HEURÍSTICAS PARA O PROBLEMA DE ROTULAÇÃO CARTOGRÁFICA DE PONTOS E DE COLORAÇÃO DE VÉRTICES COM PESOS.

CELSO DE OLIVEIRA

HEURÍSTICAS PARA O PROBLEMA DE ROTULAÇÃO CARTOGRÁFICA DE PONTOS E DE COLORAÇÃO DE VÉRTICES COM PESOS.

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais. Departamento de Ciência da Computação como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Thiago Ferreira de Noronha
Coorientador: Sebastián Alberto Urrutia

Belo Horizonte

Março de 2012

Oliveira, Celso de
048h Heurísticas para o problema de rotulação cartográfica de pontos e de coloração de vértices com pesos. / Celso de Oliveira. — Belo Horizonte, 2012
xiv, 48 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais. Departamento de Ciência da Computação
Orientador: Thiago Ferreira de Noronha

Coorientador: Sebastián Alberto Urrutia

1. Computação — Teses. 2. Programação heurística — Teses. 3. Teoria dos grafos — Teses I. Orientador. II. Coorientador. III. Título.

CDU 519.6*62 (043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Heurísticas para o problema de rotulação cartográfica
de pontos e coloração de vértices com pesos

CELSO DE OLIVEIRA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

Thiago F. Noronha
PROF. THIAGO FERREIRA DE NORONHA - Orientador
Departamento de Ciência da Computação - UFMG

PROF. SEBASTIÁN ALBERTO URRUTIA - Co-orientador
Departamento de Ciência da Computação - UFMG

Adriana Cesário Alvim
PROFA. ADRIANA CESÁRIO DE FARIA ALVIM
Departamento de Informática Aplicada - UNIRIO

PROF. GERALDO ROBSON MATEUS
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 16 de março de 2012.

A minha mãe Dirce, pelo seu esforço, privação e dedicação em proporcionar a seus filhos um horizonte de oportunidades.
Aos meus filhos Raul e Ingrid e minha esposa Luciana, pelos muitos momentos vividos.
A todos que, de alguma forma, me ajudaram a chegar aqui.

DEDICO

Agradecimentos

Agradeço a Deus e aos Seus.

Agradeço aos Professores Robson e Sebastián pela oportunidade que me proporcionaram em realizar este curso de mestrado, por acreditar que poderia superar minhas limitações e transpor os obstáculos que naturalmente viriam, por acreditar no prof. Thiago e que juntos poderíamos fazer um bom trabalho.

Agradeço ao meu orientador Thiago por acreditar em mim, por acreditar que eu poderia fazer um bom trabalho, por sua paciência, compreensão e dedicação em me ensinar a pesquisar, por sua sabedoria e amizade em nossos diálogos me mostrando um pouco mais de mim mesmo.

A minha mãe sou profundamente grato pelo que sou hoje, o resultado de seu amor, dedicação e da história de sua vida.

A minha esposa Luciana e meus filhos Ingrid e Raul que nasceram durante o curso de mestrado e agora proporcionam uma felicidade que muitas vezes duvidamos que possa existir.

Aos professores, colegas de mestrado e funcionários da secretaria do DCC, em especial a Sheila, pela amizade, compreensão e dedicação.

A todos que de uma forma ou de outra colaboraram direta, ou indiretamente para realização deste trabalho.

"Depois que termina é fácil, difícil é começar!"
(...)

Resumo

Esta dissertação propõe uma heurística *Variable Neighbourhood Descent* que alterna entre vizinhanças que são exploradas por um algoritmo de *Backtracking*. Ela foi aplicada ao problema de rotulação cartográfica de pontos e ao problema de coloração de vértices com pesos. O primeiro consiste em posicionar rótulos em regiões de um mapa cartográfico, proporcionando uma boa visualização pela redução das sobreposições de rótulos. O segundo consiste em atribuir cores aos vértices de um grafo tal que vértices adjacentes não recebam a mesma cor. Cada vértice do grafo possui um peso e para cada cor é atribuído um peso que corresponde ao peso máximo dos vértices coloridos com essa cor. O objetivo do problema de coloração de vértices com pesos é minimizar a soma dos pesos das cores utilizadas. Os experimentos computacionais mostraram que a heurística proposta é rápida e encontra resultados equivalentes ou melhores que os melhores trabalhos publicados na literatura para os dois problemas.

Abstract

This master thesis proposes an heuristic *Variable Neighbourhood Descent* that alternates between neighbourhoods exploited by *Backtracking* algorithm. This heuristic was applied to Point Feature Lable Placement Problem and Weighted Vertex Coloring Problem. The first problem consists in placing point feature entity labels in cartographic maps maximizing the map's legibility by minimizing the number of label overlaps. The second consists in given a graph $G = (V, E)$, V is a set of vertex and E is a set of edge, each vertex has associated a weight $w_v \geq 0$, assign a color to each vertex in such way that color on adjacent vertex are diferent. Given a coloring, C_k is the set of vertex with color k for $k = 1, ..., n$, the k color weight is the maximum weight of a vertex in C_k. The objective of $WVCP$ is minimize the sum of the weight colors. The low computational cost and results comparable with the best works in literature show its applicability to the both problems.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Os problemas de Otimização Combinatória podem ser descritos pela tripla (S, Ω, f) onde S é um espaço de busca definido sobre um conjunto finito de variáveis de decisão $X = \{x_1, \ldots, x_r\}$, Ω é um conjunto de restrições sobre as variáveis de decisão X e f é uma função tal que $f : S \to \mathbb{R}$ é uma função objetivo que fornece um valor para cada solução $s \in S$ [10].

Heurísticas e algoritmos exatos são os métodos mais utilizados para resolver problemas de otimização combinatória NP-Difíceis. Os métodos exatos garantem encontrar a solução ótima em um tempo finito. No entanto, assumindo que $P \neq NP$, não existem algoritmos de tempo polinomial para resolver problemas desta classe [56]. Os métodos heurísticos não garantem a otimalidade de uma solução, no entanto, muitas vezes fornecem soluções de boa qualidade com baixo custo computacional, mesmo para problemas de grande porte [19].

Os métodos heurísticos devem ser projetados com o objetivo de explorar o espaço de busca de forma eficaz e eficiente, explorando intensamente regiões promissoras do espaço de busca a procura de boas soluções [9]. As estratégias para alcançar esses objetivos são chamadas respectivamente de *intensificação* e *diversificação* e determinam o comportamento das heurísticas no processo de exploração do espaço de busca [9].

A metaheurística *Variable Neighbourhood Descent* (VND) proposta Hansen e Mladenović [41; 57; 65; 66] utiliza diversas vizinhanças para explorar o espaço de soluções, onde o principio básico consiste em encontrar um mínimo local em relação a um conjunto de vizinhanças. O VND é uma metaheurística extensivamente utilizada [6; 14; 29; 58; 63; 64; 67; 70] para resolver problemas de otimização combinatória, sendo considerado uma heurística de descida rápida, *a steepest descent heuristic* [57], e utilizada como uma busca local rápida. Em Hansen et al. [41; 57; 64; 65; 66] a estrutura do VND é extensivamente detalhada apresentando aplicações modeladas para diversos

problemas tais como: *Traveling Salesman Problem* TSP [78], *pMediam Problem* PM [71], *Minimum Sum of Squares Clustering* MSSC [55] e *Uncapacited Facility Location Problem* [24].

A metaheurística *Variable Neighbourhood Descent* (VND) utiliza diversas estruturas de vizinhanças $N_q(s), q = 1, \ldots, q_{max}$, onde S é o conjunto de soluções viáveis, $s \in S$ e $N_q(s) \subseteq S$. Estas vizinhanças podem ser aninhadas (e.g. 1-opt, 2-opt, 3-opt, etc.) ou completamente distintas umas das outras (e.g. Insertion, Exchange, etc.). VND se baseia no fato de que um ótimo local para uma vizinhança $N_r(s)$ não é necessariamente um ótimo local para outra vizinhança $N_t(s)$, $r \neq t$ [66]. Partindo de uma solução inicial s e uma vizinhança $N_1(s)$, VND realiza uma busca local na vizinhança da solução corrente, movendo-se para uma solução que possua um custo melhor que a solução corrente. Ao encontrar um ótimo local para uma vizinhança $N_r(s)$, VND seleciona outra vizinhança $N_t(s), r \neq t$, e sucessivamente realiza busca local até encontrar uma solução que é um ótimo local para todas as vizinhanças. Sempre que a melhor solução conhecida é atualizada, VND retorna para a primeira vizinhança e a busca é reiniciada a partir desta solução.

O pseudo código do VND é apresentado na Figura 1.1. A heurística VND recebe uma solução inicial s e um número de vizinhanças q_{max}. O custo de uma solução s é dado pela função $f(s)$. O laço das linhas $1-8$ é executado até que um ótimo local para todas as vizinhanças seja encontrado. Na Linha 2, o contador de vizinhanças q é inicializado. O laço das linhas $3-8$ efetua a busca na vizinhança $N_q(s)$. Na Linha 4, o ótimo local s' de $N_q(s)$ é retornado. Se o custo da solução s' for menor que o custo da solução corrente s (Linha 5), a solução s é atualizada e o contador de vizinhanças q é reinicializado na Linha 6. Do contrário, a próxima vizinhança é selecionada na Linha 8. Por fim, na Linha 9, o ótimo local em relação a todas as vizinhanças é retornado.

```
Procedimento VariableNeighbourhoodDescent(s, q_max)
1   enquanto Custo da solução s melhora faça
2       q ← 1
3       enquanto q ≤ q_max
4           s' ← BuscaLocal(N_q, s)
5           se f(s') < f(s) então
6               s ← s' e q ← 1
7           senão
8               q ← q + 1
9   retorne s
```

Figura 1.1. Pseudo código da heurística VND.

Backtracking é um algoritmo genérico utilizado para encontrar todas as possíveis soluções de um problema computacional. No contexto desta dissertação, o problema computacional a ser solucionado, é enumerar (e avaliar) todos os vizinhos de uma solução s. Backtracking utiliza uma estrutura de árvore de decisão, que é percorrida no sentido do nó raiz para os nós folhas, recursivamente na forma de uma busca em profundidade. Em cada nó da árvore, o algoritmo verifica se o caminho do nó até uma de suas folhas pode ainda levar a uma solução viável. Se for este o caso, o algoritmo continua a busca nos nós inferiores. Caso contrário, o algoritmo retorna ao nó anterior da árvore, podando toda a sub árvore do nó em questão. Esta última operação é conhecida como *backtrack*. Quando uma solução viável para o problema computacional é encontrada em uma folha, ela é processada, e o algoritmo retorna ao nó anterior da árvore (*backtrack*), continuando a busca por outras soluções para o problema. Backtracking foi extensivamente utilizado para resolver problemas de otimização combinatória [16; 37; 54; 69; 72; 79; 83]. Apesar da sua complexidade de tempo intrinsecamente exponencial, ele foi amplamente utilizado para problemas NP-Difícieis [12; 13; 22; 23; 38; 39; 60; 69].

O objetivo desta dissertação é propor e avaliar uma heurística *Variable Neighbourhood Descent* (VND) [65; 66] que alterna entre vizinhanças aninhadas (potencialmente muito grandes) que são exploradas por um algoritmo de *Backtracking* [1]. Ao invés de utilizar o algoritmo de *backtracking* para enumerar todas as soluções de um problema de otimização NP-Difícil, em busca de identificar qual é a solução ótima do problema, a heurística proposta nesta dissertação utiliza este algoritmo para enumerar todos os vizinhos de uma vizinhança (potencialmente muito grande) em busca de soluções aprimorantes. Esta técnica foi aplicada aos problemas de Rotulação Cartográfica de Pontos (PFLP, do inglês, *Point feature Label Placement Problem*) [42; 43] e Coloração de Vértices com Pesos (WVCP, do inglês *Weighted Vertex Coloring Problem*) [56].

O restante desta dissertação está organizada da seguinte forma. Os capítulos 2 e 3 apresentam respectivamente o PFLP e o WVCP e suas respectivas revisões bibliográficas. O Capítulo 4 propõe uma variação da metaheurística VND, que alterna entre vizinhanças aninhadas que são exploradas por um algoritmo de *Backtracking*, e mostra o resultado da aplicação desta técnica para PFLP e WVCP. Por fim, no Capítulo 5, encontra-se as considerações finais sobre este trabalho e propostas para trabalhos futuros.

[1] O termo *Backtrack* foi cunhado pelo matemático norte americano D. H. Lehmer na década de 1950

Capítulo 2

Problema de Rotulação Cartográfica de Pontos

De forma geral o problema de rotulação cartográfica consiste em identificar com clareza entidades cartográficas através de rótulos que são posicionados em determinadas regiões de um mapa cartográfico, evitando sobreposições dos rótulos e respeitando algumas padronizações cartográficas, tais como posição e tamanho do rótulo [3; 42; 43]. As entidades cartográficas podem ser divididas em três tipos: entidades linha (rios, estradas,...), entidades área ou polígono (estados, oceanos,...) e entidades ponto (cidades, edificações,...) sendo o problema de rotulação das entidades ponto conhecido como Problema de Rotulação Cartográfica de Pontos [36] (PFLP, do inglês, *Point Feature Lable Placement Problem*). O problema de rotulação cartográfica pode ser discreto ou contínuo onde no primeiro caso as entidades cartográficas possuem uma lista predefinida de posições candidatas que poderão receber o rótulo, no segundo o rótulo pode circular ou tangenciar o ponto que representa a entidade cartográfica, sendo inserido na melhor posição encontrada.

As sobreposições dos rótulos são chamadas de conflitos e quando não são aceitas e o tamanho do rótulo é fixo, o problema consiste em rotular sem sobreposições o maior número de entidades cartográficas. Este problema é conhecido como Problema da Maximização do Número de Rótulos (PMNR), Máximo Conjunto Independente de Vértices (MCIV) [76; 80] ou Problema de Maximização de Número de Legendas (PMNL) [15]. Quando o tamanho dos rótulos é variável o objetivo é determinar o tamanho ideal de cada rótulo para que todos as entidades cartográficas sejam rotuladas sem sobreposições. Neste caso o problema é conhecido como Problema da Maximização do Tamanho do Rótulo (PMTR) [80] ou Problema de Maximização do Tamanho de Legendas (PMTL)[74]. Por outro lado as sobreposições dos rótulos podem ser aceitas e

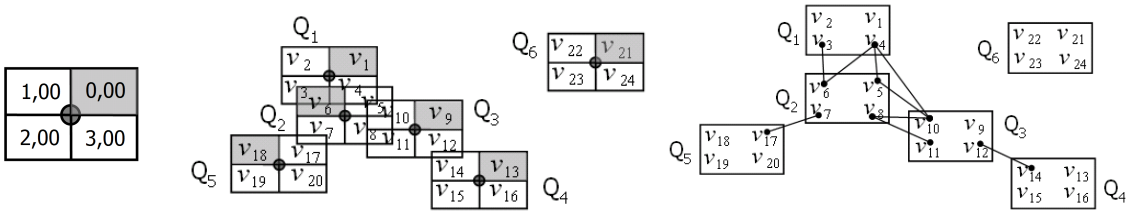
neste caso o tamanho do rótulo é único e todos as entidades cartográficas são rotuladas, fazendo com que o objetivo do problema de rotulação cartográfica seja minimizar o número de sobreposições dos rótulos ou maximizar o número de rótulos livres. No primeiro caso temos o Problema da Minimização do Número de Conflitos (PMNC) [30] e no segundo o Problema do Número Máximo de Rótulos Livres de Conflitos (PNMRLC) [31; 32].

A preferência cartográfica associa um peso a cada posição candidata da lista de posições candidatas, indicando a ordem preferencial para rotulação do ponto cartográfico; Quanto menor for o peso da preferência cartográfica, maior é a preferência para seleção da posição candidata.

Baseando-se nos trabalhos de Marks e Sheiber [44] o PFLP pode ser modelado por um grafo particionado não direcionado $G = (V, E, Q)$ e um conjunto $P = \{1, \ldots, p\}$ de pontos cartográficos, onde V é o conjunto de posições candidatas, E é o conjunto de arestas entre posições candidatas em conflito (sobrepostas), $Q = \{Q_1, \ldots, Q_p\}$ é uma partição dos vértices em V, sendo que Q_q é o conjunto das posições candidatas para cada ponto cartográfico $q \in P$. O PFLP consiste em selecionar uma posição candidata $v \in Q_q$ para cada ponto cartográfico $q \in P$ e uma solução $s = \{v_1, \ldots, v_p\}$ é um conjunto contendo as posições candidatas selecionadas. Dada uma solução s, diz-se que um ponto cartográfico q está em *conflito* se o grau da posição candidata $v \in Q_q \cap S$ no grafo induzido em G por s for maior que zero. Dados os pontos cartográficos distintos p e q, eles são adjacentes em s caso exista a aresta $[u, v] \in E$ para as posições candidatas $u \in Q_p \cap s$ e $v \in Q_q \cap s$ no grafo induzido em G por s. O objetivo do problema é encontrar a atribuição de posições candidatas a pontos cartográficos que minimiza o número de pontos cartográficos em conflito. A Figura 2 ilustra um exemplo de um problema de rotulação cartográfica de pontos com seis pontos cartográficos, cada um com quatro posições candidatas.

A formulação matemática do PMNC foi apresentada por Ribeiro e Lorena [31], Figura 2.2. Nesta formulação todos os pontos cartográficos são rotulados e o custo da solução é fornecido pelo somatório do peso da posição candidata selecionada e o número de posições candidatas em conflito. Nesta formulação cada posição candidata é representada pela variável binária $x_{i,j}$ onde $i = 1, \ldots, |P|$ e $j = 1 \ldots, |Q_i|$. A preferência cartográfica é definida por um peso $w_{i,j}$ associado a cada posição candidata. Caso a variável $x_{i,j} = 1$ a posição candidata j do ponto cartográfico i estará selecionada para receber o rótulo, do contrário não. O conjunto $S_{i,j}$ possui os pares de índices de posições candidatas (k, t) em conflito com a posição candidata $x_{i,j}$, assim $\forall (k, t) \in S_{i,j}$ a posição candidata $x_{k,t}$ esta em conflito com $x_{i,j}$.

Esta dissertação avaliará o problema discreto de rotulação cartográfica de pontos



(a) Preferências Cartográficas. (b) Problema de Rotulação Cartográfica. (c) Grafo de conflitos.

Figura 2.1. (a) Preferências cartográficas de um ponto cartográfico tendo um peso associado a cada posição candidata [42]. (b) Exemplo de um problema de rotulação cartográfica com seis pontos cartográficos onde $v_1, \ldots, v_{24}$ são posições candidatas. (c) Grafo de conflitos correspondente ao problema da Figura 2.1(b), onde $Q_1, \ldots, Q_6$ representam respectivamente o conjunto de posições candidatas para os pontos cartográficos $1, \ldots, 6$. Cada conjunto de posições candidatas Q_k possui quatro elementos tal que v_1, v_2, v_3 e $v_4 \in Q_1$, ..., v_{21}, v_{22}, v_{23} e $v_{24} \in Q_6$.

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{|P|} \sum_{j=1}^{|Q_i|} \left(w_{i,j} x_{i,j} + \sum_{(k,t) \in S_{(i,j)}} y_{i,j,k,t} \right) \\
s.a. \quad & \sum_{j=1}^{|Q_i|} x_{i,j} = 1 && \forall i = 1, \ldots, |P| \\
& x_{i,j} + x_{k,t} - y_{i,j,k,t} \leq 1 && \forall i = 1, \ldots, |P| \\
& && \forall j = 1, \ldots, |Q_i| \\
& && (k,t) \in S_{i,j} \\
& x_{i,j}, y_{i,j,k,t} \in \{0,1\} && \forall i = 1, \ldots, |P| \\
& && \forall j = 1, \ldots, |Q_i| \\
& && (k,t) \in S_{i,j}
\end{aligned}
$$

Figura 2.2. Formulação de programação linear inteira $0-1$ de Ribeiro e Lorena [31].

baseando-se no Problema da Minimização do Número de Conflitos (PMNC) [30; 31]. O custo da solução será fornecido pelo número de rótulos em conflito não sendo avaliada a preferência cartográfica, conforme formulação 2.3.

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{|P|} \sum_{j=1}^{|Q_i|} \sum_{(k,t)\in S_{(i,j)}} y_{i,j,k,t} & \\
s.a. \quad & \sum_{j=1}^{|Q_i|} x_{i,j} = 1 & \forall i = 1,\ldots,|P| \\
& x_{i,j} + x_{k,t} - y_{i,j,k,t} \leq 1 & \forall i = 1,\ldots,|P| \\
& & \forall j = 1,\ldots,|Q_i| \\
& & (k,t) \in S_{i,j} \\
& x_{i,j}, y_{i,j,k,t} \in \{0,1\} & \forall i = 1,\ldots,|P| \\
& & \forall j = 1,\ldots,|Q_i| \\
& & (k,t) \in S_{i,j}
\end{aligned}
$$

Figura 2.3. Formulação de programação linear inteira $0-1$ de Ribeiro e Lorena [31] sem o custo da preferência cartográfica na função objetivo

2.1 Trabalhos Relacionados

Marks e Sheiber [44] provaram que o PFLP é NP-difícil, modelando uma versão simplificada do PFLP (SLP, do inglês, *Simple Admissible Point-feature-label Placement*) e o reduzindo a um problema NP-Completo Planar 3-Sat.

Hirsch [77] trabalhou no problema contínuo de rotulação cartográfica e desenvolveu um algoritmo baseado em um sistema vetorial. Os rótulos em conflito são modelados como partículas se repelem, sendo posicionados ao redor do ponto cartográfico de forma a não se sobreporem, minimizando o número de conflitos.

Zoraster [76] propôs uma formulação de programação linear inteira $0-1$, Figura 2.4, com restrições que não aceitam sobreposições das posições candidatas e a função objetivo minimiza o custo da solução utilizando o somatório da preferência das posições candidatas selecionadas, onde dado:

- $k = |P|$, número de pontos cartográficos a serem rotulados.
- $m = |E|$, número de arestas de conflitos entre posições candidatas.
- $N_j = |Q_j|$, número de posições candidatas do ponto cartográfico j.
- $x_{i,j}$, posição candidata i do ponto cartográfico j tal que:

  a Se $x_{i,j} = 1$ a posição candidata será selecionada.

b Se $x_{i,j} = 0$ a posição candidata não será selecionada.

- $w_{i,j}$, peso da preferência cartográfica da posição candidata i do ponto cartográfico j.
- $\sum_{i=1}^{|N_j|} x_{i,j} = 1$, restrição que garante que uma posição candidata para cada ponto cartográfico será selecionada.
- (r, s), posição candidata r do ponto cartográfico s.
- (r', s'), posição candidata r' do ponto cartográfico s'.
- $x_{r_q,s_q} + x_{r'_q,s'_q} \leq 1$, restrição que garante que duas posições candidatas em conflito não serão selecionadas simultaneamente.

$$
\begin{aligned}
\min \quad & \sum_{j=1}^{k} \sum_{i=1}^{N_j} w_{i,j} x_{i,j} \\
s.a. \quad & \sum_{i=1}^{N_j} x_{i,j} = 1 && 1 \leq j \leq k \\
& x_{r_q,s_q} + x_{r'_q,s'_q} \leq 1 && 1 \leq q \leq m \\
& x_{i,j} \in \{0,1\}
\end{aligned}
$$

Figura 2.4. Formulação de programação linear inteira $0-1$ de Zoraster [76].

Em [76] Zoraster propôs também a combinação de heurísticas com uma relaxação lagrangeana, utilizando como penalidade a restrição de sobreposição das posições candidatas, obtendo então a formulação apresentada na Figura 2.5.

$$
\begin{aligned}
\min \quad & \sum_{j=1}^{k} \sum_{i=1}^{N_j} w_{i,j} x_{i,j} + \sum_{q=1}^{m} \left(x_{r_q,s_q} + x_{r'_q,s'_q} - 1 \right) d_q \\
s.a. \quad & \sum_{i=1}^{N_j} x_{i,j} = 1 && 1 \leq j \leq k \\
& x_{i,j} \in \{0,1\} \\
& d_q \geq 0 && 1 \leq q \leq m && \textit{multiplicador lagrangeano}
\end{aligned}
$$

Figura 2.5. Formulação de programação linear inteira $0-1$ de Zoraster [76] com o multiplicador lagrangeano.

Para resolver os problemas de ciclagem e dificuldade na convergência para ótimos locais da heurística proposta, o autor propõe aplicar o multiplicador lagrangeano em apenas um dos rótulos de um conflito e reduzir o valor do multiplicador lagrangeano após um determinado número de iterações, caso o custo da solução não melhore.

Christensen et al. [42; 43] propuseram um algoritmo baseado em *Simulated Anneling* (SA) [73], um algoritmo enumerativo, um algoritmo guloso e um algoritmo denominado *Discret Descent Gradient*. O algoritmo guloso gera soluções rápidas, mas com o número de conflitos 50% maior em média que a solução baseada em *Discret Gradient Descent*, que realiza sucessivas buscas locais minimizando o número de conflitos. Na heurística de SA, os movimentos de seleção das posições candidatas são aleatórios, onde ΔK representa a variação do custo da solução causada pelo movimento. Caso o movimento piore ele poderá ser aceito com base na probabilidade $\rho = 1.0 - e^{-\frac{\Delta k}{T}}$. A heurística de SA alcançou resultados melhores para problemas com um menor número de pontos cartográficos quando comparados com a literatura no momento da sua publicação.

Verner et al. [59] propuseram um Algoritmo Genético (AG) [5] para o PFLP, onde o cromossomo é representado por uma lista de posições candidatas contendo uma posição candidata para cada ponto cartográfico. Uma função mapeia as posições candidatas nos cromossomos *pais* de acordo com um mapa de *bits*, onde cada *bit* corresponde uma posição candidata do cromossomo e seu valor determina de qual cromossomo *pai* a posição candidata será selecionada para gerar a próxima geração.

Yamamoto et al. [49] propuseram uma heurística de busca tabu onde o tamanho da lista tabu varia em função do número de pontos cartográficos em conflitos. Desta forma cresce quando o número de conflitos aumentam e decresce quando número de conflitos diminuem. Esta heurística obteve os melhores resultados quando comparados com a literatura no momento de sua publicação.

Yamamoto et al. [50; 51] propuseram um algoritmo genético construtivo (AGC) [45] e uma heurística baseada nos trabalhos de Strijk et al. [80], denominada FALP, (do inglês, *Fast Algorithm for Label Placement*) que se divide nas três etapas descritas a seguir: Na primeira etapa o algoritmo constrói um conjunto contendo os pontos cartográficos livres. Na segunda etapa o algoritmo constrói um conjunto contendo os pontos cartográficos em conflito. Na terceira etapa é aplicada uma busca local para reduzir o número de pontos cartográficos em conflitos.

Ribeiro et al. [32] apresentaram um modelo matemático baseado em minimização de conflitos e utilizou uma relaxação lagrangeana dividida em duas etapas. Na primeira o grafo de conflitos é divido em grupos de posições candidatas e na segunda, as arestas que conectam os agrupamentos são relaxadas no algoritmo baseado na relaxação

lagrangeana.

Cravo et al. [26] propuseram uma heurística gulosa baseada na heurística *Recursive Smallest First* (RSF) [25] que utiliza a técnica de redução proposta por Wagner et al. [21]. Esta técnica consiste em (i) reduzir o tamanho do grafo de conflitos removendo os pontos cartográficos com posições candidatas sem arestas de conflitos, em seguida (ii) selecionar a posição candidata de cada ponto cartográfico que acarreta no menor número de conflitos.

Cravo et al. [27] propuseram uma heurística baseada em GRASP [61; 81], aplicando a técnica de redução proposta por Wagner et al [21] para reduzir o tamanho do grafo de conflitos. Esta heurística obteve os melhores resultados da literatura no momento da sua publicação.

Mauri et al. [35] propuseram uma modelagem por programação linear inteira $0-1$ e uma decomposição lagrangeana que divide o PFLP em vários subproblemas que correspondem a partições do grafo de conflitos. As conexões entre as partições são removidas copiando os vértices de maior grau destas conexões para as outras partições conectadas, por fim os subproblemas são resolvidos utilizando o CPLEX [40] e as soluções obtidas são reunidas na solução final. Ribeiro et al. [33] propuseram a inserção de desigualdades válidas no modelo proposto por Mauri et al.[35] para fortalecer a formulação, apresentando no momento de sua publicação os melhores valores da literatura para os problemas propostos.

Alvim e Taillard [3] propuseram uma heurística baseada em *Partial Optimization Metaheuristic Under Special Intensification Conditions* (POPMUSIC), metaheurística proposta inicialmente por Taillard e Voß [20]. Ela consiste em dividir o problema em subproblemas, otimizando os subproblemas com a busca tabu de Yamamoto [49]. Atualmente, esta é a heurística com melhores resultados na literatura para o PFLP.

Capítulo 3

Problema de Coloração de Vértices com Pesos

Os problemas de coloração de grafos (GCP, do inglês *Graph Coloring Problems*), consistem em: dado um grafo $G = (V, E)$, onde V é um conjunto com n vértices e E é um conjunto de m arestas, deve-se atribuir cores a elementos de G, sujeito a um conjunto de restrições [82]. Dentre os problemas de coloração de grafos podem ser destacados o problema de coloração de faces em um grafo planar, problema de coloração de arestas e problema de coloração de vértices [82].

O problema de coloração de vértices (VCP, do inglês, *Vertex Coloring Problem*), consiste em: dado um grafo $G = (V, E)$, atribuir cores a todos vértices de G tal que os vértices adjacentes recebam cores diferentes [82]. Um conjunto $C \subseteq V$ é chamado conjunto independente se dados $u, v \in C$, $[u, v] \notin E$. Uma k-coloração $s = \{C_1, \ldots, C_k\}$ de G é um particionamento de V em k conjuntos independentes. O VCP pode ser aplicado como problema de decisão ou otimização onde no primeiro o objetivo é avaliar se o grafo G é k-colorível e no segundo o objetivo é minimizar o número de cores k. O número cromático de G definido por $\chi(G)$ corresponde ao menor valor para k tal que o grafo G seja k-colorível [82].

O problema de coloração de vértices com pesos (WVCP, do inglês *Weighted Vertex Coloring Problem*) é uma generalização do VCP, onde cada vértice $v \in V$ possui um peso $w_v \geq 0, w_1 \geq w_2 \geq \ldots \geq w_n$. Dada uma k-coloração $s = \{C_1, \ldots, C_k\}$, onde $C_r \subseteq V$ é um conjunto independente de vértices coloridos com a cor r, defini-se o peso W_r da cor r como o maior peso dos vértices $v \in C_r$. O objetivo do problema WVCP é encontrar a k-coloração cuja soma dos pesos de suas cores é mínima.

Diversas aplicações podem ser modeladas como um WVCP, onde recursos limitados são compartilhados para atender demandas, que podem ou não ser atendidas

simultaneamente por um mesmo recurso. No WVCP, as cores representam recursos, os vértices representam demandas, o peso dos vértices representa o custo das demandas e as arestas representam a impossibilidade que duas demandas sejam atendidas simultaneamente. Numa solução para o WVCP as cores fornecem o número de recursos e o peso das cores o custo necessário para atendimento das demandas.

A formulação matemática para o WVCP utilizada como base neste trabalho foi apresentada em [18] Figura 3.1. O número máximo de cores é definido pela variável $n = |V|$, as cores dos vértices são definidas pela variável binária $x_{ih}, (i \in V, h = 1, \ldots, n)$ onde para $x_{ih} = 1$ a cor h é atribuída ao vértice i e a variável z_h denota o custo da cor h. A função objetivo 3.1 minimiza o somatório dos custos das cores z_h. O custo de cada cor z_h é definido na restrição 3.2 onde é atribuído a z_h o peso do vértice i colorido com a cor h. A restrição 3.3 obriga que será atribuído a cada vértice $i \in V$ uma única cor $h \in 1, \ldots, n$. A restrição 3.4 obriga que será atribuído cores diferentes a vértices adjacentes e por último a restrição 3.5 define que a variável x é binária.

$$
\begin{aligned}
\min \quad & \sum_{h=1}^{n} z_h & & & (3.1)\\
s.a. \quad & z_h \geq w_i x_{ih}, & & i \in V, h = 1, \ldots, n, & (3.2)\\
& \sum_{h=1}^{n} x_{ih} = 1 & & i \in V, & (3.3)\\
& x_{ih} + x_{jh} \leq 1, & & (i,j) \in E, h = 1, \ldots, n, & (3.4)\\
& x_{ih} \in \{1, 0\}, & & i \in V, h = 1, \ldots, n, & (3.5)
\end{aligned}
$$

Figura 3.1. Formulação de programação linear inteira $0-1$ de Malaguti [18].

3.1 Trabalhos Relacionados

Galinier e Hao [62] apresentaram uma heurística baseada em algoritmos evolucionários, (HEA, do inglês, *Hybrid Evolutionary Algorithms*), onde a ideia principal da heurística é criar novas populações com operadores $CrossOver$ especializados e melhorar o custo da solução com uma busca local baseada na busca tabu. Os autores apresentam resultados comparáveis aos melhores resultados para um conjunto de instâncias da literatura.

Avanthay et al. [8] propuseram um VNS [58; 75] apresentando diversas estruturas de vizinhanças, sendo algumas de tamanho exponencial. Os autores utilizaram uma

busca tabu na busca local, no entanto não fornecem uma forma de explorar a vizinhança com eficiência, não apresentando resultados competitivos quando comparado com os resultados alcançados por Galinier e Hao [62].

Escoffier et al. [7] prosseguiram com a investigação baseando-se no trabalho de Guan e Zhu [17], provando que o problema de coloração com pesos (WCP, do inglês, *Weighted Coloring Problems*) é fortemente NP-Difícil para grafos de intervalos e discutiram a aproximação polinomial para o WCP, demonstrando algoritmos de aproximação para grafos que podem ser coloridos com poucas cores e para subgrafos k-tree.

Demange et al. [48] investigaram a complexidade e aproximabilidade do WCP, discutindo algumas propriedades das soluções ótimas, complexidade e apresentando resultados de estratégias de aproximações.

Trick e Yildiz [53] propuseram uma heurística de busca local aplicada a problemas de coloração de grafos caracterizados por grandes vizinhanças, baseando-se em heurísticas utilizadas em visão computacional. A heurística proposta baseia-se na resolução do problema MAX-CUT em etapas, onde encontrar o melhor vizinho, para as estruturas de vizinhanças definidas pelos autores, corresponde à resolução de um problema MAX-CUT.

As características das vizinhanças do GCP foram avaliadas em Chiarandini et al. [47] apresentando um estudo sobre a busca local em problemas onde as vizinhanças são grandes. No primeiro exemplo o uso de vizinhanças de grande escala não ajudaram a melhorar o desempenho de algoritmos de busca local estocástica devido o alto custo computacional, apesar dos resultados iniciais serem promissores. No segundo exemplo os resultados foram melhores quando comparado com o primeiro, onde uma vizinhança de grande escala foi adaptada para o problema podendo ser pesquisada de forma eficiente, resultando em um componente essencial de um novo algoritmo de estado da arte para varias classes de instâncias de k-coloração.

Hertz et al. [1] apresentaram um novo algoritmo de busca local chamado de Pesquisa Espacial Variável, (VSS, do inglês, *Variable Space Search*) aplicado ao problema de k-coloração. O VSS estende a metodologia da Formulação de Pesquisa Espacial (FSS, do inglês, *Formulation Space Search*), considerando várias formulações diferentes para um mesmo problema, cada uma sendo associada a um conjunto de vizinhanças e uma função objetivo, onde os movimentos trocam de vizinhança quando a busca local fica presa em um ótimo local. Os problemas de k-coloração são resolvidos através da combinação de 3 diferentes formulações, onde as duas primeiras possuem diferentes restrições relaxadas a terceira formulação satisfaz todas as restrições.

Malaguti et al. [18] propuseram duas formulações utilizando programação linear inteira, a primeira produz um limite inferior para a solução e a segunda formulação,

baseada no problema SCP (do inglês, *Set Covering Problem*), é utilizada para derivar um algoritmo de duas fases. Na primeira fase um algoritmo de geração de colunas produz um grande número de conjuntos independentes e na segunda fase a solução é melhorada, apresentando no momento de sua publicação os melhores resultados para um grande número de instâncias da literatura.

Capítulo 4

Heurística VND com Backtracking

Esta dissertação propõe uma variação da metaheurística *Variable Neighbourhood Descent* [65; 66] que alterna entre vizinhanças aninhadas (potencialmente muito grandes) que são exploradas por um algoritmo de *Backtracking*. Esta técnica foi chamada de VNDBS (do inglês *Variable Neighbourhood Descent with Backtracking Search*).

VNDBS recebe uma solução inicial s e realiza uma busca local em uma vizinhança exponencialmente grande $\mathcal{N}(s)$. Entretanto, por uma questão de eficiência, esta vizinhança não é explorada exaustivamente. A eficiência de VNDBS é alcançada através dos três princípios descritos a seguir. O pseudo código de VNDBS é apresentado logo em seguida.

Primeiramente, $\mathcal{N}(s)$ é particionada em várias subvizinhanças $\mathcal{N}_\beta(s)$, $\beta = \{1, \ldots, \beta_{max}\}$, de forma que $\mathcal{N}_\beta(s) \subseteq \mathcal{N}_{\beta+1}(s)$ e $\mathcal{N}(s) = \mathcal{N}_{\beta_{max}}(s)$. Por exemplo, caso $\mathcal{N}$ seja uma vizinhança k-opt, ela pode ser particionada em subvizinhanças 1-opt, 2-opt, $\cdots$, k-opt. A busca em VNDBS é realizada na forma de um VND[65; 66]. Inicialmente aplica-se busca local na vizinhança $\mathcal{N}_1(s)$, e a medida que soluções aprimorantes não são encontradas para a vizinhança $\mathcal{N}_\beta(s)$, VNDBS parte para uma busca local na vizinhança $\mathcal{N}_{\beta+1}(s)$. Entretanto, quando uma solução aprimorante é encontrada, VNDBS volta para a vizinhança $\mathcal{N}_1(s)$. Desta forma, soluções aprimorantes são encontradas mais rapidamente nas vizinhanças de tamanho menor, e as vizinhanças maiores são exploradas menos frequentemente.

Para que as vizinhanças de tamanho maior sejam exploradas de forma eficiente, a busca local é realizada na forma de um algoritmo de *Backtracking*. Dada uma solução corrente $s \in S$, onde S é o conjunto de soluções viáveis para o problema, uma vizinhança qualquer $\mathcal{N}(s)$ é caracterizada por um conjunto de variáveis de decisão $X = \{x_1, \ldots, x_n\}$, onde s_i é o valor atribuído para a variável x_i na solução s e o domínio $dom(x_i)$ de cada variável $x_i \in X$ é o conjunto de valores cuja variável x_i pode

assumir. Por exemplo, em PFLP, X contém uma variável x_i para cada ponto cartográfico $p_i \in P$. O domínio $dom(x_i)$ da variável $x_i \in X$ contém todas as posições candidatas para o ponto cartográfico p_i, exceto a posição candidata s_i que está atribuída a p_i na solução corrente. Em WVCP, X contém uma variável x_i para cada vértice $v_i \in V$ do grafo e o domínio $dom(x_i)$ da variável x_i contém todas as cores com o qual v_i pode ser colorido, exceto aquela com o qual v_i está colorido na solução corrente. A cada chamada recursiva do algoritmo de *backtracking*, uma variável $x_i \in X$ é selecionada. Para cada valor $d \in dom(x_i)$, uma nova chamada recursiva é realizada para um vizinho s', onde s' é idêntico a s exceto pelo valor de $s'_i = d \neq s_i$. A chamada recursiva é realizada mesmo se s' for uma solução inviável ou se $f(s') > f(s)$, uma vez que dependendo da atribuição de valores para as outras variáveis em nós inferiores da árvore de *backtracking*, a viabilidade da solução pode ser restaurada e seu custo reduzido. A profundidade das chamadas recursivas é limitada por um valor calculado em função do parâmetro β_{max}.

Apesar do algoritmo de *backtracking* reduzir consideravelmente o número de vizinhos que precisam ser explicitamente avaliados, o tempo computacional necessário para avaliar todos os vizinhos de uma vizinhança exponencialmente grande $\mathcal{N}_\beta(s)$ pode ainda ser muito grande. Para reduzir ainda mais o tempo computacional da busca, VNDBS seleciona um subconjunto dos vizinhos $\mathcal{N}^\alpha_\beta(s)$ de $\mathcal{N}_\beta(s)$ e realiza busca local apenas neste subconjunto. Isto é implementado selecionando um subconjunto $X^\alpha \subseteq X$ de variáveis de decisão que definem a vizinhança $\mathcal{N}_\beta(s)$ e chamando o algoritmo de *backtracking* apenas para as variáveis em X^α. O número de vizinhos em $\mathcal{N}^\alpha_\beta(s)$ depende de um parâmetro α, $\alpha = \{1, \ldots, \alpha_{max}\}$, de forma que $|\mathcal{N}^\alpha_\beta(s)| \leq |\mathcal{N}^{\alpha+1}_\beta(s)|$ e $\mathcal{N}^{\alpha_{max}}_\beta(s) = \mathcal{N}_\beta(s)$. Uma forma de gerar um subconjunto de vizinhos $\mathcal{N}^\alpha_\beta(s)$ de $\mathcal{N}_\beta(s)$ é, por exemplo, selecionar $\alpha \times |X|$ variáveis em X aleatoriamente. Para cada valor de β de 1 até β_{max}, VNDBS inicialmente realiza a busca local na vizinhança $\mathcal{N}^1_\beta(s)$ e a medida que soluções aprimorantes não são mais encontradas para a vizinhança $\mathcal{N}^\alpha_\beta(s)$, VNDBS parte para uma busca local na vizinhança $\mathcal{N}^{\alpha+1}_\beta(s)$. Entretanto, quando uma solução aprimorante é encontrada, VNDBS volta para a vizinhança $\mathcal{N}^1_\beta(s)$. Desta forma, soluções aprimorantes são encontradas mais rapidamente em subconjuntos menores da vizinhança $\mathcal{N}_\beta(s)$. Da mesma maneira quando soluções aprimorantes não são mais encontradas na vizinhança $\mathcal{N}^{\alpha_{max}}_\beta(s)$, o valor de β é incrementado.

A eficiência em termos de tempo computacional e qualidade de solução de uma heurística VNDBS depende principalmente dos parâmetros α_{max} e β_{max}, assim como do quão eficiente é o algoritmo de backtracking implementado para avaliar a vizinhança $\mathcal{N}^\alpha_\beta$. O pseudo código de VNDBS é apresentado na Figura 4.1. A heurística recebe como parâmetros α_{max}, β_{max}, o conjunto X de variáveis de decisão do problema e uma

solução inicial s, onde s_i denota o valor fixado para a variável x_i em s.

Na Linha 1, as variáveis α e β são inicializadas. O laço das linhas 2-8 é executado enquanto uma condição de parada não seja satisfeita. Na Linha 3, um subconjunto X^α das variáveis X é criado, de forma que a cardinalidade X^α é definida em função de α. Na Linha 4, o algoritmo de *backtracking* é executado para realizar uma busca local na vizinhança $\mathcal{N}_\beta^\alpha(s)$ (definida por X^α e β) e retornar a solução s' cujo custo é menor ou igual ao custo de s. Se o custo da solução s' for menor que o custo da solução corrente s (Linha 5) as variáveis α e β são reinicializadas na Linha 6. Caso contrário, α e β são atualizadas na Linha 7. O processo de atualização destas variáveis é discutido abaixo. A solução corrente é sempre atualizada na Linha 8, uma vez que o algoritmo de *backtracking* sempre retorna uma solução com custo igual ou menor que o custo da solução corrente. A atualização constante da solução corrente é o principal mecanismo de diversificação de VNDBS. Por fim, a melhor solução encontrada é retornada na Linha 9.

```
Procedimento VNDBS(s, X, α_max, β_max)
1  α, β ← 1
2  enquanto condição de parada não satisfeita faça
3     X^α ← Subconjunto(X, α)
4     s' ← Backtracking(s, X^α, β)
5     se f(s') < f(s)
6        então α, β ← 1
7        senão α, β ← Atualizar(α, β, α_max, β_max)
8     s ← s'
9  retorne s
```

Figura 4.1. Pseudo código do procedimento VNDBS.

A atualização das variáveis α e β (Linha 7 da Figura 4.1) pode ser realizada de diferentes maneiras. Um exemplo de como estas variáveis podem ser atualizadas é apresentada na Figura 4.2. O procedimento *Atualizar* recebe como parâmetros o valor corrente de α e β e seus respectivos valores máximos α_{max} e β_{max} e retorna os valores de α e β atualizados. Se o valor de α for menor que α_{max} (Linha 1), o valor de α é incrementado na Linha 2. Caso contrário, α atingiu seu valor máximo e é reinicializado na Linha 4. Neste caso, o valor de β é atualizado nas linhas 5-7. Se o valor de β for menor do que β_{max} (Linha 5), o primeiro é incrementado na Linha 6. Caso contrário, β atingiu seu valor máximo e é reinicializado na Linha 7.

O procedimento Backtracking utilizado em VNDBS é apresentado na Figura 4.3. O procedimento recebe como parâmetros a solução corrente s, o valor corrente de β e o subconjunto X^α de variáveis de decisão. Backtracking efetua a busca local na

```
Procedimento Atualizar(α, β, α_max, β_max)
1  se α < α_max então
2     α ← α + 1
3  senão
4     α ← 1
5     se β < β_max
6        então β ← β + 1
7        senão β ← 1
```

Figura 4.2. Pseudo código do procedimento Atualizar.

vizinhança $\mathcal{N}_\beta^\alpha(s)$, definida por X^α e β. Se $\beta = 0$ (Linha 1) o algoritmo de backtracking retorna a solução corrente, uma vez que a vizinhança $\mathcal{N}_\beta^\alpha(s)$ para $\beta = 0$ é vazia. O laço das linhas 2-6 enumera todos os valores no domínio $dom(x_i)$ de cada variável $x_i \in X^\alpha$. Para cada variável de decisão x_i (Linha 2) e para cada valor d em $dom(x_i)$ (Linha 3), uma solução s' é gerada na Linha 4, de forma que s' é igual a s, exceto pelo valor de $s'_i = d \neq s_i$, onde s_i e s'_i são o valor da variável de decisão x_i nas soluções s e s', respectivamente. Ou seja, s' é o vizinho obtido através da modificação do valor de s_i para d. Na Linha 5, o algoritmo de *backtracking* é executado recursivamente para enumerar todos os valores no domínio $dom(x_j)$ de cada variável $x_j \in X^\alpha \backslash \{x_i\}$ e retorna em s'' a melhor solução encontrada. Se o custo de s'' é menor ou igual que o custo da solução corrente s, esta é atualizada na Linha 6. A melhor solução encontrada é tornada na Linha 7.

```
Procedimento Backtracking(s, X^α, β)
1  se β ≥ 1 então
2     para cada x_i ∈ X^α faça
3        para cada d ∈ dom(x_i) faça
4           s' ← s e s'_i ← d
5           s'' ← Backtracking(s', X^α\{x_i}, β − 1)
6           se f(s'') ≤ f(s) então s ← s''
7  retorne s
```

Figura 4.3. Pseudo código do procedimento Backtracking.

4.1 VND com Backtracking para PFLP

A estratégia de solução proposta para PFLP recebe o grafo particionado $G = (V, E, Q)$ e o conjunto P de pontos cartográficos e retorna uma solução $s \subseteq V$ que

consiste em um conjunto com $|P|$ posições candidatas, uma para cada ponto cartográfico em P. A estratégia consiste em três etapas. Primeiramente, um algoritmo de pré-processamento é aplicado ao grafo de conflitos com o intuito de reduzir o número de vértices e arestas no grafo e consequentemente o tamanho do espaço de busca a ser explorado pelos algoritmos de otimização. Em seguida, uma heurística construtiva gulosa é utilizada para criar uma solução inicial para o problema. Por fim, aplica-se uma heurística de busca baseada na heurística VNDBS, introduzida na seção anterior. Estas três etapas são detalhadas a seguir.

4.1.1 Pré-processamento

O algoritmo de pré-processamento utilizado foi baseado no algoritmo proposto por Wagner et al. [21]. A cada iteração, ele tenta fixar uma posição candidata para um ponto cartográfico. Se o grau de uma posição candidata $v \in Q_p$ em G for igual a zero, esta posição é fixada para $p \in P$ e todas as outras posições candidatas em Q_p são removidas de G. Este procedimento é repetido até que o grau de todas as posições candidatas em V seja maior que zero.

4.1.2 Heurísticas Construtivas

Duas heurísticas construtivas são propostas e avaliadas para o PFLP. A primeira foi denominada Heurística de Construção Aleatória (HA) e a segunda foi chamada de Heurística de Construção Ordenada (HO). A heurística HA (Figura 4.4) seleciona aleatoriamente as posições candidatas para os pontos cartográficos. Na Linha 1, s é inicializada. O laço das linhas $2-4$ seleciona uma posição candidata para cada ponto cartográfico. Na Linha 3, seleciona-se de forma aleatória uma posição candidata $v \in Q_p$ para o ponto cartográfico $p \in P$. Na Linha 4, v é armazenada na solução s e por fim na Linha 5, a solução s é retornada.

```
Heurística HA(G = (V, E, Q))
1  s ← φ
2  para p = 1 até |P| faça
3     v ← Selecione aleatoriamente v ∈ Q_p
4     s ← s ∪ {v}
5  retorne s
```

Figura 4.4. Pseudo código da heurística HA.

A heurística de construção ordenada HO recebe além do grafo de conflitos $G = (V, E, Q)$ uma permutação $\pi = [\pi_1, \ldots, \pi_{|P|}]$ dos pontos cartográficos. Esta permutação define a ordem na qual os pontos cartográficos serão processados pela heurística HO. HO inicia com uma solução vazia e percorre sequencialmente a permutação π, selecionando para cada ponto cartográfico π_i a posição candidata que acarreta o menor incremento no número conflitos da solução.

O pseudo código de HO é apresentado na Figura 4.5. Na Linha 1, a solução s é inicializada. O laço das linhas $2-4$ seleciona uma posição candidata de cada ponto cartográfico. Na Linha 3, a posição candidata $v \in Q_{\pi_i}$ que acarreta o menor incremento no número de rótulos em conflitos é selecionada. Na Linha 4, s recebe a posição candidata v. Por fim, a solução s e retornada na Linha 5.

```
Heurística HO(G = (V, E, Q), π)
1  s ← φ
2  para i = 1 até |P| faça
3     v ← Seleciona a posição candidata v ∈ Q_{π_i} que acarreta
          o menor incremento do número de conflitos
4     s ← S ⋃ {v}
5  retorne s
```

Figura 4.5. Pseudo código da heurística HO.

4.1.3 VNDBS para PFLP

A heurística VNDBS proposta para PFLP foi chamada de VNDBS-PFLP. A heurística recebe como parâmetros α_{max}, β_{max} uma solução inicial s e um conjunto X que contém uma variável de decisão x_i para cada ponto cartográfico $p_i \in P$. O domínio $dom(x_i)$ da variável $x_i \in X$ contém todas as posições candidatas para o ponto cartográfico p_i.

O pseudo código de VNDBS-PFLP é apresentado na Figura 4.6. Na Linha 1, as variáveis α, β e γ são inicializadas. O laço das linhas 2-8 é executado enquanto uma solução com zero conflitos não for encontrada ou um determinado limite de tempo seja atingido. Na Linha 3, é criado um subconjunto $X^\alpha \subseteq X$ com os pontos cartográficos que estão a no máximo $\alpha - 1$ arestas distantes de algum ponto cartográfico em conflito. Ou seja, para $\alpha = 1$, X^α contém apenas os pontos cartográficos em conflitos; para $\alpha = 2$, X^α possui os pontos cartográficos em conflitos e os pontos cartográficos que são vizinhos de algum ponto cartográfico em conflito; para $\alpha = 3$, X^α contém além dos

pontos cartográficos em conflito aqueles que estão a até duas arestas distantes de um ponto cartográfico em conflito, e assim sucessivamente. Na Linha 4, o algoritmo de *backtracking* é executado para realizar uma busca local na vizinhança $\mathcal{N}_{\beta}^{\alpha}(s)$ (definida por X^{α} e β) e retornar a solução s' cujo o número de conflitos é menor ou igual ao número de conflitos de s. Se o o número de conflitos da solução s' for menor que o número de conflitos da solução corrente s (Linha 5) as variáveis α, *beta* e γ são reinicializadas na Linha 6. Caso contrário, α, β e γ são atualizadas na Linha 7. O processo de atualização destas variáveis é discutido abaixo. A solução corrente é sempre atualizada na Linha 8, uma vez que o algoritmo de *backtracking* sempre retorna uma solução com o número de conflitos igual ou menor que o número de conflitos da solução corrente. Por fim, a melhor solução encontrada é retornada na Linha 9.

```
Procedimento VNDBS-PFLP(s, X, α_max, β_max)
1  α, β, γ ← 1
2  enquanto condição de parada não satisfeita faça
3     X^α ← Subconjunto(X, α)
4     s' ← Backtracking(s, X^α, β)
5     se f(s') < f(s)
6        então α, β, γ ← 1
7        senão α, β, γ ← Atualizar(X^α, X, α, β, γ, α_max, β_max)
8     s ← s'
9  retorne s
```

Figura 4.6. Pseudo código da heurística VNDBS-PFLP.

O procedimento *Atualizar* da Figura 4.6 é apresentado na Figura 4.7. O procedimento recebe como parâmetros o conjunto X de pontos cartográficos, o subconjunto $X^{\alpha} \subseteq X$ de pontos cartográficos, o valor corrente de α, β, γ e os valores máximos α_{max} e β_{max} e retorna os valores de α, β e γ atualizados. O número de iterações de VNDBS-PFLP que serão executadas antes dos valores de α e β serem atualizados é definido por γ e γ_{max} é o valor máximo para γ. O valor de γ_{max} é fornecido pelo logaritmo do somatório dos graus dos pontos cartográficos contidos no conjunto X^{α}.

Se o valor de γ for menor que γ_{max} (Linha 1), o valor de γ é incrementado na Linha 2. Caso contrário, γ atingiu seu valor máximo e é reinicializado na Linha 4. Neste caso, o valor de α é atualizado nas linhas 5-8. Se o valor de α for menor que α_{max} e a cardinalidade de X^{α} for menor que a de X (Linha 5), o valor de α é incrementado na Linha 6. Caso contrário, α é reinicializado na Linha 8 porque atingiu seu valor máximo ou todos os pontos cartográficos estão no máximo $\alpha - 1$ arestas distantes de algum ponto cartográfico em conflito. Neste caso, o valor de β é atualizado nas linhas

9-11. Se o valor de β for menor do que β_{max} (Linha 9), o valor de β é incrementado na Linha 10. Caso contrário, β atingiu seu valor máximo e é reinicializado na Linha 11.

```
Procedimento
Atualizar(X^α, X, α, β, γ, α_max, β_max)
1   se γ < γ_max então
2     γ ← γ + 1
3   senão
4     γ ← 1
5     se α < α_max e |X^α| < |X|
6       então α ← α + 1
7     senão
8       α ← 1
9       se β < β_max
10        então β ← β + 1
11      senão β ← 1
```

Figura 4.7. Pseudo código do procedimento Atualizar para PFLP.

O procedimento *Backtracking* da Figura 4.6 é apresentado na Figura 4.8. O procedimento recebe como parâmetros a solução corrente s, o valor corrente de β e o subconjunto X^α de pontos cartográficos. Cada variável $x_i \in X^\alpha$ está associada a um ponto cartográfico $p_i \in P$. Denota-se por s_i, a posição candidata atribuída ao ponto cartográfico p_i em s. O procedimento *Backtracking* efetua a busca local na vizinhança $\mathcal{N}^\alpha_\beta(s)$, definida por X^α e β. Se $\beta < 1$ o algoritmo de backtracking retorna a solução corrente, uma vez que a vizinhança $\mathcal{N}^\alpha_\beta(s)$ para $\beta = 0$ é vazia. O laço das linhas 2-7 enumera todas as posições candidatas no domínio $dom(x_i)$ de cada ponto cartográfico p_i associado a variável $x_i \in X^\alpha$. Para cada ponto cartográfico p_i associado a variável x_i (Linha 2) e para cada posição candidata d em $dom(x_i)$ (Linha 3), uma solução s' é gerada na Linha 4, de forma que s' é igual a s, exceto pelo valor de $s'_i = d \neq s_i$. Ou seja, s' é o vizinho obtido através da modificação da posição candidata de p_i para d. Na Linha 5, é criado um subconjunto $\overline{X}^\alpha \subseteq X^\alpha$ apenas com os pontos cartográficos que estão em conflito com o ponto cartográfico p_i (associado a variável x_i). Na Linha 6, o algoritmo de *backtracking* é chamado recursivamente para enumerar todos os valores no domínio $dom(x_j)$ de cada ponto cartográfico $x_j \in \overline{X}^\alpha \backslash \{x_i\}$ e retorna em s'' a melhor solução encontrada. Se o número conflitos de s'' é menor ou igual que o número de conflitos da solução corrente s, esta é atualizada na Linha 7. A melhor solução encontrada é retornada na Linha 8.

```
Procedimento Backtracking(s, X^α, β)
1  se β ≥ 1 então
2     para cada x_i ∈ X^α faça
3        para cada d ∈ dom(x_i) faça
4           s' ← s e s'_i ← d
5           X̄^α ← Subconjunto(X^α, s'_i)
6           s'' ← Backtracking(s', X̄^α \ {x_i}, β − 1)
7           se f(s'') ≤ f(s) então s ← s''
8  retorne s
```

Figura 4.8. Pseudo código do procedimento Backtracking para PFLP.

4.1.4 Experimentos Computacionais

Os experimentos computacionais foram realizados em dois grupos de instâncias. O primeiro grupo foi proposto por Lorena [46], sendo composto por quatro conjuntos de instâncias com 250, 500, 750 e 1000 pontos cartográficos, cada conjunto, possuindo 25 instâncias por conjunto. O segundo grupo foi proposto em Alvim e Taillard [3] e é composto por dois conjuntos, ambos contendo 20 instâncias com 13206 pontos cartográficos. As instâncias do primeiro conjunto possuem 4 posições candidatas por ponto cartográfico e as instâncias do segundo conjunto possuem 8 posições candidatas por ponto cartográfico.

Para avaliar a convergência da heurística VNDBS-PFLP foram utilizadas quatro heurísticas construtivas diferentes como ponto de partida: a heurística HA e três versões da heurística HO, HOC, HOD e HOM. HOC recebe uma permutação dos pontos cartográficos ordenados em ordem crescente da cardinalidade dos conjuntos $Q_i \in Q$. HOD recebe uma permutação do pontos cartográficos ordenados em ordem decrescente da cardinalidade dos conjuntos $Q_i \in Q$. Já HOM recebe uma permutação dos pontos cartográficos ordenados em ordem decrescente da média do número de pontos cartográficos adjacentes as posições candidatas $v \in Q_i$, $Q_i \in Q$.

As heurísticas HA-VNDBS, HOC-VNDBS, HOD-VNDBS e HOM-VNDBS denotam respectivamente as heurísticas HA, HOC, HOD e HOM descritas na seção 4.1.2 seguidas da aplicação da heurística VNDBS-PFLP descrita na Seção 4.1.3. As quatro heurísticas foram implementadas em linguagem C e compiladas com o compilador GCC V4.4.1. Os experimentos foram realizados em um Pentium IV 3.0 Ghz com 1 Gb de memória RAM e sistema operacional Linux SlackWare. As quatro heurísticas foram executadas 10 vezes para cada instância, variando-se a semente do gerador de números aleatórios. Experimentos preliminares indicaram que o melhor valor para os parâmetros α_{max} e β_{max} foram 10 e 2 respectivamente. O valor de γ_{max} é fornecido

pelo logaritmo do somatório dos graus dos pontos cartográficos contidos no conjunto X^{α}. O tempo limite de cada execução foi fixado em 120 segundos.

O primeiro experimento computacional visa identificar qual das versões de VNDBS propostas obtém os melhores resultados para PFLP. A Figura 4.9 apresenta o gráfico da média do custo das soluções encontradas pelas quatro heurísticas propostas em função do tempo, para 10 execuções das 25 instâncias de Lorena [46] com 1000 pontos cartográficos. Já as figuras 4.10 e 4.11 apresentam os gráficos da média do custo das soluções encontradas pelas quatro heurísticas propostas em função do tempo, para 10 execuções das 20 instâncias de Alvim e Taillard [2] com 4 posições candidatas por ponto cartográfico (Figura 4.10) e com 8 posições candidatas por ponto cartográfico (Figura 4.11). Pode-se observar que nos três gráficos às heurísticas de construção ordenada HOC, HOD e HOM produziram soluções iniciais melhores que a heurística de construção aleatória HA e que a heurística HOC forneceu em média as melhores soluções iniciais dentre as heurísticas construtivas estudadas. Pode-se observar também que nos três gráficos, independente da solução inicial VNDBS-PFLP convergiu para a melhor solução conhecida antes de 120 segundos.

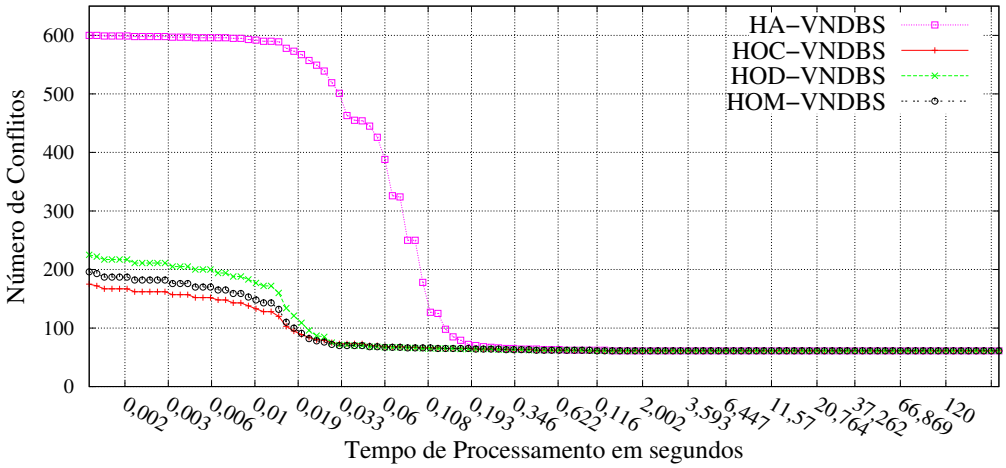


Figura 4.9. Gráfico da média do custo da melhor solução encontrada pelas heurísticas HA-VNDBS, HOC-VNDBS, HOD-VNDBS e HOM-VNDBS, em função do tempo de processamento, para as instâncias propostas em Lorena [46], com 1000 pontos cartográficos e 4 posições candidatas.

O segundo experimento computacional compara a heurística HOC-VNDBS com os melhores métodos descritos na literatura para as instâncias propostas por Lorena [46]. Na Tabela 4.1 a coluna 1 apresenta o nome das heurísticas e seus autores. As colunas $2-5$ apresentam o custo para cada grupo de instâncias com 250, 500, 750 e 1000 pontos cartográficos respectivamente.

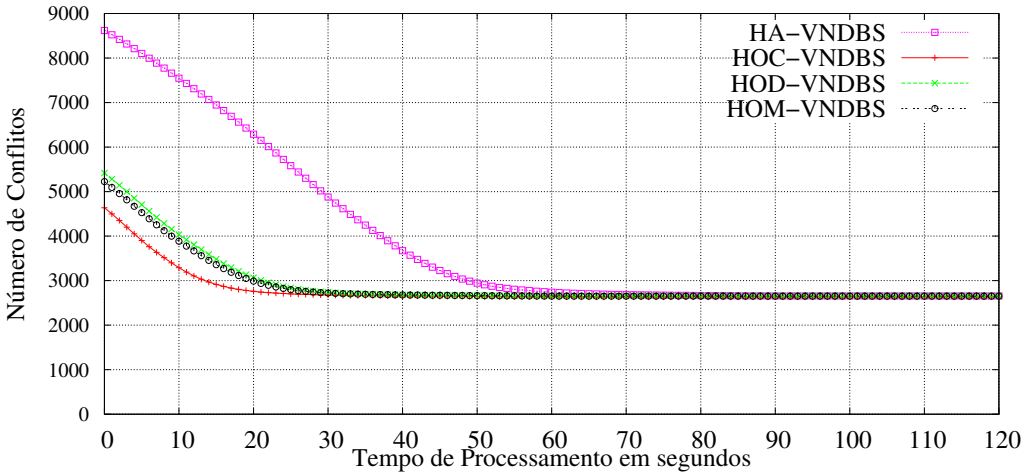


Figura 4.10. Gráfico da média do custo da melhor solução encontrada pelas heurísticas HA-VNDBS, HOC-VNDBS, HOD-VNDBS e HOM-VNDBS, em função do tempo de processamento, para as instâncias propostas em Alvim e Taillard [3], para 4 posições candidatas por ponto.

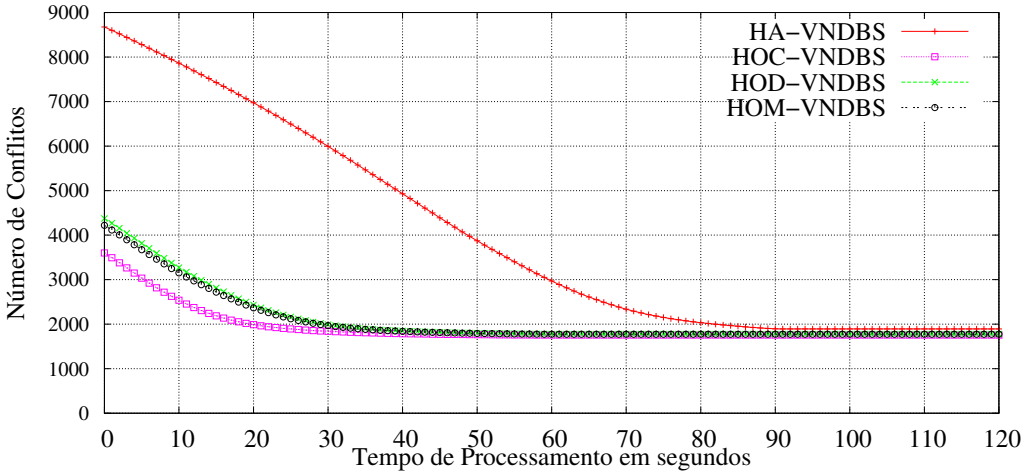


Figura 4.11. Gráfico da média do custo da melhor solução encontrada pelas heurísticas HA-VNDBS, HOC-VNDBS, HOD-VNDBS e HOM-VNDBS, em função do tempo de processamento, para as instâncias propostas em Alvim e Taillard [3], para 8 posições candidatas por ponto.

Na primeira linha é apresentado a média dos resultados alcançados pela heurística VNDBS para 10 execuções em 120 segundos. Os resultados alcançados por Ribeiro et al. [33] foram extraídos da Tabela 4. Neste trabalho os autores avaliaram a formulação de Mauri [34] e identificaram 2 cortes correlacionados e baseados na vizinhança de cada vértice do grafo de conflitos. Em Mauri [34] os dados foram extraídos da Tabela 6.5 e de Oliveira et al. [11], Tabela 3. Os resultados da Heurística POPMUSIC de Alvim e Taillard [3] foram extraídos das tabelas 2 e 3.

Ribeiro e Lorena [32] reportaram os resultados alcançados pelo método LagClus nas tabelas $2, 3$ e 4. A heurística GRASP de cravo et al. [28], Tabela 9. Yamamoto e Yamamoto et al. reportaram os resultados alcançados pelo algoritmo genético construtivo [51] nas tabelas 1 e 2, da heurística FALP [50] na Tabela 1 e da busca TABU [49] na Tabela 3.

Os resultados alcançados pelo Algoritmo Genético com Máscara de Verner et al. [59], Algoritmo Genético de Verner et al. [59], Zoraster [76], Hirsch [77] e das heurísticas Simulated Annealing, 3-opt Gradient Descent, 2-opt Gradient Descent, Gradient Descent e Algoritmo Guloso publicados em Christensen et al. [43] foram extraídos da Tabela 7 página 19 de Verner et al. [59].

Em [11; 28] foi calculado a média dos resultados alcançados para 10 execuções. Em [32; 33; 34] os resultados foram alcançados por métodos exatos e no restante foi calculado a média dos resultados alcançados para 25 execuções para cada heurística.

Pode-se observar que a porcentagem média de vértices em conflito das soluções obtidas com HOC-VNDBS é sempre igual a dos melhores algoritmos da literatura para as instâncias com 250, 500 e 750 vértices e um pouco melhor que estes para as instâncias com 1000 vértices.

O terceiro experimento computacional compara a heurística HOC-VNDBS com a heurística POPMUSIC [3]. Os resultados deste experimento são apresentados na Tabela 4.2. As colunas 1, 2 e 3 apresentam a altura do rótulo, o comprimento do rótulo e o número de posições candidatas por ponto cartográfico respectivamente. Para cada tamanho de rótulo são apresentadas duas linhas, uma linha para 4 e outra para 8 posições candidatas por ponto cartográfico respectivamente. As colunas $4 - 5$ apresentam o percentual de rótulos em conflitos e o tempo em segundos demandado obtidos da Tabela 4 coluna 8 de Alvim e Taillard [3]. As colunas $6 - 14$ apresentam os resultados alcançados pela heurística HOC-VNDBS para 1, 2, 3, 4, 5, 10, 15, 20 e 120 segundos de execução respectivamente. Pode-se observar em **negrito** na coluna $1s$, que em 1 segundo de processamento a porcentagem de rótulos em conflitos alcançados pela heurística HOC-VNDBS é melhor que os resultados alcançados por Alvim e Taillard [3], para todos os problemas com *quatro* posições candidatas por ponto cartográfico e para *seis* problemas com *oito* posições candidatas. Para os 14 problemas restantes em **negrito** na coluna $2s$, a heurística HOC-VNDBS supera os resultados alcançados por Alvim e Taillard [3] em *dois* segundos de processamento.

Algoritmo	Percentual de conflitos por grupo de instâncias			
	250	500	750	1000
Heurística VNDBS para o PFLP	0,00	0,32	2,04	6,13
$PNMRLC_{C2}$ de Ribeiro et al. [33]	0,00*	0,32*	2,04	6,14
$PNMRLC_{C1}$ Ribeiro et al. [33]	0,00*	0,32*	2,04	6,14
CPLEX com a Formulação de Mauri [34]	0,00*	0,32*	2,09	9,81
$DC_{\alpha\beta}PMNRSC^n$ de Mauri [34]	0,00*	0,32*	2,04*	6,26
Heurística ILS de Oliveira et al. [11]	0,00	0,32	2,05	6,24
Pop(asc) de Alvim e Taillard [3]	0,00	0,33	2,28	7,32
Pop(30) de Alvim e Taillard [3]	0,00	0,33	2,28	7,46
Pop(70) de Alvim e Taillard [3]	0,00	0,33	2,27	7,42
LagClus de Ribeiro e Lorena [32]	0,00	0,33	2,35	8,58
GRASP(RCL = 6) de Cravo et al. [28]	0,00	0,33	2,28	7,80
Algoritmo Genético Construtivo de Yamamoto et al. [51]	0,00	0,40	2,90	9,30
FALP de Yamamoto et al. [50]	0,00	0,50	3,30	9,88
Busca Tabu de Yamamoto et al. [49]	0,00	0,74	3,24	10,00
Algoritmo Genético com Máscara de Verner et al. [59]	0,02	1,21	4,01	11,04
Algoritmo Genético de Verner et al. [59])	1,60	7,41	17,62	34,30
Simulated Annealing de Christensen et al. [43]	0,10	1,70	7,70	17,91
Zoraster [76]	0,21	3,79	20,22	46,94
Hirsch [77]	0,42	4,30	17,96	39,76
3-opt Gradient Descent de Christensen et al. [43]	0,24	2,66	10,56	22,17
2-opt Gradient Descent de Christensen et al. [43]	0,64	4,38	14,40	26,63
Gradient Descent de Christensen et al. [43]	4,53	13,54	27,60	41,71
Algoritmo Guloso de Christensen et al. [43]	11,18	24,85	41,43	56,59

Tabela 4.1. Comparação com algoritmos da literatura em número de rótulos em conflitos para as instâncias propostas por Lorena [46]. * Indica que a solução ótima foi encontrada para todas as instâncias do grupo.

4.1.5 Considerações Finais

Avaliando os resultados dos experimentos computacionais, pode-se concluir que as heurísticas propostas neste capítulo encontraram soluções equivalentes ou melhores que as soluções das melhores heurísticas da literatura em um tempo computacional menor. A heurística VNDBS mostrou-se robusta para as instâncias avaliadas, dado que para as instâncias propostas por Lorena [46] chegou aos mesmos valores de soluções em 120 segundos, independentemente da qualidade da solução inicial fornecida pela heurística construtiva.

O desvio padrão para a média dos resultados alcançados de 10 execuções pelas heurísticas HA-VNDBS, HOD-VNDBS, HOC-VNDBS e HOM-VNDBS para as ins-

tâncias propostas por Alvim e Taillard [3] com 4 e 8 posições candidatas por ponto cartográfico foi de $1,35$ e $6,49$ respectivamente.

Para as instâncias propostas por Alvim e Taillard [3] as heurísticas HA-VNDBS, HOD-VNDBS e HOM-VNDBS alcançaram em $99,9\%$ dos casos os resultados alcançados pela heurística HOC-VNDBS.

4.2 VND com Backtracking para WVCP

A estratégia da heurística proposta para o WVCP consiste em duas etapas: Primeiramente, uma heurística construtiva gulosa é utilizada para criar uma solução inicial para o problema. Em seguida, aplica-se uma heurística de melhoria baseada na heurística VNDBS. Estas duas etapas são detalhadas a seguir.

4.2.1 Heurística Construtiva

A heurística construtiva proposta, detalhada na Figura 4.12, recebe o grafo $G = (V, E)$ e o vetor $\pi = [\pi_1, \ldots, \pi_{|V|}]$, uma permutação dos vértices que descreve a ordem na qual eles serão considerados. O vetor π é processado sequencialmente selecionando os vértices v_{π_r} e inserindo no conjunto independente $C_t \subseteq V$ que acarreta o menor custo para a solução s, retornando ao final a melhor solução encontrada s. A heurística construtiva foi avaliada com a lista de vértices selecionada em ordem crescente do grau dos vértices, em ordem decrescente do grau dos vértices e com os vértices dispostos de forma aleatória, denotando heurística construtiva crescente HOC, heurística construtiva decrescente HOD e heurística construtiva aleatória HA.

Na Linha 1, a solução s é inicializada. O laço das linhas 2-5 seleciona os vértices $v \in V$ e insere em um conjunto independente $C_t \in S$. Na Linha 3, é selecionado o conjunto independente $C_t \in s$ para o vértice v_{π_r} que acarreta o menor custo para a solução s. Na Linha 4, o vértice v_{π_r} é inserido em C_t. Na Linha 5, o conjunto independente C_t é atualizado na solução s e por fim na Linha 6, a melhor solução s é retornada.

4.2.2 VNDBS para WVCP

A vizinhança aqui proposta para o WVCP é uma função que mapeia uma solução s em um conjunto de soluções $\mathcal{N}_\beta^\alpha(s)$ trocando na solução s as cores de um conjunto de vértices $V' \subseteq V$. As estruturas de vizinhanças $\mathcal{N}_\beta^\alpha(s), \beta = 1, \ldots, \beta_{max}$, são obtidas

```
Procedimento HOC(G = (V, E), π)
1  s ← ∅
2  para r = 1 até |V| faça
3     C_t ← ConjuntoIndependenteDeMenorCusto(G = (V, E), S, v_{π_r})
4     C_t ← C_t ∪ {v_{π_r}}
5     s ← S ∪ C_t
6  retorne s
```

Figura 4.12. Pseudo código da heurística construtiva.

realizando a substituição de cores nos vértices adjacentes a $v \in V'$, em β níveis pelo algoritmo de *Backtracking*.

O pseudo código de VNDBS é apresentado na Figura 4.13. A heurística recebe como parâmetros α_{max}, β_{max}, uma solução inicial s e o conjunto de vértices V, de forma que s é uma partição do conjunto V em conjuntos independentes $C_r, r = 1, ..., k$ consistindo de uma atribuição de valores para cada variável em v e o domínio $dom(v)$ de cada variável $v \in V$ é o número de cores $k \in \mathbb{N}^*$.

A função *Subconjunto* retorna um conjunto de vértices $X^\alpha \subseteq V$ de acordo com o parâmetro α, descrito na Seção 4.2.3, que determina a cardinalidade do conjunto X^α e a forma a qual os vértices $v \in V$ são selecionados para o subconjunto X^α. Na Linha 1, as variáveis α, β e γ são inicializadas. O laço das linhas 2-9 seleciona um subconjunto de vértices $X^\alpha \subseteq V$ em função do valor de α, efetua a busca local na vizinhança $\mathcal{N}^\alpha_\beta(s)$ e atribui a s as soluções s', até que a condição de parada seja satisfeita. Na Linha 3, é criado um subconjunto de vértices $X^\alpha \subseteq V$ em função do valor de α, onde α determina a cardinalidade de X^α e a forma a qual os vértices $v \in V$ são selecionados para o subconjunto X^α. Na Linha 5, o algoritmo de *backtracking* é executado para realizar uma busca local na vizinhança $\mathcal{N}^\alpha_\beta(s)$ (definida por X^α e β) e retornar a solução s' cujo o custo é menor ou igual ao custo de s. Se o custo da solução s' for menor que o custo da solução corrente s (Linha 5) as variáveis α, β e γ são reinicializadas na Linha 6. Caso contrário, α, β e γ são atualizadas na Linha 7. O processo de atualização destas variáveis é discutido abaixo. A solução corrente é sempre atualizada na Linha 8, uma vez que o algoritmo de *backtracking* sempre retorna uma solução com o custo igual ou menor que o custo da solução corrente. A atualização constante da solução corrente é um mecanismo de diversificação de VNDBS. Por fim, a melhor solução encontrada é retornada na Linha 9.

O procedimento *Atualizar* apresentada na Figura 4.14 recebe como parâmetros o valor corrente de α, β, γ e os valores máximos α_{max} e β_{max} e retorna os valores de

```
Procedimento VNDBS-WVCP(s, V, α_max, β_max)
1  α, β, γ ← 1
2  enquanto condição de parada não satisfeita faça
3     X^α ← Subconjunto(V, α)
4     s' ← Backtracking(s, V, X^α, β)
5     se f(s') < f(s)
6        então α, β, γ ← 1
7        senão α, β, γ ← Atualizar(α, β, γ, α_max, β_max)
8     s ← s'
9  retorne s
```

Figura 4.13. Pseudo código da heurística VNDBS para o WVCP.

α, β e γ atualizados. O número de iterações de VNDBS-WVCP que serão executadas antes dos valores de α e β serem atualizados é definido por γ e γ_{max} é o valor máximo para γ. O valor máximo para γ é calculado pela expressão $\gamma_{max} = \frac{|V|}{\log(((\beta-1)*k)+1)+1}$, onde γ_{max} assume o maior valor quando $\beta = 1$ e o menor valor quando $\beta = \beta_{max}$ para um número de cores k. Se o valor de γ for menor que γ_{max} (Linha 1), o valor de γ é incrementado na Linha 2. Caso contrário, γ atingiu seu valor máximo e é reinicializado na Linha 4. Neste caso, o valor de α é atualizado nas linhas 5-8. Se o valor de α for menor que α_{max} (Linha 5), o valor de α é incrementado na Linha 6. Caso contrário, α atingiu seu valor máximo e é reinicializado na Linha 8. Neste caso, o valor de β é atualizado nas linhas 9-11. Se o valor de β for menor do que β_{max} (Linha 9), o valor de β é incrementado na Linha 10. Caso contrário, β atingiu seu valor máximo e é reinicializado na Linha 11, reiniciando o VNDBS na menor vizinhança de s.

```
Procedimento Atualizar(α, β, γ, α_max, β_max)
1  se γ < γ_max então
2     γ ← γ + 1
3  senão
4     t ← 1
5     se α < α_max
6        então α ← α + 1
7     senão
8        α ← 1
9        se β < β_max
10          então β ← β + 1
11       senão β ← 1
```

Figura 4.14. Pseudo código do procedimento Atualizar para o WVCP.

O procedimento *Backtracking* apresentado na Figura 4.15 recebe como parâme-

tros o conjunto de vértices V, a solução corrente s, o valor corrente de β e o subconjunto X^α de vértices. O procedimento *Backtracking* efetua a busca local na vizinhança $\mathcal{N}_\beta^\alpha(s)$, definida por X^α e β. Se $\beta < 1$ o algoritmo de backtracking retorna a solução corrente, uma vez que a vizinhança $\mathcal{N}_\beta^\alpha(s)$ é vazia. O laço das linhas 2-7 enumera todas as cores $dom(x_i)$ de cada vértice $x_i \in X^\alpha$. Para cada vértice x_i (Linha 2) e para cor d em $dom(x_i)$, uma solução s' é gerada na Linha 4, de forma que s' é igual a s, exceto pelo valor de $s'_i = d \neq s_i$, onde s_i e s'_i são as cores do vértice x_i nas soluções s e s', respectivamente. Ou seja, s' é o vizinho obtido através da modificação da cor de s_i para d. Na Linha 5, é criado um subconjunto $\overline{X}^\alpha \subseteq V$ com os vértices inviabilizaram a solução s' pela atribuição de d a s_i. Na Linha 6, o algoritmo de *backtracking* é chamado recursivamente para enumerar todos os valores no domínio $dom(x_j)$ de cada vértice $x_j \in \overline{X}^\alpha \backslash \{x_i\}$ e retorna em s'' a melhor solução encontrada. Se o custo de s'' é menor ou igual que o custo da solução corrente s, esta é atualizada na Linha 7. A melhor solução encontrada é tornada na Linha 8.

```
Procedimento Backtracking(s, V, X^α, β)
1  se β ≥ 1 então
2     para cada x_i ∈ X^α faça
3        para cada d ∈ dom(x_i) faça
4           s' ← s e s'_i ← d
5           X̄^α ← Subconjunto(V, s'_i)
6           s'' ← Backtracking(s', V, X̄^α \ {x_i}, β − 1)
7           se f(s'') ≤ f(s) então s ← s''
8  retorne s
```

Figura 4.15. Pseudo código do procedimento Backtracking para o PFLP.

4.2.3 Experimentos Computacionais

Os experimentos computacionais foram realizados em um conjunto de instâncias proposto por Trick [52]. As heurísticas HA-VNDBS, HOC-VNDBS e HOD-VNDBS foram estudadas, onde HA-VNDBS, HOC-VNDBS e HOD-VNDBS denotam as heurísticas HA, HOC, e HOD seguidas da aplicação da heurística VNDBS descrita na Seção 4.2. As heurísticas foram implementadas em linguagem C e compiladas com o compilador GCC V4.4.1.

Os experimentos foram realizados em um PC com processador Pentium IV 3.0 Ghz, 1 Gb de memória RAM e sistema operacional Linux SlackWare. O programa *benchmark* ($dfmax$) [52] juntamente com a instância ($r500.5$) fornece uma métrica que permite uma comparação aproximada dos tempos demandados para o processa-

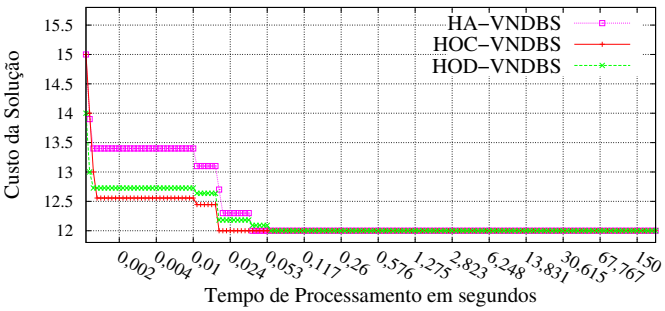
mento dos problemas de coloração em máquinas diferentes. Por exemplo, dado uma instância de coloração I, b_1, t_1, b_2 e t_2 são o *benchmark* e o tempo demandado para o processamento da instância I em duas máquinas diferentes m_1 e m_2 respectivamente. A expressão $t'_1 = \frac{b_1 * t_2}{b_2}$ fornece uma proporção do tempo t_1 na máquina m_2, permitindo a comparação do tempo de processamento entre a heurística que foi executada na máquina m_1 com a heurística que foi executada na máquina m_2. A máquina utilizada nos experimentos computacionais realizados alcançou um *benchmark* ($dfmax$) [52] de 8 segundos.

As heurísticas foram executadas 10 vezes para cada instância, variando-se a semente do gerador de números aleatórios, sendo fixado o tempo limite de cada execução em 150 segundos, sendo armazenado para cada instância o custo da melhor solução encontrada e seu tempo computacional.

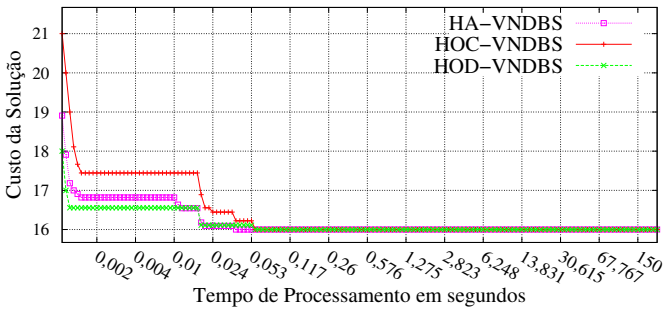
O parâmetro β_{max} foi definido em 2 níveis para o algoritmo de *Backtracking*. O parâmetro α_{max} define a cardinalidade do conjunto V' e a forma a qual os vértices $v \in V$ são selecionados para o conjunto V'.

O parâmetro α_{max} foi definido em 10 unidades e três faixas, onde cada unidade representa 2.5% do número de vértices da instância e cada faixa a forma a qual os vértices serão selecionados do conjunto V na Figura 4.13 onde: Para $\alpha = 1, \ldots, 10$ a função $Subconjunto$ seleciona $\alpha * 2.5\%$ de vértices do conjunto V de forma aleatória, para $\alpha = 11, \ldots, 20$ a função $Subconjunto$ seleciona aleatoriamente os subconjuntos $C_r \in s, r \in \{1, \ldots, k\}$ e depois seleciona um número aleatório de vértices $v \in C_r$ em ordem decrescente dos pesos até totalizar $(\alpha - 10) * 2.5\% * |V|$ vértices e para $\alpha = 21, \ldots, 30$ a função $Subconjunto$ seleciona aleatoriamente os subconjuntos $C_r \in s, r \in \{1, \ldots, k\}$ e depois seleciona os vértices de maior peso até totalizar $(\alpha - 20) * 2.5\% * |V|$ vértices.

As figuras 4.16 e 4.17 apresentam o custo da solução em função do tempo para as instâncias GEOM30b, GEOM40b, GEOM50b e GEOM60b respectivamente. As figuras de 4.18 a 4.24 apresentam o GAP do custo das soluções em um momento t em relação ao custo da melhor solução encontrada s^*, sendo o GAP g^t calculado pela expressão $g^t = \frac{f(S^t) - f(S^*)}{f(S^t)}, t = 1, \ldots, t_{max}, t_{max} = 150$. O GAP g^t foi utilizado para normalizar os valores alcançados pela heurística VNDBS para as instâncias avaliadas O cálculo do GAP do custo das soluções para um grupo de instâncias foi efetuado calculando-se a média aritmética dos $GAP's$ g^t para todas as instâncias do grupo.

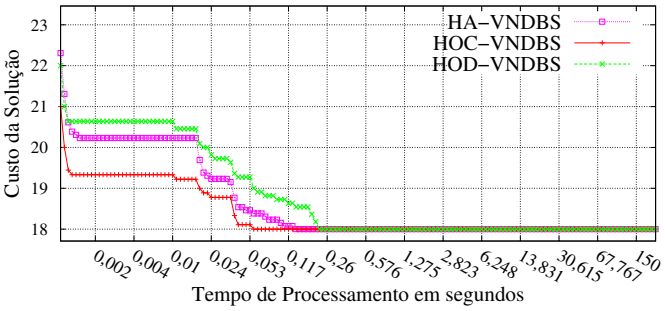


(a) Instância GEOM30b.

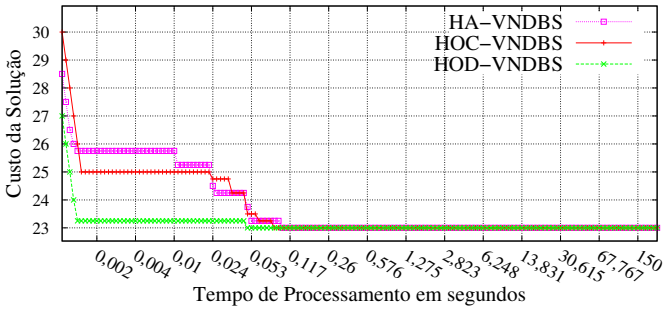


(b) Instância GEOM40b.

Figura 4.16. Custos das soluções para as instâncias GEOM30b e GEOM40b.



(a) Instância GEOM50b.

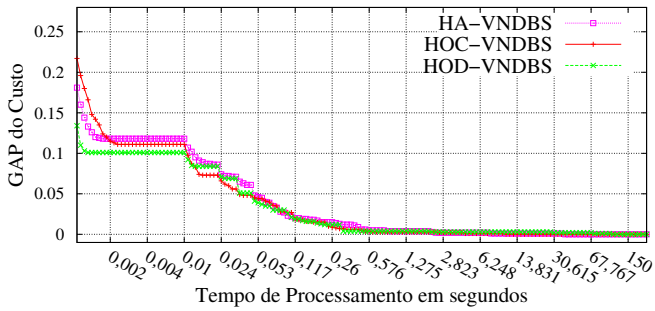


(b) Instância GEOM60b.

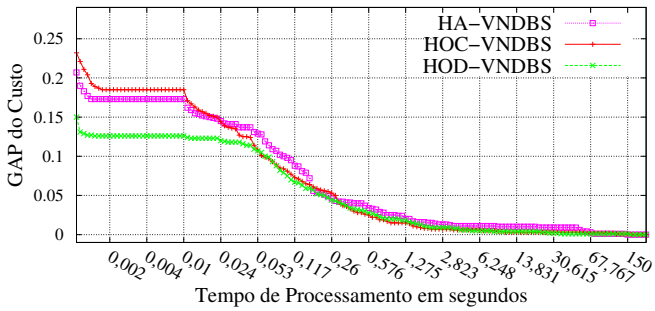
Figura 4.17. Custos das soluções para as instâncias GEOM50b e GEOM60b.

As figuras de 4.18 a 4.22 apresentam o *GAP* do custo das soluções para os conjuntos de instâncias com prefixo R50, R75, GEOM70, GEOM80, GEOM90, GEOM100, R100, GEOM110, GEOM120 e DSJC125 respectivamente.

As figuras 4.23(a), 4.23(b) e 4.24(a) apresentam o *GAP* do custo das soluções para as instâncias com o número de vértices entre 30 a 75, 80 a 100 e 110 a 125 respectivamente. A figura 4.24(b) apresenta o *GAP* do custo das soluções em relação a todas as instâncias avaliadas.

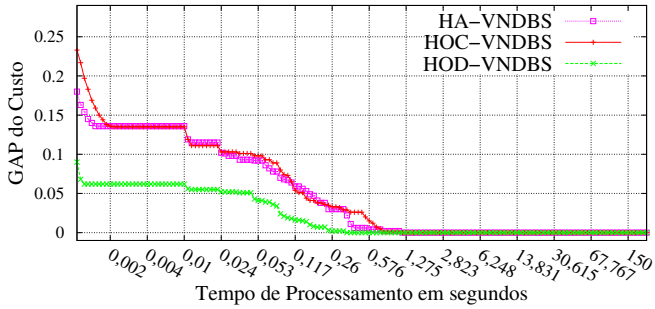


(a) Conjunto de instâncias R50.

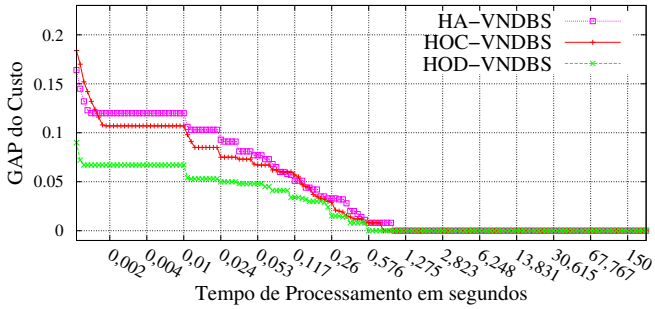


(b) Conjunto de instâncias R75.

Figura 4.18. *GAP's* dos custos das soluções para os conjuntos de instâncias de prefixo R50 e R75.

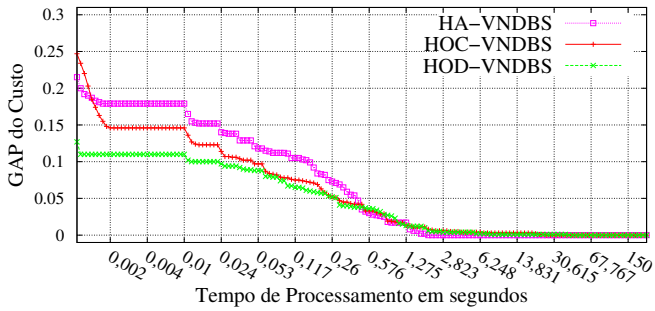


(a) Conjunto de instâncias GEOM70.

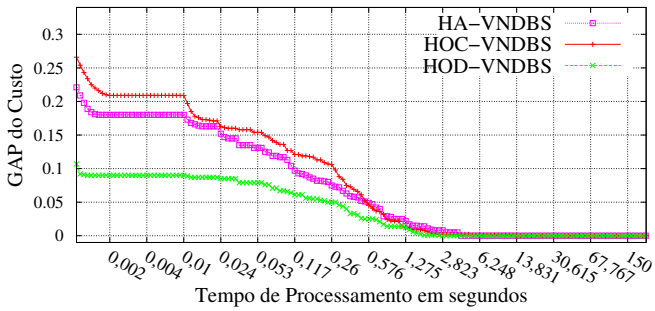


(b) Conjunto de instâncias GEOM80.

Figura 4.19. *GAP's* dos custos das soluções para os conjuntos de instâncias de prefixo GEOM70 e GEOM80.



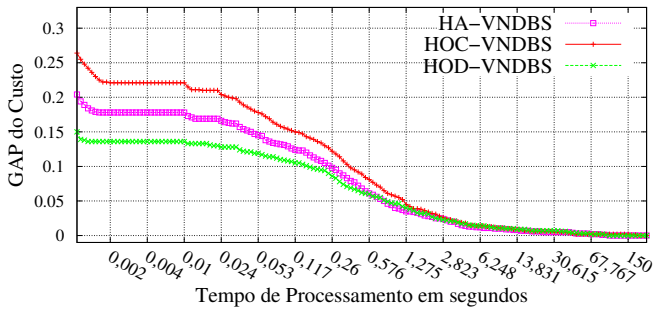
(a) Conjunto de instâncias GEOM90



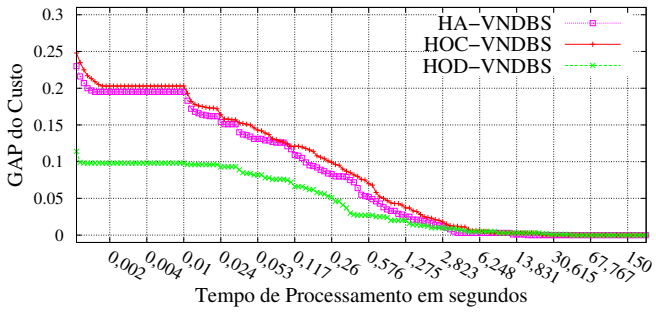
(b) Conjunto de instâncias GEOM100

Figura 4.20. *GAP's* dos custos das soluções para os conjuntos de instâncias GEOM90 e GEOM100.

Pode-se observar que as heurísticas construtivas HOD e HA produzem soluções iniciais melhores que a heurística HOC e que a heurística HOD obteve os melhores resultados na média aritmética dentre as heurísticas construtivas estudadas. Pelas figuras de 4.16 a 4.24 pode-se observar que a heurística tende a uma mesma característica no processo de exploração no espaço de busca, independente das soluções iniciais

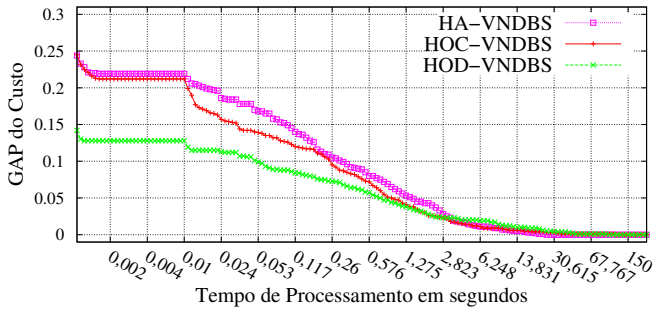


(a) Conjunto de instâncias R100.

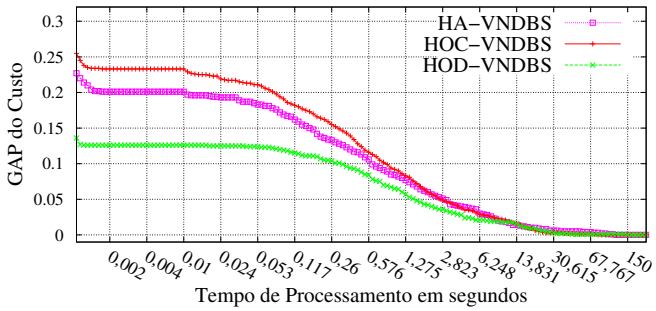


(b) Conjunto de instâncias GEOM110.

Figura 4.21. *GAP's* dos custos das soluções para os conjuntos de instâncias R100 e GEOM110.



(a) Conjunto de instâncias GEOM120.

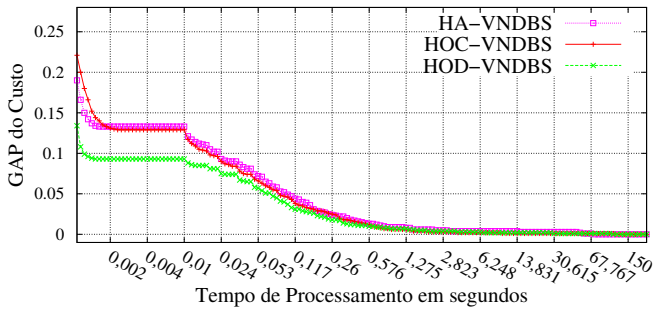


(b) Conjunto de instâncias DSJC125.

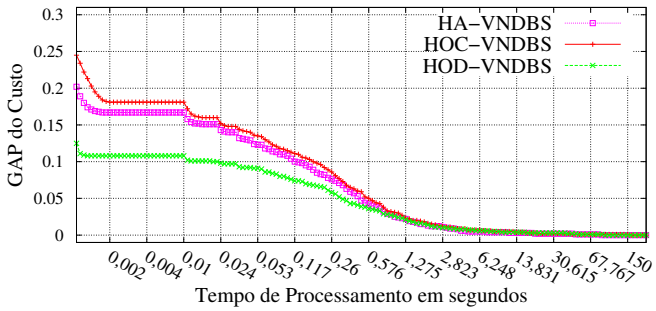
Figura 4.22. *GAP's* dos custos das soluções para os conjuntos de instâncias GEOM120 e DSJC125.

fornecidas pelas heurísticas construtivas.

A Tabela 4.3 apresenta as características das soluções para as heurísticas HA-VNDBS, HOC-VNDBS e HOD-VNDBS onde o nome da instância é definido na coluna 1. As colunas 2, 6 e 10 apresentam o custo médio, as colunas 3, 7 e 11 apresentam a moda, as colunas 4, 8 e 12 apresentam o desvio padrão e as colunas 5, 9 e 13 o coeficiente de variância para cada heurística, HOC, HOD e HA seguida da aplicação da heurística VNDBS descrita na seção 4.2. Pode-se observar que o desvio padrão é zero em 31 instâncias para a heurística HOC-VNDBS, 33 instâncias para a heurística HOD-VNDBS e em 26 instâncias para a heurística HA-VNDBS demonstrando uma baixa dispersão para os resultados alcançados. Observa-se também que o coeficiente de variância é baixo atingido no máximo 40% da amostra para a instância $DSJC125\_1g$. Pode-se observar também que a heurística proposta na média converge para soluções com um GAP inferior a $1,3\%$ antes de 150 segundos, independente das soluções iniciais fornecidas pelas heurísticas construtivas .

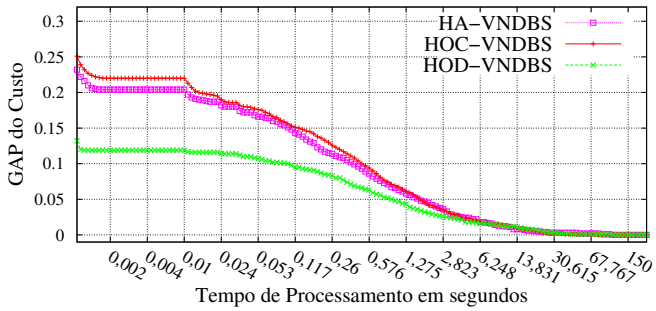


(a) Conjunto de instâncias de 30 a 75 vértices.

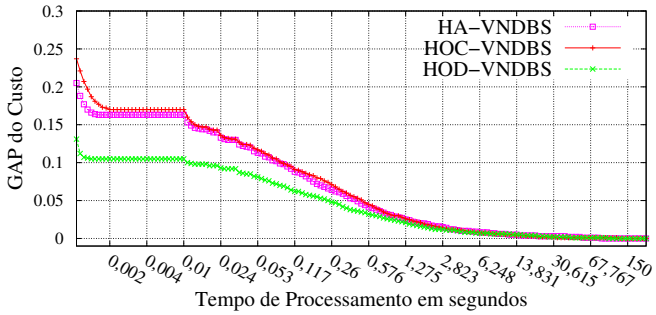


(b) Conjunto de instâncias de 80 a 100 vértices.

Figura 4.23. *GAP's* dos custos das soluções para os conjuntos de instâncias de 30 a 75 e 80 a 100 vértices.



(a) Conjunto de instâncias de 110 a 125 vértices.



(b) Conjunto com todas instâncias avaliadas.

Figura 4.24. *GAP's* dos custos das soluções para o conjunto de instâncias de 110 a 125 vértices e para todas as instâncias avaliadas.

A Tabela 4.4 apresenta a comparação entre a heurística proposta neste trabalho e a heurística proposta por Malaguti et al. [18]. O nome das instâncias é definido na coluna 1. As colunas $2, 3$ e 4 apresentam o tempo computacional em segundos para cada heurística, HOC, HOD e HA seguida da aplicação da heurística VNDBS descrita na seção 4.2. As colunas $5, 6$ e 7 apresentam o custo médio da solução encontrada para cada heurística, HOC, HOD e HA seguida da aplicação da heurística VNDBS descrita na seção 4.2. Os resultados alcançados pela heurística proposta por Malaguti et al. [18] são apresentados nas coluna $8, 9$ e 10 sendo respectivamente o custo, o *Lower Bound* e o tempo computacional total em segundos. Os valores em negrito denotam os melhores resultados alcançados entre heurísticas avaliadas, com custo igual ou inferior aos resultados alcançados por Malaguti et al. [18]. Observamos também que as heurísticas HOC-VNDBS, HOD-VNDBS e HA-VNDBS demandam de tempos distintos para o processamento das instâncias. Os custos computacionais apresentados pelas heurísticas HOC-VNDBS, HOD-VNDBS e HA-VNDBS são inferiores a $\frac{1}{5}$ da

média demandada pela heurística proposta por Malaguti et al. [18]. Comparando os resultados alcançados por Malaguti et al. [18], as médias dos custos das soluções geradas pelas heurísticas HOC-VNDBS e HA-VNDBS, são maiores em $0,30\%$ e $0,30\%$ respectivamente e menor para a média da heurística HOD-VNDBS em $0,07\%$. No geral comparando os resultados alcançados por Malaguti et al. [18] a heurística proposta fornece soluções de custo inferior ou igual em 44 instâncias para a heurística HOD-VNDBS e 40 para as heurística HOC-VNDBS e HA-VNDBS.

No trabalho de Malaguti et al. [18] a pos-otimização foi implementada em ANSI C e o restante da solução proposta em $FORTRAN77$, ambos compilados com todas opções de otimização habilitadas. As relaxações LP foram resolvidas utilizando $CPLEX$ 9.0. A configuração utilizada foi um Pentium IV 2.4 Ghz, 512 MB de memória RAM e sistema operacional Windows XP, obtendo um desempenho de 7 segundos de *benchmark* no programa ($dfmax$) [52] juntamente com a instância ($r500.5$).

4.2.4 Considerações Finais

Avaliando a heurística construtiva seguida da heurística VNDBS aplicada ao problema de coloração de vértices com pesos, podemos concluir pelos resultados obtidos nos experimentos computacionais que a heurística HOD-VNDBS encontra na média, soluções iguais ou melhores do que suas antecessoras. A heurística VNDBS mostrou-se bastante robusta alcançando 95,34% em média, o custo das soluções apresentadas por Malaguti et al. [18], independente da qualidade da solução inicial fornecida pela heurística construtiva. Outro ponto observado foi que a heurística VNDBS apresentou um bom desempenho em problemas onde as vizinhanças são grandes, conforme apresentado na Seção 3.1

			POPMUSIC		Heurística HOC-VNDBS								
H	L	Pos.	%Conflitos	Tempo(s)	1s	2s	3s	4s	5s	10s	15s	20s	120s
2	24	4	7,45	2,64	**5,66**	5,58	5,51	5,47	5,44	5,32	5,27	5,24	5,21
		8	2,78	4,88	**2,03**	1,99	1,98	1,97	1,95	1,92	1,90	1,89	1,86
3	16	4	11,27	3,68	**8,61**	8,46	8,39	8,35	8,29	8,14	8,07	8,04	7,98
		8	7,30	9,66	**5,87**	5,42	5,33	5,29	5,29	5,22	5,18	5,16	5,08
4	12	4	13,31	4,23	**10,43**	10,18	10,13	10,08	10,05	9,95	9,83	9,79	9,66
		8	6,94	9,92	**5,82**	5,03	4,88	4,86	4,84	4,79	4,75	4,72	4,67
6	8	4	15,58	4,91	**12,26**	11,60	11,51	11,48	11,45	11,33	11,26	11,22	11,10
		8	8,46	11,74	**7,45**	6,30	6,07	6,00	5,98	5,93	5,91	5,89	5,82
2	32	4	15,27	4,42	**11,75**	11,23	11,13	11,12	11,07	10,92	10,83	10,78	10,61
		8	7,49	10,16	**6,53**	5,52	5,31	5,26	5,20	5,11	5,08	5,06	4,96
4	16	4	24,51	6,90	**20,94**	18,67	18,10	17,91	17,80	17,61	17,49	17,42	17,21
		8	15,04	18,06	15,19	**12,71**	11,87	11,47	11,18	10,70	10,61	10,56	10,48
8	8	4	28,84	7,96	**25,44**	22,03	21,24	20,90	20,73	20,52	20,46	20,39	20,21
		8	18,65	21,75	19,27	**16,13**	15,08	14,59	14,15	13,20	12,97	12,90	12,81
16	4	4	* 25,06	* 4,49	**21,86**	19,38	18,76	18,50	18,37	18,20	18,12	18,05	17,84
		8	* 15,74	* 19,20	16,32	**13,42**	12,51	12,05	11,69	11,04	10,91	10,87	10,80
3	24	4	25,72	6,82	**22,54**	19,45	18,72	18,47	18,33	18,17	18,08	18,02	17,81
		8	18,93	21,07	19,32	**16,29**	15,21	14,70	14,29	13,45	13,21	13,17	13,08
4	18	4	30,04	8,12	**27,27**	23,30	22,29	21,90	21,63	21,35	21,27	21,21	20,97
		8	19,39	21,52	20,52	**17,26**	15,96	15,40	14,98	13,94	13,65	13,57	13,49
6	12	4	33,34	9,14	**31,48**	27,13	25,41	24,84	24,48	23,90	23,81	23,74	23,55
		8	22,54	25,29	24,35	**21,07**	18,95	18,25	17,67	16,35	15,85	15,68	15,46
8	9	4	34,90	9,57	**32,36**	28,16	26,55	26,01	25,64	24,84	24,70	24,65	24,48
		8	24,83	27,71	26,39	**23,16**	21,03	20,26	19,67	18,21	17,69	17,52	17,33
2	42	4	25,85	6,75	**22,29**	19,40	18,79	18,58	18,49	18,35	18,26	18,18	17,95
		8	15,52	17,12	**15,24**	12,63	11,78	11,36	11,04	10,58	10,47	10,43	10,35
3	28	4	32,88	8,72	**30,56**	26,21	24,81	24,34	23,98	23,40	23,33	23,28	23,11
		8	25,42	26,79	26,69	**23,49**	21,35	20,44	19,94	18,60	18,00	17,81	17,53
4	21	4	37,14	9,76	**35,47**	31,36	28,81	28,09	27,59	26,72	26,57	26,54	26,36
		8	26,71	28,51	28,36	**25,21**	22,96	21,79	21,16	19,65	18,97	18,71	18,42
6	14	4	41,13	11,18	**39,95**	36,18	33,33	31,97	31,31	29,97	29,59	29,51	29,32
		8	29,96	31,73	31,71	**28,72**	26,46	25,16	24,22	22,44	21,59	21,19	20,70
2	48	4	31,83	8,22	**29,44**	25,12	23,78	23,33	23,00	22,50	22,44	22,40	22,19
		8	20,48	21,25	21,14	**17,83**	16,37	15,80	15,33	14,21	13,85	13,73	13,61
3	32	4	39,56	10,30	**37,54**	33,62	30,86	29,88	29,29	28,19	27,97	27,92	27,76
		8	31,87	32,42	33,30	**30,33**	28,13	26,76	25,66	23,72	22,83	22,44	21,79
4	24	4	43,45	11,56	**42,08**	38,44	35,65	34,04	33,23	31,81	31,32	31,20	31,03
		8	32,14	33,88	33,71	**30,76**	28,60	27,22	26,08	24,05	23,18	22,76	22,06
6	16	4	48,02	12,82	**46,91**	43,53	40,93	39,23	37,67	35,47	34,79	34,62	34,41
		8	36,34	37,69	38,49	**35,70**	33,61	32,24	30,90	28,06	27,03	26,50	25,45

Tabela 4.2. Comparação com o algoritmo POPMUSIC proposto por Alvim e Taillard [3] e o algoritmo VNDBS .* Indica valores corrigidos por Alvim e Taillard [3] e disponibilizados na **URL:http://mistic.heig-vd.ch/taillard/articles.dir/articles.html**

	$HOC-VNDBS$				$HOD-VNDBS$				$HA-VNDBS$			
Instâncias	Média	**M**oda	σ	c_v	Média	**M**oda	σ	c_v	Média	**M**oda	σ	c_v
DSJC125_1g	24	25	0.88	0.04	24	24	0.42	0.02	23	23	0.52	0.02
DSJC125_1gb	94	93	1.35	0.01	94	94	0.74	0.01	93	92	1.84	0.02
DSJC125_5g	77	79	1.51	0.02	76	76	0.52	0.01	77	76	1.35	0.02
DSJC125_5gb	263	263	0.00	0.00	254	254	0.00	0.00	260	259	4.07	0.02
DSJC125_9g	171	170	0.88	0.01	169	169	0.00	0.00	171	171	1.14	0.01
DSJC125_9gb	614	614	0.00	0.00	612	613	0.82	0.00	610	615	4.32	0.01
GEOM30b	12	12	0.00	0.00	12	12	0.00	0.00	12	12	0.00	0.00
GEOM40b	16	16	0.00	0.00	16	16	0.00	0.00	16	16	0.00	0.00
GEOM50b	18	18	0.00	0.00	18	18	0.00	0.00	18	18	0.00	0.00
GEOM60b	23	23	0.00	0.00	23	23	0.00	0.00	23	23	0.00	0.00
GEOM70	47	47	0.00	0.00	47	47	0.00	0.00	47	47	0.00	0.00
GEOM70a	73	73	0.00	0.00	73	73	0.00	0.00	73	73	0.00	0.00
GEOM70b	24	24	0.00	0.00	24	24	0.00	0.00	24	24	0.00	0.00
GEOM80	66	66	0.00	0.00	66	66	0.00	0.00	66	66	0.00	0.00
GEOM80a	76	76	0.00	0.00	76	76	0.00	0.00	76	76	0.00	0.00
GEOM80b	27	27	0.00	0.00	27	27	0.00	0.00	27	27	0.00	0.00
GEOM90	61	61	0.00	0.00	61	61	0.00	0.00	61	61	0.00	0.00
GEOM90a	73	73	0.48	0.01	74	74	0.82	0.01	73	73	0.48	0.01
GEOM90b	30	30	0.00	0.00	30	30	0.00	0.00	30	30	0.00	0.00
GEOM100	65	65	0.00	0.00	65	65	0.00	0.00	65	65	0.00	0.00
GEOM100a	89	89	0.00	0.00	89	89	0.00	0.00	89	89	0.52	0.01
GEOM100b	32	32	0.00	0.00	32	32	0.00	0.00	32	32	0.00	0.00
GEOM110	68	68	0.00	0.00	68	68	0.00	0.00	68	68	0.00	0.00
GEOM110a	97	97	0.00	0.00	97	97	0.00	0.00	97	97	0.00	0.00
GEOM110b	37	37	0.00	0.00	37	37	0.00	0.00	37	37	0.00	0.00
GEOM120	72	72	0.00	0.00	72	72	0.00	0.00	72	72	0.00	0.00
GEOM120a	105	105	0.52	0.00	105	105	0.52	0.00	105	105	0.48	0.00
GEOM120b	35	35	0.52	0.01	35	36	0.48	0.01	35	35	0.52	0.01
R50_1g	14	14	0.00	0.00	14	14	0.00	0.00	14	14	0.00	0.00
R50_1gb	53	53	0.00	0.00	53	53	0.00	0.00	53	53	0.00	0.00
R50_5g	37	38	0.32	0.01	37	37	0.42	0.01	37	37	0.00	0.00
R50_5gb	135	135	1.48	0.01	135	135	0.48	0.00	138	138	1.66	0.01
R50_9g	74	74	0.00	0.00	74	74	0.00	0.00	74	74	0.00	0.00
R50_9gb	262	262	0.00	0.00	262	262	0.00	0.00	262	262	0.00	0.00
R75_1g	18	18	0.52	0.03	18	19	0.53	0.03	18	18	0.52	0.03
R75_1gb	72	72	0.00	0.00	72	72	1.23	0.02	71	71	0.00	0.00
R75_5g	52	52	1.06	0.02	52	52	0.82	0.02	52	53	0.57	0.01
R75_5gb	192	186	4.18	0.02	190	189	1.32	0.01	193	194	2.62	0.01
R75_9g	110	110	0.00	0.00	110	110	0.00	0.00	110	110	0.00	0.00
R75_9gb	396	396	0.00	0.00	396	396	0.00	0.00	397	396	2.91	0.01
R100_1g	22	22	0.42	0.02	22	22	0.42	0.02	22	22	0.42	0.02
R100_1gb	85	86	1.70	0.02	84	83	1.32	0.02	86	89	2.67	0.03
R100_5g	62	64	1.69	0.03	61	62	0.63	0.01	62	61	1.25	0.02
R100_5gb	232	233	2.08	0.01	232	232	0.00	0.00	232	235	6.02	0.03
R100_9g	141	141	0.00	0.00	142	142	0.52	0.00	142	142	1.14	0.01
R100_9gb	519	520	1.05	0.00	518	518	0.52	0.00	520	521	2.50	0.00
Média	105.76	105.78			105.39	105.43			105.72	105.91		

Tabela 4.3. Comparação da Média, **M**oda, desvio padrão σ e coeficiente de variância c_v entre as heurísticas HOC-VNDBS, HOD-VNDBS e HA-VNDBS.

Instâncias	Tempo(s)			Custo Médio			Malaguti		
	HOC VNDBS	HOD VNDBS	HA VNDBS	HOC VNDBS	HOD VNDBS	HA VNDBS	Custo	LB	Tempo(s)
DSJC125_1g	3	3	10	**24**	**24**	**23**	24	19	152
DSJC125_1gb	17	20	73	**94**	**94**	**93**	95	74	170
DSJC125_5g	38	13	7	77	**76**	77	76	-	180
DSJC125_5gb	20	28	79	263	263	263	**251**	-	182
DSJC125_9g	3	127	4	171	**169**	171	169	152	162
DSJC125_9gb	67	86	127	614	614	614	**605**	568	237
GEOM30b	0	0	0	**12**	**12**	**12**	12	12	0
GEOM40b	0	0	0	**16**	**16**	**16**	16	16	1
GEOM50b	0	0	0	**18**	**18**	**18**	18	18	0
GEOM60b	0	0	0	**23**	**23**	**23**	23	23	0
GEOM70	0	0	0	**47**	**47**	**47**	47	47	96
GEOM70a	0	0	0	**73**	**73**	**73**	73	73	3
GEOM70b	0	0	0	**24**	**24**	**24**	24	24	6
GEOM80	0	0	0	**66**	**66**	**66**	66	66	0
GEOM80a	0	0	0	**76**	**76**	**76**	76	76	102
GEOM80b	0	0	0	**27**	**27**	**27**	27	27	90
GEOM90	1	1	0	**61**	**61**	**61**	61	61	166
GEOM90a	17	1	1	**73**	73	**73**	73	73	157
GEOM90b	0	0	0	**30**	**30**	**30**	30	30	11
GEOM100	1	0	1	**65**	**65**	**65**	65	65	131
GEOM100a	1	1	1	**89**	**89**	**89**	89	89	112
GEOM100b	0	0	0	**32**	**32**	**32**	32	32	15
GEOM110	4	1	1	**68**	**68**	**68**	69	66	172
GEOM110a	10	9	5	**97**	**97**	**97**	97	97	111
GEOM110b	0	0	0	**37**	**37**	**37**	37	37	5
GEOM120	2	7	3	**72**	**72**	**72**	72	72	157
GEOM120a	30	42	10	**105**	**105**	**105**	105	105	136
GEOM120b	4	26	8	**35**	**35**	**35**	35	35	14
R50_1g	0	0	0	**14**	**14**	**14**	14	14	0
R50_1gb	0	0	0	**53**	**53**	**53**	53	52	95
R50_5g	0	0	0	**37**	**37**	**37**	37	35	167
R50_5gb	9	62	13	**135**	**135**	135	137	126	145
R50_9g	0	0	0	**74**	**74**	**74**	74	73	36
R50_9gb	7	0	1	**262**	**262**	**262**	262	257	33
R75_1g	0	0	1	**18**	**18**	**18**	19	17	154
R75_1gb	0	1	62	**72**	**72**	**71**	72	63	166
R75_5g	9	5	45	**52**	**52**	**52**	53	43	172
R75_5gb	109	127	67	192	**190**	192	190	160	173
R75_9g	0	0	0	**110**	**110**	**110**	110	108	79
R75_9gb	127	24	7	**396**	**396**	**397**	399	393	50
R100_1g	1	1	2	**22**	**22**	**22**	22	18	155
R100_1gb	17	30	20	85	**84**	85	84	70	171
R100_5g	10	79	19	**62**	**61**	**62**	62	48	179
R100_5gb	4	86	73	**232**	**232**	**232**	234	182	179
R100_9g	150	3	3	**141**	**142**	**142**	142	138	123
R100_9gb	4	53	42	**519**	**518**	**520**	520	499	127
Média	14.46	18.17	14.89	105.76	105.39	105.72	105,46	96,67	103,74

Tabela 4.4. Comparação das heurísticas HOC-VNDBS, HOD-VNDBS e HA-VNDBS com o algoritmo proposto por Malaguti et al. [18].

Capítulo 5

Conclusões e Trabalhos Futuros

Nesta dissertação foi realizado um estudo sobre uma heurística *Variable Neighbourhood Descent* que alterna entre vizinhanças de diferentes tamanhos, que são exploradas por um algoritmo de *Backtracking*, sendo chamada de VNDBS. A heurística VNDBS foi avaliada pela sua aplicação ao problema de rotulação cartográfica de pontos e ao problema de coloração de vértices com pesos.

Comparando com os resultados disponíveis na literatura, a heurística VNDBS alcançou resultados iguais ou melhores para todas as instâncias avaliadas para o problema de rotulação cartográfica de pontos. Para o problema de coloração de vértices com pesos, a heurística VNDBS alcançou resultados iguais ou melhores em $95,65\%$ das instâncias avaliadas, sendo superior quando é calculado a média dos resultados alcançados. Os resultados alcançados mostram que a heurística VNDBS se mostrou robusta em ambos problemas .

Como proposta de trabalhos futuros, podemos destacar

(i) Realizar um estudo aprofundado sobre as características da heurística VNDBS.

(ii) Avaliar a heurística VNDBS em classes de problemas tais como: Problemas de Rede, Problemas de Localização e Problemas de Programação.

(iii) Propor variantes adaptadas as características de diferentes problemas tais como: problema do caixeiro viajante [78], problema da mochila [4] e problema de atribuição [68].

Referências Bibliográficas

[1] Hertz A., Plumettaz M., and Zufferey N. Variable space search for graph coloring. *Discrete Applied Mathematics*, 156:2551–2560, 2008.

[2] Alvim A.C.F. and Taillard É.D. Instâncias para o problema de rotulação cartográfica de pontos com 13206 pontos cartográficos em. `http://mistic.heig-vd.ch/taillard/problemes.dir/problemes.html/` (Último acesso Março 2012).

[3] Alvim A.C.F. and Taillard É.D. POPMUSIC for the point feature label placement problem. *European Journal of Operational Research*, 192:396–413, 2009.

[4] Kleywegt A.J. and Papastavrou J.D. The dynamic and stochastic knapsack problem. *Operations Research*, 46:17–35, 1998.

[5] Corcoran A.L. and Wainwright R.L. Using libga to develop genetic algorithms for solving combinatorial optimization problems. In *of Lance Chambers, Editor, Practical Handbook of Genetic Algorithms, Applications*, pages 143–172. CRC Press, 1995.

[6] Martins A.X., Duhamel C., Mahey P., Saldanha R.R., and Souza M.C. Variable neighborhood descent with iterated local search for routing and wavelength assignment. *Computers & Operational Research*, 39(9):2133–2141, 2012.

[7] Escoffier B., Monnot J., and Paschos V.T. Weighted coloring: further complexity and approximability results. *Information Processing Letters*, 97:98–103, 2006.

[8] Avanthay C., Hertz A., and Zufferey N. A variable neighborhood search for graph coloring. *European Journal of Operational Research*, 151:379–388, 2003.

[9] Blum C. and Roli A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35:268–308, 2003.

[10] Blum C., Aguilera M.J.B, and Sampels A., Roli M. (Ed.). Hybrid Metaheuristics: Studies in computational intelligence. *Springer*, 35:1–30, 2008.

[11] Oliveira C., Urrutia S.A., and Noronha T.F. Heurística ILS para o problema da rotulação cartográfica de pontos. *XII SPOLM*, 84, 2009.

[12] Qi C., Marek C., and Gopalakrishnan S. Computing simple paths among obstacles. *Computational Geometry: Theory and Applications*, 16:223–233, 2000.

[13] Simona C. and Rémi M. Heuristic average-case analysis of the backtrack resolution of random 3-satisfiability instances. *Theoretical Computer Science*, 320:345–372, 2004.

[14] Ribeiro C.C. and Souza M.C. Variable neighborhood search for the degree-constrained minimum spanning tree problem. *Discrete Applied Mathematics*, 118:43–54, 2001.

[15] Ebner D., Klau G.W., and Weiskircher R. Label number maximization in the slider model. In *Proceedings of the 12th international conference on Graph Drawing*, GD'04, pages 144–154, Berlin, Heidelberg, 2004. Springer-Verlag.

[16] Knuth D.E. and Szwarcfiter J.L. A structured program to generate all topological sorting arrangements. *Information Processing Letters*, 2:153–157, 1974.

[17] Guan D.J. and Zhu X. A coloring problem for weighted graphs. *Information Processing Letters*, 61:77–81, 1997.

[18] Malaguti E., Monaci M., and Toth P. Models and heuristic algorithms for a weighted vertex coloring problem. *Journal of Heuristics*, 15:503–526, 2009.

[19] Talbi E. *Metaheuristics: From Design to Implementation*. Wiley Publishing, 2009.

[20] Taillard É.D. and Voß S. Popmusic: Partial optimization metaheuristic under special intensification conditions. *Kluwer Academic*, pages 613–629, 2000.

[21] Wagner F., Wolff A., Kapoor V., and Strijk T. Three rules suffice for good label placement. *Algorithmica*, 30, 2001.

[22] Butler G. and Lam C.W.H. A general backtrack algorithm for the isomorphism problem of combinatorial objects. *Journal of Symbolic Computation*, 1:363–381, 1985.

[23] Christelle G., Narendra J., and Christian P. Using intelligent backtracking to improve branch-and-bound methods: An application to open-shop problems. *European Journal of Operational Research*, 127:344–354, 2000.

[24] Cornuejols G., Nemhauser G.L., and Wolsey L.A. The uncapacitated facility location problem. In *Discrete Location Theory. Wiley Interscience*, 1990.

[25] Cravo G.L., Ribeiro G.M., and Lorena L.A.N. Novos algoritmos para o problema de rotulação cartográfica de pontos. Master's thesis, Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2003.

[26] Cravo G.L., Ribeiro G.M., and Lorena L.A.N. Heurística gulosa para o problema da rotulação cartográfica de pontos. *Revista Educação Tecnológica,*, (1/2), 2006.

[27] Cravo G.L., Ribeiro G.M., and Lorena L.A.N. Um Grasp eficiente para o problema da rotulação de pontos. *XXXVII SBPO*, 2006.

[28] Cravo G.L., Ribeiro G.M., and Lorena L.A.N. A greedy randomized adaptive search procedure for the point-feature cartographic label placement. *Computers & Geosciences*, 34(4):373–386, 2008.

[29] Félix G.L., Belén M.B., José A.M.P., and Marcos J.M.V. The parallel variable neighborhood search for the p-median problem. *Journal of Heuristics*, 8:375–388, May 2002.

[30] Ribeiro G.M. and Lorena L.A, N. Column generation approach for the point-feature cartographic label placement problem. *Journal of Combinatorial Optimization*, 15(2):147–164, 2008.

[31] Ribeiro G.M. and Lorena L.A.N. Modelagem matemática e relaxações lagrangeana e lagrangeana/surrogate para o problema de rotulação cartográfica de pontos. *XXXVI SBPO*, 2004.

[32] Ribeiro G.M. and Lorena L.A.N. Lagrangean relaxation with clusters for point-feature cartographic label placement problems. *Computers & Operational Research*, 35:2129–2140, 2008.

[33] Ribeiro G.M., Constantino M.F., and Lorena L.A.N. Um estudo sobre desigualdades válidas para o problema de maximização de rótulos livres. *XLI SBPO*, 2009.

[34] Mauri G.R. Novas abordagens para representação e obtenção de limitantes e soluções para alguns problemas de otimização combinatória. Master's thesis, Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2008.

[35] Mauri G.R., Ribeiro G.M., and Lorena L.A.N. New approaches for the point-feature cartographic label placement problems. *XLI SBPO*, 2009.

[36] Klau G.W. and Mutzel P. Optimal labeling of point features in rectangular labeling models. *Mathematical Programming*, 94(2-3):435–458, 2003.

[37] Priestley H.A. and Ward M.P. A multipurpose backtracking algorithm. *Journal of Symbolic Computation*, 18:1–40, 1994.

[38] Wilf H.S. Backtrack: An o(1) expected time algorithm for the graph coloring problem. *Information Processing Letters*, 18:119–121, 1984.

[39] Lynce I. and Silva J.M. Random backtracking in backtrack search algorithms for satisfiability. *Discrete Applied Mathematics*, 155:1604–1612, 2007.

[40] Ilog, Inc. Solver cplex, 2011. `http://www-01.ibm.com/software/websphere/` (Ültimo acesso outubro de 2011).

[41] Brinberg J., Hansen P., and Mladenović N. Convergence of variable neighborhood search. *Les Cahiers du GERAD and Mathematical Institute*, (130):449–467, 2004.

[42] Christensen J., Marks J., and Shieber S. Algorithms for cartographic label placement. In *Proc. American Congress on Surveying and Mapping 1*, pages 75–89, 1993.

[43] Christensen J., Marks J., and Shieber S. An empirical study of algorithms for point-feature label placement. *ACM Transactions on Graphics*, 14:203–232, 1995.

[44] Marks J. and Sheiber S. The computational complexity of cartographic label placement. Technical Report 05-91, Harvard CS, 1991.

[45] Davis L., editor. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991.

[46] Lorena L.A.N. Instâncias para o problema de rotulação cartográfica de pontos em. `http://www.lac.inpe.br/lorena/instancias.html/` (Último acesso Março 2012).

[47] Chiarandini M., Dumitrescu I., and Stützle T. Very large-scale neighborhood search: Overview and case studies on coloring problems. In Christian Blum, Maria J. Blesa, Andrea Roli, and Michael Sampels, editors, *Hybrid Metaheuristics*, volume 114 of *Studies in Computational Intelligence*, pages 117–150. Springer, 2008.

[48] Demange M., Werra de D., Monnot J., and Paschos V.T. Time slot scheduling of compatible jobs. *J. Scheduling*, 10:111–127, 2007.

[49] Yamamoto M., Camara G., and Lorena L.A.N. TABU search heuristic for point-feature cartographic label placement. *GeoInformatica*, 6:77–90, 2002.

[50] Yamamoto M., Camara G., and Lorena L.A.N. Fast point-feature label placement algorithm for real time screen maps, 2005.

[51] Yamamoto M. and Lorena L.A.N. A constructive genetic approach to point-feature cartographic label placement. In *Metaheuristics: Progress as Real Problem Solvers*, pages 285–300. Kluwer Academic Publishers, 2005.

[52] Trick M.A. *Computational symposium: Graph coloring and its generalizations.* Ithaca, NY, USA, 2002.

[53] Trick M.A. and Yildiz H. A large neighborhood search heuristic for graph coloring. In *Proceedings of the 4th international conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, CPAIOR '07, pages 346–360, Berlin, Heidelberg, 2007. Springer-Verlag.

[54] Wells M.B. *Elements of combinatorial computing.* Pergamon Press Oxford, New York,, [1st ed.] edition, 1971.

[55] Anderberg M.R. *Cluster analysis for applications.* Probability and mathematical statistics. Academic Press, 1973.

[56] Garey M.R. and Johnson D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences).* W. H. Freeman, first edition edition, 1979.

[57] Mladenović N. A tutorial on variable neighborhood search. *Les Cahiers du GERAD and Mathematical Institute, SANU*, 2003.

[58] Mladenović N. and Hansen P. Variable neighborhood search. *Computers & Operations Research*, 24:1097–1100, 1997.

[59] Verner O., Wainwright R., and Schoenefeld D. Placing text labels on maps and diagrams using genetic algorithms with masking. *INFORMS Journal on Computing*, 9:266–275, 1997.

[60] Özaltin O.Y., Prokopyev O.A., and Schaefer A.J. The bilevel knapsack problem with stochastic right-hand sides. *Operational Research Letters.*, 38:328–333, 2010.

[61] Festa P. and Resende M.G.C. Grasp: An annotated bibliography. In *Essays and Surveys in Metaheuristics*, pages 325–367. Kluwer Academic Publishers, 2002.

[62] Galinier P. and Hao J. Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization*, 3:379–397, 1999.

[63] Hansen P. and Mladenović N. An introduction to variable neighborhood search. In *Eds., Metaheuristics: Advances and Trends in Local Search Paradigms for Optimization*, pages 433–458. Kluwer Academic Publisher, 1999.

[64] Hansen P. and Mladenović N. *Variable Neighborhood Search. In Handbook of Applied Optimization.* Oxford University Press, January 2001.

[65] Hansen P. and Mladenović N. Variable neighbourhood search: Principles and applications. *European Journal of Operational Research*, pages 449–467, 2001.

[66] Hansen P., Mladenović N., and Perez J.A.M. Variable neighbourhood search: methods and applications. *European Journal of Operational Research*, 6:319–360, 2008.

[67] Hansen P., Mladenović N., and Dionisio P.B. Variable neighborhood decomposition search. *Journal of Heuristics*, 7:335–350, July 2001.

[68] Burkard R., Mell'Amico M., and Martello S. *Assignment Problems.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2009.

[69] Dechter R. and Frost D. Backjump-based backtracking for constraint satisfaction problems. *Artifcial Intelligence.*, 136:147–188, 2002.

[70] Mario R. and Günther R. R. Computer aided systems theory - eurocast 2009. chapter A Kruskal-Based Heuristic for the Rooted Delay-Constrained Minimum Spanning Tree Problem, pages 713–720. Springer-Verlag, Berlin, Heidelberg, 2009.

[71] Whitaker R.A. Some interchange algorithms for median location problems. *Environment and Planning B: Planning and Design*, 9(2):119–129, 1982.

[72] Walker R.J. An enumerative technique for a class of combinatorial problems. In *Proc. of the Tenth Symposium in Applied Mathematics of the American Mathematical Society*, pages 91–94, 1958.

[73] Kirkpatrick S., Gelatt C.D., and Vecchi M.P. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[74] Toriumi S., Endo H., and Imai K. Label size maximization for rectangular node labels. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, E89-A(4):1035–1041, April 2006.

[75] Voß. S., Osman I.H., and Roucairol C. *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*. Kluwer Academic Publishers, Norwell, MA, USA, 1999.

[76] Zoraster S. The solution of large 0-1 integer programming problems encountered in automated cartography. *Operations Research*, 38:752–759, 1990.

[77] Hirsch S.A. An algorithm for automatic name placement around point data. *The American Cartographer*, 9:5–17, 1982.

[78] David S.J. and Lyle A.M. *The Traveling Salesman Problem: A Case Study in Local Optimization*. 1997.

[79] Gerhart S.L. and Yelowitz L. Control structure abstractions of the backtracking programming technique. In *Proceedings of the 2nd international conference on Software engineering*, ICSE '76, pages 391–, Los Alamitos, CA, USA, 1976. IEEE Computer Society Press.

[80] Strijk T., Verweij A.M., and Aardal K.I. Algorithms for maximum independent set applied to map labelling. Technical Report UU-CS-2000-22, Department of Information and Computing Sciences, Utrecht University, 2000.

[81] Feo T.A. and Resende M.G.C. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

[82] Jensen T.R. and Toft B. *Graph coloring problems*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1995.

[83] Roever W.P. On backtracking and greatest fixpoints. In Arto Salomaa and Magnus Steinby, editors, *Automata, Languages and Programming, Fourth Colloquium, University of Turku, Finland, July 18-22, 1977, Proceedings*, volume 52 of *Lecture Notes in Computer Science*, pages 412–429. Springer, 1977.