5	rcp-nb_120_40_2	13,327	0,510409	3,83	20,30
5	rcp-nb_120_40_3	12,972	0,721659	5,56	19,90
5	rcp-nb_120_40_4	12,9344	0,49942	3,86	18,82
5	rcp-nb_120_40_5	13,969	0,399992	2,86	17,06
5	rcp-nb_120_41_1	12,0422	0,481607	4,00	18,25
5	rcp-nb_120_41_2	12,7942	0,616092	4,82	20,01
5	rcp-nb_120_41_3	11,184	0,6712	6,00	18,60
5	rcp-nb_120_41_4	10,6722	0,394249	3,69	13,69
5	rcp-nb_120_41_5	11,3102	0,345116	3,05	16,60

Tabela A.5: Resultados: ILS-PCR-RVND-30

#Conjunto	Instância	$\bar{z}$	σ_z	C_v%	GAP%
1	1-FullIns_3	13292	0	0,00	0,00
1	2-Insertions_3	32397	0	0,00	0,00
1	DSJC125.5	863841,6	184,3483	0,02	0,08
1	DSJC125.9	150418,8	2699,097	1,79	5,39
1	miles1000	187502,8	32,78262	0,02	0,06
1	miles1500	4315,6	52,35265	1,21	0,74
1	myciel3	110	0	0,00	0,00
1	myciel4	2600	0	0,00	0,00
1	myciel5	52829	0	0,00	0,00
1	queen5_5	3850,8	1,788854	0,05	0,02
1	queen6_6	12062,6	0,894427	0,01	0,01
1	queen7_7	48013,6	4,505552	0,01	0,02
1	queen8_8	109801,8	1,30384	0,00	0,01
1	queen9_9	277430,4	9,343447	0,00	0,01
2	1-FullIns_3	8787	0	0,00	0,00
2	2-FullIns_3	70830	0	0,00	0,00
2	2-Insertions_3	22101	0	0,00	0,00
2	DSJC125.5	582784	53,69823	0,01	0,04
2	DSJC125.9	39272,8	829,3224	2,11	8,04
2	huck	106103	0	0,00	0,00
2	miles1000	58805,2	12,07063	0,02	0,06
2	myciel3	28	0	0,00	0,00
2	myciel4	1648	0	0,00	0,00
2	myciel5	35156,8	0,447214	0,00	0,00
2	myciel6	597470	0	0,00	*
2	queen10_10	410436,8	1,788854	0,00	0,00
2	queen5_5	2521	0	0,00	0,00
2	queen6_6	7741,8	0,447214	0,01	0,01
2	queen7_7	33383,8	0,83666	0,00	0,00
2	queen8_12	301868,8	1,643168	0,00	0,00
2	queen8_8	75461,8	0,83666	0,00	0,00
2	queen9_9	185511,4	1,140175	0,00	0,00
3	rcp-nb_40_15_1	5,367	0	0,00	0,00
3	rcp-nb_40_15_2	4,819	0	0,00	0,00
3	rcp-nb_40_15_3	5,224	0	0,00	0,00
3	rcp-nb_40_15_4	5,5792	0,048618	0,87	0,63
3	rcp-nb_40_15_5	4,9766	0,05467	1,10	0,76
4	rcp-nb_90_30_1	11,2676	0,209936	1,86	4,47
4	rcp-nb_90_30_2	10,866	0,189799	1,75	5,33
4	rcp-nb_90_30_3	11,1214	0,194028	1,74	5,62
4	rcp-nb_90_30_4	10,7442	0,268999	2,50	6,24
4	rcp-nb_90_30_5	9,951	0,103586	1,04	4,96
4	rcp-nb_90_31_1	9,7712	0,215042	2,20	5,49
4	rcp-nb_90_31_2	9,2662	0,213985	2,31	3,83
4	rcp-nb_90_31_3	9,3648	0,086886	0,93	4,45
4	rcp-nb_90_31_4	9,2392	0,167743	1,82	3,24
4	rcp-nb_90_31_5	7,9476	0,112378	1,41	3,66
5	rcp-nb_120_40_1	11,7376	0,307684	2,62	13,48
5	rcp-nb_120_40_2	12,697	0,693073	5,46	14,61
5	rcp-nb_120_40_3	12,5146	0,551298	4,41	15,67
5	rcp-nb_120_40_4	12,7488	0,31573	2,48	17,11
5	rcp-nb_120_40_5	13,7332	0,314639	2,29	15,09
5	rcp-nb_120_41_1	11,569	0,476284	4,12	13,60
5	rcp-nb_120_41_2	12,1314	0,424439	3,50	13,79
5	rcp-nb_120_41_3	10,6806	0,349187	3,27	13,26
5	rcp-nb_120_41_4	10,4728	0,584189	5,58	11,57
5	rcp-nb_120_41_5	11,0506	0,187137	1,69	13,92

Tabela A.6: Resultados: ILS-PCR-RVND-60

#Conjunto	Instância	$\bar{z}$	σ_z	C_v%	GAP%
1	1-FullIns_3	13292	0	0,00	0,00
1	2-Insertions_3	32397	0	0,00	0,00
1	DSJC125.5	863768,2	151,2637	0,02	0,07
1	DSJC125.9	148438	1913,207	1,29	4,01
1	miles1000	187459,6	31,29377	0,02	0,04
1	miles1500	4308,4	54,56006	1,27	0,57
1	myciel3	110	0	0,00	0,00
1	myciel4	2600	0	0,00	0,00
1	myciel5	52829	0	0,00	0,00
1	queen5_5	3850,8	1,788854	0,05	0,02
1	queen6_6	12062,6	0,894427	0,01	0,01
1	queen7_7	48013,6	4,505552	0,01	0,02
1	queen8_8	109801,2	2,167948	0,00	0,00
1	queen9_9	277426,6	1,949359	0,00	0,00
2	1-FullIns_3	8787	0	0,00	0,00
2	2-FullIns_3	70830	0	0,00	0,00
2	2-Insertions_3	22101	0	0,00	0,00
2	DSJC125.5	582767,2	56,68068	0,01	0,04
2	DSJC125.9	38833,6	793,655	2,04	6,83
2	huck	106103	0	0,00	0,00
2	miles1000	58803,6	10,78425	0,02	0,06
2	myciel3	28	0	0,00	0,00
2	myciel4	1648	0	0,00	0,00
2	myciel5	35156,8	0,447214	0,00	0,00
2	myciel6	597470	0	0,00	*
2	queen10_10	410436,8	1,788854	0,00	0,00
2	queen5_5	2521	0	0,00	0,00
2	queen6_6	7741,8	0,447214	0,01	0,01
2	queen7_7	33383,8	0,83666	0,00	0,00
2	queen8_12	301868,2	1,643168	0,00	0,00
2	queen8_8	75461,4	0,894427	0,00	0,00
2	queen9_9	185510,2	1,30384	0,00	0,00
3	rcp-nb_40_15_1	5,367	0	0,00	0,00
3	rcp-nb_40_15_2	4,819	0	0,00	0,00
3	rcp-nb_40_15_3	5,224	0	0,00	0,00
3	rcp-nb_40_15_4	5,5792	0,048618	0,87	0,63
3	rcp-nb_40_15_5	4,9766	0,05467	1,10	0,76
4	rcp-nb_90_30_1	11,2676	0,209936	1,86	4,47
4	rcp-nb_90_30_2	10,7014	0,169096	1,58	3,74
4	rcp-nb_90_30_3	11,0566	0,140072	1,27	5,00
4	rcp-nb_90_30_4	10,565	0,166143	1,57	4,47
4	rcp-nb_90_30_5	9,811	0,107796	1,10	3,48
4	rcp-nb_90_31_1	9,6272	0,143134	1,49	3,93
4	rcp-nb_90_31_2	9,2284	0,164819	1,79	3,41
4	rcp-nb_90_31_3	9,353	0,077444	0,83	4,32
4	rcp-nb_90_31_4	9,2134	0,119134	1,29	2,95
4	rcp-nb_90_31_5	7,8834	0,052795	0,67	2,82
5	rcp-nb_120_40_1	11,6052	0,31895	2,75	12,20
5	rcp-nb_120_40_2	12,5032	0,461908	3,69	12,87
5	rcp-nb_120_40_3	12,0932	0,712575	5,89	11,78
5	rcp-nb_120_40_4	12,5188	0,196285	1,57	15,00
5	rcp-nb_120_40_5	13,3684	0,331606	2,48	12,03
5	rcp-nb_120_41_1	11,2548	0,22157	1,97	10,51
5	rcp-nb_120_41_2	11,9304	0,233668	1,96	11,91
5	rcp-nb_120_41_3	10,453	0,290514	2,78	10,85
5	rcp-nb_120_41_4	10,1954	0,343405	3,37	8,61
5	rcp-nb_120_41_5	10,7622	0,351193	3,26	10,95