HEURÍSTICA VND COM BACKTRACKING PARA O PROBLEMA DE COLORAÇÃO DE VÉRTICES COM PESOS

Celso Oliveira, Thiago F. Noronha, Sebastián Urrutia
Universidade Federal de Minas Gerais (UFMG)
Av. Antônio Carlos, 6627, ICEX Sala 4020, Belo Horizonte, 31270-901
{celso-olv,tfn,surrutia}@dcc.ufmg.br

Resumo

O problema de coloração de vértices com pesos, $WVCP$ é NP-Difícil consistindo em: Dado um grafo $G = (V, E)$, onde V é um conjunto de vértices, cada vértice $v \in V$ possui peso $w_v \geq 0$, deve-se atribuir uma cor a cada vértice tal que vértices adjacentes não recebam a mesma cor. Dado uma coloração, seja $C_k \subseteq V$ o subconjunto de vértices com cor k, para $k = 1, ..., n$, definimos o peso da cor k como o peso máximo de um vértice em C_k. O objetivo do $WVCP$ é minimizar a soma dos pesos das cores k. Este trabalho avalia a heurística VND, com busca local com *backtracking* aplicada ao $WVCP$ e propõe sua utilização como técnica na resolução de problemas de otimização combinatória. Os resultados alcançados apontam sua aplicabilidade devido o baixo custo computacional e resultados comparáveis aos melhores trabalhos da literatura.

Palavras-chave: Coloração de grafos, Heurísticas, VND, Backtracking.

Abstract

The weighted vertex coloring problem, $WVCP$ is NP-Hard in which given a graph $G = (V, E)$, V is a set of vertex, 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, $C_k \subseteq V$ 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. This paper evaluates the heuristic VND with local search with *backtracking* applied to WVCP and proposes its as a technique to solve combinatorial optimization problems. The low computational cost and results comparable to best works in literature show its applicability.

Key Words : Graph coloring, Heuristic, VND, Backtracking.

1 Introdução

O problema de coloração de vértices com pesos, do inglês *Weighted Vertex Coloring Problem*, $WVCP$ é NP-Difícil Garey e Johnson (1979) e consiste em: Dado um grafo $G = (V, E)$, onde V é um conjunto de vértices, cada vértice $v \in V$ possui peso $w_v \geq 0$, deve-se atribuir uma cor a cada vértice tal que vértices adjacentes não recebam a mesma cor. Dado uma coloração, seja $C_k \subseteq V$ o subconjunto de vértices com cor k para $k = 1, ..., n$. Definimos o peso da cor k como o peso máximo de um vértice em C_k. O objetivo do problema é minimizar a soma dos pesos das cores k.

O $WVCP$ é uma generalização do VCP, do inglês *Vertex Coloring Problem*, onde todas as cores possuem o mesmo custo e diferente do $WVCP$ seu objetivo é minimizar o número de cores utilizadas. O $WVCP$ é um problema de coloração de grafos GCP, do inglês *Graph Coloring Problem*, uma k-coloração de G é o número de cores k utilizadas para colorir todos os vértices $v \in V$ e o número cromático definido por $\chi(G)$ é o menor número de cores possíveis de uma k-coloração, sendo $k \leq n$ e $n = |V|$.

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 os recursos, os vértices as demandas e as arestas entre duas demandas representam a impossibilidade do atendimento simultâneo. Numa solução para o $WVCP$ as cores fornecem o número de recursos e o custo necessário para atendimento das demandas. O $WVCP$ se aplica também ao problema de programação de horários, do inglês *TimeTable* Cangalovic e Schreuder (1991), sendo uma generalização do problema de matriz de decomposição, do inglês *Matrix Decomposition Problem in Time Division Multiple Access Traffic Assignment* Prais et al. (1998).

Este trabalho avalia a heurística VND, do inglês *Variable Neighbourhood Descent*, com busca local com *backtracking* aplicada ao $WVCP$ e propõe sua utilização como técnica na resolução de problemas de otimização combinatória. Os resultados alcançados pelos experimentos computacionais no conjunto de instâncias avaliados, apontam a viabilidade de sua utilização, devido o baixo custo computacional e obter resultados comparáveis aos melhores trabalhos da literatura.

O restante deste trabalho se divide em: Na Seção 2 apresentamos um breve estudo sobre a vizinhança do $WVCP$, na Seção 3 introduzimos a estratégia da solução, heurística construtiva e a heurística VND com busca local com *backtracking*, na Seção 4 descrevemos os experimentos computacionais com resultados sobre instâncias da literatura e por último na Seção 5 apresentamos as conclusões finais.

2 Trabalhos Relacionados

Analisando as abordagens para solução do $WVCP$ Malaguti et al. (2009) 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 refinada, apresentando resultados para um grande número de instâncias da literatura.

Demange et al. (2007) investigaram a complexidade e aproximabilidade dos problemas de coloração com pesos WCP, do inglês *Weighted Coloring Problems*, discutindo algumas propriedades das soluções ótimas, complexidade e apresentando resultados de estratégias de aproximações. Escoffier et al. (2006) prosseguiram com a investigação baseando-se no trabalho de Guan e Zhu (1997), provando que o WCP é fortemente NP-Difícil para grafos de

intervalos e discutiram a aproximação polinomial para o WCP, demonstrando algoritmos de aproximação para grafos coloriveis com poucas cores e para subgrafos $k-tree$.

As características das vizinhanças do GCP foram avaliadas em Chiarandini et al. (2008) 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 t-coloração.

Hertz et al. (2008) apresentaram um novo algoritmo de busca local chamado de Pesquisa Espacial Variável, do inglês *Variable Space Search*, VSS aplicado ao problema de k-coloração. O VSS estende a metodologia da Formulação de Pesquisa Espacial, do inglês *Formulation Space Search*, FSS, 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 diferentes formulações do problema, sendo algumas restrições relaxadas para atender outras formulações num mesmo espaço de busca.

Trick e Yildiz (2007) 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 a resolução de um problema $MAX-CUT$.

Avanthay et al. (2003) propôs um VNS Voss et al. (1999); Mladenović e Hansen (1997) apresentando diversas estruturas de vizinhanças, sendo algumas de tamanho exponencial. Os autores utilizaram uma busca tabu na busca local, no entanto não fonecem uma forma de explorar a vizinhança com eficiencia, não apresentando resultados competitivos quando comparado com os resultados alcançados por Galinier e Hao (1999).

3 Estratégia da Solução

Nesta seção apresentaremos a nossa estratégia para resolução do problema $WVCP$. Tal estratégia consiste em três etapas: Na primeira aplicaremos um algoritmo de pré-processamento para reduzir o tamanho do grafo $G=(V,E)$ e adequar o conjunto de vértices V para a heurística construtiva. Na segunda uma solução inicial é construída pela heurística construtiva e por final, na terceira etapa a heurística VND com busca local com *Backtracking* processa a solução inicial reduzindo o custo final da solução.

Variable Neighbourhood Descent VND é uma heurística proposta por Hansen et al. (2008); Hansen e Mladenovic (2001) que utiliza diversas estruturas de vizinhanças $N_k(x)$, $k=1,...,K$, sendo uma estrutura de vizinhança $N(x)$ definida para todo $x \in X$ onde X é o conjunto de soluções viáveis, x é uma solução e $N(x) \subseteq X$ um conjunto de soluções obtido através de alguma modificação ou movimento nos elementos de x por uma função N. A partir de uma solução inicial x e uma vizinhança $N(x)$, o VND realiza uma busca local explorando toda a vizinhança de x na direção de descida mais íngreme no espaço de busca, movendo-se para cada solução que possua um custo melhor que a solução corrente. Ao encontrar um ótimo local para uma vizinhança $N_y(x)$ o VND seleciona outra vizinhança $N_z(x)$ e sucessivamente realiza a busca local até encontrar um ótimo local em relação a

todas vizinhanças $N_k(x)$.

O VND se baseia no fato que um ótimo local para uma vizinhança $N_y(x)$ não é necessariamente um ótimo local para outra vizinhança $N_z(x)$ Hansen et al. (2008), desta maneira o VND explora o espaço de busca alternando sistematicamente a forma pela qual uma solução x é modificada pelas funções N_k.

O algoritmo de *Backtracking* é baseado no DFS, do inglês *Deph-first Search* Cormen et al. (2001), onde na exploração dos vértices de um grafo, múltiplas soluções podem ser eliminadas sem serem explicitamente examinadas. O algoritmo de *Backtracking* foi extensivamente estudado Priestley e Ward (1994) e utilizado como estratégia para resolver problemas combinatórios, podendo no pior caso ser extremamente ineficiente, sendo necessária uma análise sistemática para melhorar sua eficiência. Devido esta característica o algoritmo de *Backtracking* foi considerado como um método de último recurso, sendo todavia amplamente utilizados, especialmente em problemas NP-completos Priestley e Ward (1994).

3.1 Pré-processamento

Dado um grafo não orientado $G = (V, E)$, um subconjunto $C \subseteq V$ é independente se não possui vértices adjacentes. Uma solução S para o problema $WVCP$ é uma partição $\{C_1, C_2, ..., C_k\}$ de V, onde cada C_i contem os vértices coloridos com a cor i. Para cada vértice $v \in V$ denotamos A_v o conjunto de vértices adjacentes a v.

O objetivo do pré-processamento é a redução do tamanho do grafo $G = (V, E)$ e adequar o conjunto de vértices V para a heurística construtiva. Nesta etapa o conjunto de vértices V é percorrido e caso para um vértice $v \in V$, $A_v = \emptyset$, o vértice v e removido do conjunto V, sendo colorido ao final do processamento com a cor de maior custo. Após esta etapa o conjunto de vértices V é armazenado numa lista de vértices π, sendo ordenada de acordo com a heurística construtiva.

3.2 Heurística construtiva

A heurística construtiva proposta, detalhada na Figura 1, recebe o grafo $G = (V, E)$ e a lista de vértices π, seleciona sequencialmente os vértices π_j da lista de vértices π e insere no conjunto independente $C_k \subseteq V$ que resulta no menor custo para solução S, retornando ao final a solução S contendo a melhor solução encontrada. A heurística construtiva foi avaliada com a lista de vértices ordenada 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 HC, heurística construtiva decrescente HD e heurística construtiva aleatória HA.

Na linha 1, o conjunto solução S é inicializado. O laço das linhas 2-4 seleciona os vértices $v \in \pi$ e insere em um conjunto independente C_k. Na linha 3, o vértice π_j é inserido em um conjunto independente C_k que resulta no menor custo para a solução S. Na linha 4, o conjunto independente C_i é atualizado na solução S e por fim na linha 5, o conjunto solução S é retornado com a melhor solução encontrada.

```
Procedure HC(G = (V, E), π)
1  S ← ∅
2  for j = 1 to |V| do
3     C_k ← SelecionaConjuntoIndependenteComMenorCusto(G = (V, E), S, π_j)
4     S ← S ∪ C_k
5  return S
```

Figura 1: Psedo-código da heurística construtiva.

3.3 Heurística VND

Variable Neighbourhood Descent VND, Hansen et al. (2008); Hansen e Mladenovic (2001) é uma metaheurística que explora sistematicamente a idéia de mudar de vizinhança, onde a partir de uma solução inicial o VND explora as vizinhanças cada vez mais distantes da solução corrente, movendo-se somente quando encontra uma solução melhor. Para o $WVCP$ uma vizinhança é uma função que mapeia uma solução S em um conjunto de soluções $N(S,Y)$ substituindo na solução S as cores de um conjunto de vértices Y. As estruturas de vizinhanças $N_k(S,Y), k = 1, ..., k_{max}$, são obtidas realizando a substituição de cores nos vértices adjacentes a $v \in Y$, em k níveis pelo algoritmo de *Backtracking*.

A complexidade da vizinhança $N_k(S,Y)$ é definida pelo nível de profundidade do algoritmo de *Backtracking*, sendo determinado pela variável β e limitado pelo parâmetro β_{max}. O tamanho ou amplitude da vizinhança é definida pela cardinalidade do conjunto Y, sendo determinada pela variável α e limitada pelo parâmetro α_{max}. A idéia principal do VND proposto é pesquisar a vizinhança $N_k(S,Y)$ em função da profundidade e da amplitude, $N_\beta(S, Y = f(G = (V,E), \alpha, \gamma))$ onde α e β determinam respectivamente a amplitude e profundidade do VND e γ determina a forma como uma função f seleciona os vértices $v \in V$ para o conjunto Y.

O algoritmo de *Backtracking* pesquisa a vizinhança $N_k(S,Y)$ avaliando o vértice v em cada conjunto indepedente $C_i \subseteq S$, aceitando as soluções de custo igual ao custo da solução corrente S ou retornando a primeira solução de custo melhor. O algoritmo de *backtracking* é chamado recursivamente para todos os vértices adjacentes a v que também estejam no conjunto independente C_i. O parâmetro β limita a profundidade do algoritmo de *backtracking* sendo decrementando até atingir seu valor mínimo 0. Ao final do processamento o algoritmo de *backtracking* retornará a melhor solução encontrada.

Conforme detalhado na Figura 2, o algoritmo de *Backtracking* recebe o grafo $G = (V,E)$, a solução S, o vértice v e o nível do *Bactracking* $\beta \geq 0$. Na linha 1, o conjunto solução S'' é inicializado. Na linha 2, S' recebe a solução S. O laço das linhas 3-19 seleciona aleatoriamente os conjuntos independentes $C_i \subseteq S'$ e avalia a inserção do vértice v na solução S' , selecionando a solução S' que resulta no menor aumento do custo ou retorna a primeira solução melhor que S. Na linha 4, o vértice v é inserido no conjunto independente C_i, atualizando a solução S' e retorna o conjunto J_v com os vértices adjacentes a v que pertencem a C_i. Caso o custo da solução S' seja menor que o custo de S a nova solução S' é retornada na linha 7.

Na linha 8, a solução S'' é atualizada, armazenando a melhor solução encontrada. Caso o custo da solução S' aumente, na linha 10 é avaliado se o custo da solução S' é o melhor custo encontrado. Na linha 11, a solução S'' é atualizada, armazenando a melhor solução encontrada. Na linha 12, é avaliado se o conjunto J_v é diferente de vazio e se o nível de profundidade do algoritmo de *backtracking* β é maior que 0, caso verdadeiro o laço das linhas 13-18 chama o algoritmo de *Backtracking* selecionando aleatoriamente todos os vértices $u \in J_v$, retornando a solução S''' caso seja melhor que S ou atribuindo a S'' as soluções S''' que não aumentam custo da solução S'.

Na linha 14, S''' recebe a solução retornada pelo algoritmo de *Backtracking* para o vértice u, a solução S' e o nível do algoritmo de *Backtracking* $\beta - 1$. Nas linhas 15-16, a solução S''' é avaliada e caso seja melhor que S, S''' é retornada na linha 17. Caso o custo de S''' seja melhor que S'', na linha 18, a solução S'' é atualizada, armazenando a melhor solução encontrada. Na linha 19, a solução S' recebe a melhor solução encontrada S'' com o vértice $v \in C_i$ removido do conjunto C_i que v pertence e por fim na linha 20, a solução S'' é retornada tendo a ultima melhor solução encontrada.

Conforme detalhado na Figura 3, a busca local é realizada pelo procedimento *Back-*

```
Procedure Backtracking(G = (V, E), S, v, β)
1   S'' ← ∅
2   S' ← S
3   for all C ⊂ S' or S' = ∅ do
4      J_v ← Insere v no conjunto independente C_i, atualiza S' e retorna os vértices A_v ∩ C_i
5      if Custo(S') ≤ Custo(S) and J_v = ∅ then
6         if Custo(S') < Custo(S) then
7            return S'
8         S'' ← S'
9      else
10        if Custo(S') ≤ Custo(S'') or S'' = ∅ then
11           S'' ← S'
12        if J_v ≠ ∅ and β > 0 then
13           for all u ∈ J_v do
14              S''' ← Backtracking(G = (V, E), S' \ {u}, u, β − 1)
15              if Custo(S''') ≤ Custo(S'') then
16                 if Custo(S''') < Custo(S) then
17                    return S'''
18                 S'' ← S'''
19     S' ← S'' \ {v}
20  return S''
```

Figura 2: Pseudo-código do procedimento Backtracking.

tracking para todos os vértices do conjunto Y, retornando a solução encontrada S'' caso o custo seja melhor ou igual a solução corrente S. O procedimento *BuscaLocal*, recebe o grafo $G = (V, E)$, a solução S, o conjunto de vértices Y e o nível do *Bactracking* β. Na linha 1, S' recebe a solução S com todos os vértices de Y removidos de seus respectivos conjuntos C. O laço das linhas 2-3 chama o procedimento *Backtracking* para todos os vértices de Y, armazenando a solução em S'. Na linha 4, caso a solução S' tenha um custo melhor ou igual a S, S' é retornada na linha 5 e por fim na linha 6, caso o custo da solução S' não seja melhor que S, a solução corrente S é retornada

```
Procedure BuscaLocal(G = (V, E), S, Y, β)
1  S' ← S \ Y
2  for i = 1 to |Y| do
3     S' ← Backtracking(G = (V, E), S', Y_i, β)
4  if Custo(S') ≤ Custo(S) then
5     return S'
6  return S
```

Figura 3: Pseudo-código do procedimento *BuscaLocal*.

A estrutura do VND é detalhada na Figura 4 onde o algoritmo recebe o grafo $G = (V, E)$, a solução inicial S, o limite de amplitude α_{max} e o limite de profundidade do algoritmo de *backtracking* β_{max}. A construção dos conjuntos de vértices Y é realizado pela função f que retorna um conjunto de vértices em função de α e γ. O parâmetro α determina a cardinalidade do conjunto Y e γ o algoritmo a qual os vértices $v \in S$ serão selecionados. A variável k é um contador de processamento determinando a latência que α, β e γ serão incrementadas, sendo limitada por k_{max}. Na linha 1, α, β, γ e k são iniciados em 0. O laço das linhas 2-24, constrói os conjuntos Y a partir da solução S, efetua a busca local e armazena em S as soluções S' que tenham o custo menor ou igual a S, até que o

critério de parada seja satisfeito.

Na linha 3, Y recebe o conjunto com $T(\alpha)$ vértices que são obtidos da solução S pelo algoritmo A(γ). Na linha 4, é realizado a busca local na vizinhança $N_k(S, Y)$ com a profundidade β, retornando a solução S'. Na linha 5, caso o custo da solução S' diminua, as variáveis α, β, γ e k são iniciadas em 0. Na linha 8, caso k menor que k_{max}, na linha 9 k é incrementada, do contrário, na linha 11 k é inicializado com 0. Na linha 12, caso a amplitude do VND α seja menor que α_{max}, α é incrementado na linha 13, do contrário na linha 15, α é inicializada com 0.

Na linha 16, caso γ seja menor que γ_{max}, γ é incrementado na linha 17 alternando o algoritmo de construção dos conjuntos Y, do contrário na linha 19, γ é inicializada com 0. Na linha 20, caso a profundidade do algoritmo de *backtracking* β seja menor que β_{max}, β é incrementado na linha 21, do contrário na linha 23, β é inicializada com 0 reiniciando o VND com a menor amplitude e profundidade. Na linha 24, a solução corrente S recebe a solução S' que possui um custo menor ou igual a S e por fim na linha 25, a solução S é retornada.

```
Procedure VND(G = (V, E), S, α_max, β_max)
1   k, α, β, γ ← 0
2   until critério de parada não satisfeito do
3     Y ← f((G = V, E), T(α), A(γ))
4     S' ← BuscaLocal((G = V, E), S, Y, β)
5     if Custo(S') < Custo(S) then
6       k, α, β, γ ← 0
7     else
8       if k < k_max then
9         k ← k + 1
10      else
11        k ← 0
12        if α < α_max then
13          α ← α + 1
14        else
15          α ← 0
16          if γ < γ_max then
17            γ ← γ + 1
18          else
19            γ ← 0
20            if β < β_max then
21              β ← β + 1
22            else
23              β ← 0
24    S ← S'
25  return S
```

Figura 4: Pseudo-código do procedimento *VND*.

4 Resultados experimentais

Os experimentos computacionais foram realizados em um conjunto de instâncias proposto por Trick (2002). As heurísticas HA-VND, HC-VND e HD-VND foram estudadas, onde HA-VND, HC-VND e HD-VND denotam as heurísticas HA, HC, e HD seguidas da aplicação da heurística VND descrita na Seção 3.3. 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. Para permitir uma comparação aproximada dos tempos de processamento em problemas de coloração em máquinas diferentes, foi utilizado o programa *benchmark* ($dfmax$) Trick (2002) juntamente com a instância ($r500.5$), obtendo um desempenho 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 300 segundos. Para cada instância foram armazenados o número de cores utilizadas, 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} foi definido em 10 unidades onde cada unidade representa 3.333% do número de vértices da instância, conforme função $T(\alpha) = \alpha * 0,03333 * |V|$. O parâmetro γ_{max} foi definido em 2 onde para $\gamma = 0$ o algoritmo A copia de V $T(\alpha)$ vértices de forma aleatória e para $\gamma = 1$, seleciona aleatoriamente os subconjuntos C_k e copia um número aleatório de vértices em ordem decrescente dos pesos até totalizar $T(\alpha)$ vértices. O parâmetro k_{max} fornece a latência do aumento da amplitude e profundidade do VND, sendo definido pela função $k_{max} = \frac{|V|}{\log((\beta * |S|)+1)+1}$, onde k_{max} assume o maior valor quando $\beta = 0$ e o menor valor quando $\beta = \beta_{max}$ para um número de cores $|S|$.

A Tabela 1 apresenta a comparação entre a heurística proposta neste trabalho e a heurística proposta por Malaguti et al. (2009). O nome das instâncias é definido na coluna *instâncias*. As colunas *Cores*, *Tempo* e *Custo* apresentam o número de cores, o tempo computacional em segundos e custo médio da solução encontrada para cada heurística, HC, HD e HA seguida da aplicação da heurística VND descrita na seção 3.3. A coluna *Malaguti* fornece os resultados alcançados pela heurística proposta por Malaguti et al. (2009) sendo o custo fornecido pela coluna *Custo*, o *Lower Bound* na coluna LB e o tempo computacional total em segundos na coluna *Tempo*. Os valores em negrito denotam os resultados alcançados pelas heurísticas $HC-VND$, $HD-VND$ e $HA-VND$, com custo igual ou inferior as resultados alcançados por Malaguti et al. (2009).

Pelo desvio padrão pode-se observar a baixa dispersão do número de cores e do custo final das soluções para as heurísticas $HC-VND$, $HD-VND$ e $HA-VND$. Observamos também que as heurísticas $HC-VND$, $HD-VND$ e $HA-VND$ demandam de tempos distintos para o processamento das instâncias, devido a maior dispersão do desvio padrão. O custo computacional apresentados pelas heurísticas $HC-VND$, $HD-VND$ e $HA-VND$ são inferiores a $\frac{1}{3}$ da média demandada pela heurística proposta por Malaguti et al. (2009). Comparando os resultados alcaçados por Malaguti et al. (2009), as médias dos custos das soluções geradas pelas heurísticas $HC-VND$ e $HA-VND$, são maiores em $0,37\%$ e $0,94\%$ respectivamente e menor para a média da heurística $HD-VND$ em $0,05\%$. No geral comparando os resultados alcançados por Malaguti et al. (2009) a heurística proposta fornece soluções de custo inferior ou igual para 44 instâncias e a heurística $HD-VND$ para 38 instâncias.

Instâncias	Cores			Tempo(s)			Custo			Malaguti		
	HC VND	HD VND	HA VND	HC VND	HD VND	HA VND	HC VND	HD VND	HA VND	Custo	LB	Tempo(s)
DSJC125_1g	6	6	6	115	47	39	**24**	**24**	**23**	24	19	152
DSJC125_1gb	6	6	6	61	53	62	**93**	**92**	**94**	95	74	170
DSJC125_5g	20	21	20	31	68	121	79	**76**	77	76	-	180
DSJC125_5gb	21	21	21	127	96	87	259	252	266	251	-	182
DSJC125_9g	49	48	49	37	66	31	172	170	171	169	152	162
DSJC125_9gb	49	48	49	36	58	80	608	606	610	605	568	237
GEOM30b	5	5	5	0	0	0	**12**	**12**	**12**	12	12	0
GEOM40b	7	7	7	0	0	0	**16**	**16**	**16**	16	16	1
GEOM50b	8	8	8	0	0	0	**18**	**18**	**18**	18	18	0
GEOM60b	9	9	9	0	0	0	**23**	**23**	**23**	23	23	0
GEOM70	8	8	8	1	1	1	**47**	**47**	**47**	47	47	96
GEOM70a	11	11	11	1	2	1	**73**	**73**	**73**	73	73	3
GEOM70b	10	10	10	1	1	1	**24**	**24**	**24**	24	24	6
GEOM80	8	8	8	0	0	0	**66**	**66**	**66**	66	66	0
GEOM80a	12	12	12	1	1	1	**76**	**76**	**76**	76	76	102
GEOM80b	12	12	12	13	14	5	**27**	**27**	**27**	27	27	90
GEOM90	9	9	9	6	38	7	**61**	**61**	**61**	61	61	166
GEOM90a	12	13	12	77	49	36	**73**	74	**73**	73	73	157
GEOM90b	15	15	15	2	2	3	**30**	**30**	**30**	30	30	11
GEOM100	9	9	9	3	15	6	**65**	**65**	**65**	65	65	131
GEOM100a	14	14	14	13	9	6	**89**	**89**	**89**	89	89	112
GEOM100b	15	15	15	4	4	3	**32**	**32**	**32**	32	32	15
GEOM110	9	9	9	4	10	8	**68**	**68**	**68**	69	66	172
GEOM110a	14	14	14	62	96	47	**97**	**97**	**97**	97	97	111
GEOM110b	15	15	15	11	28	3	**37**	**37**	**37**	37	37	5
GEOM120	11	11	11	10	11	31	**72**	**72**	**72**	72	72	157
GEOM120a	16	16	16	28	23	48	**105**	**105**	**105**	105	105	136
GEOM120b	16	16	16	137	47	90	**35**	36	**35**	35	35	14
R50_1g	4	4	4	0	0	0	**14**	**14**	**14**	14	14	0
R50_1gb	4	4	4	0	0	1	**53**	**53**	**53**	53	52	95
R50_5g	11	11	11	16	11	1	38	**37**	**37**	37	35	167
R50_5gb	11	11	11	2	18	18	139	**137**	138	137	126	145
R50_9g	22	22	22	1	1	0	**74**	**74**	**74**	74	73	36
R50_9gb	23	23	23	2	5	4	**262**	**262**	**262**	262	257	33
R75_1g	4	4	4	17	10	22	**18**	**19**	**18**	19	17	154
R75_1gb	4	5	4	50	29	10	**72**	73	**71**	72	63	166
R75_5g	14	15	14	41	57	3	**53**	**53**	**53**	53	43	172
R75_5gb	15	15	15	75	54	48	195	191	195	190	160	173
R75_9g	33	35	34	7	1	3	**110**	**110**	**110**	110	108	79
R75_9gb	34	34	34	1	21	51	**396**	**397**	**396**	399	393	50
R100_1g	6	5	5	23	22	9	**22**	**22**	23	22	18	155
R100_1gb	5	6	6	42	79	62	87	**84**	87	84	70	171
R100_5g	16	17	17	60	105	37	63	**62**	**62**	62	48	179
R100_5gb	17	18	17	56	145	66	**233**	**232**	**233**	234	182	179
R100_9g	39	40	39	54	75	13	**141**	143	**142**	142	138	123
R100_9gb	39	40	40	101	69	63	**519**	**518**	523	520	499	127
Média	15,18	15,32	15,48	28,93	31,31	24,57	105,83	105,41	106,04	105,46	96,67	103,74
Desvio Padrão		0,15			3,42			0,32				

Tabela 1: Comparação das heurísticas HC-VND, HD-VND e HA-VND com o algoritmo proposto por Malaguti et al. (2009).

No trabalho de Malaguti et al. (2009) 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 ($dfmax$) Trick (2002) juntamente com a instância ($r500.5$).

5 Considerações finais

Este trabalho propôs e avaliou uma heurística construtiva seguida de um VND com busca local com *backtracking* para o problema de coloração de vértices com pesos. Os resultados obtidos pelos experimentos computacionais mostraram que a heurística $HD - VND$ encontra na média, soluções melhores do que suas antecessoras. A heurística VND mostrou-se bastante robusta alcançando acima de 99,4% em média, o custo das soluções fornecidos por Malaguti et al. (2009), independente da qualidade da solução inicial fornecida pela heurística construtiva. Outro ponto observado foi que a heurística VND com busca local com *backtracking* apresentou um bom desempenho em problemas onde as vizinhanças são grandes, conforme apresentado na Seção 2, apontando a viabilidade de sua utilização como técnica na resolução de problemas de otimização combinatória.

References

Avanthay, C.; Hertz, A. e Zufferey, N. (2003). A variable neighborhood search for graph coloring. *European Journal of Operational Research*, v. 151, p. 379–388.

Cangalovic, M. e Schreuder, J.A.M. March(1991). Exact colouring algorithm for weighted graphs applied to timetabling problems with lectures of different lengths. *European Journal of Operational Research*, v. 51, n. 2, p. 248–258.

Chiarandini, Marco; Dumitrescu, Irina e Stützle, Thomas. (2008). Very large-scale neighborhood search: Overview and case studies on coloring problems. Blum, Christian; Blesa, Maria J.; Roli, Andrea e Sampels, Michael, editors, *Hybrid Metaheuristics*, volume 114 of *Studies in Computational Intelligence*, p. 117–150. Springer.

Cormen, T.H.; Leiserson, R.L.C.E.and Rivest e Stein, C. (2001). *Introduction to Algorithms*. MIT Press, Cambridge, MA.

Demange, M.; deWerra, D.; Monnot, J. e Paschos, V.Th. (2007). Time slot scheduling of compatible jobs. *J. Scheduling*, v. 10, n. 2, p. 111–127.

Escoffier, B.; Monnot, J. e Paschos, V.Th. February(2006). Weighted coloring: further complexity and approximability results. *Inf. Process. Lett.*, v. 97, p. 98–103.

Galinier, P. e Hao, J. (1999). Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization*, v. 3, n. 4, p. 379–397.

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

Guan, D.J. e Zhu, X. January(1997). A coloring problem for weighted graphs. *Inf. Process. Lett.*, v. 61, p. 77–81.

Hansen, P. e Mladenovic, N. (2001). Variable neighbourhood search: principles and applications. *EJOR*, v. , n. 130, p. 449–467.

Hansen, P.; Mladenovic, N. e Pérez, J.A.M. (2008). Variable neighbourhood search: methods and applications. *EJOR*, v. 6, n. 4, p. 319–360.

Hertz, A.; Plumettaz, M. e Zufferey, N. (2008). Variable space search for graph coloring. *Discrete Applied Mathematics*, v. 156, n. 13, p. 2551 – 2560.

Malaguti, E.; Monaci, M. e Toth, P. October(2009). Models and heuristic algorithms for a weighted vertex coloring problem. *Journal of Heuristics*, v. 15, p. 503–526.

Mladenović, N. e Hansen, P. November(1997). Variable neighborhood search. *Comput. Oper. Res.*, v. 24, p. 1097–1100.

Prais, Marcelo; Ribeiro, Celso C.; Celso, e Ribeiro, C. (1998). Reactive grasp: An application to a matrix decomposition problem in tdma traffic assignment. *INFORMS Journal on Computing*, v. 12, p. 164–176.

Priestley, H.A. e Ward, M.P. July(1994). A multipurpose backtracking algorithm. *J. Symb. Comput.*, v. 18, p. 1–40.

Trick, M.A. (2002). *Computational symposium: Graph coloring and its generalizations*. Ithaca, NY, USA. URL `http://mat.gsia.cmu.edu/COLOR02/ Último acesso Abril 2011`.

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

Voss, S.; Osman, I.H. e Roucairol, C. (1999). *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*. Kluwer Academic Publishers, Norwell, MA, USA.