VARIABLE NEIGHBORHOOD SEARCH COM PATH RELINKING APLICADO AO PROBLEMA DE COLORAÇÃO DE VÉRTICES

Gustavo Cunha de Bittencourt
Universidade Federal do Rio de Janeiro (COPPE/UFRJ)
Av. Athos da Silveira Ramos, 149,bloco F- Sl. 105- Cidade Universitária, Ilha do Fundão – RJ
gustavo.bittencourt@engenharia.ufjf.br

Miranda Albino Martins Muaualo
Universidade Federal do Rio de Janeiro (COPPE/UFRJ)
Av. Athos da Silveira Ramos, 149,bloco F- Sl. 105- Cidade Universitária, Ilha do Fundão – RJ
mirandam939@gmail.com

Laura Silvia Bahiense da Silva Leite
Universidade Federal do Rio de Janeiro (COPPE/UFRJ)
Av. Athos da Silveira Ramos, 149,bloco F- Sl. 105- Cidade Universitária, Ilha do Fundão – RJ
laura@pep.ufrj.br

RESUMO

O *Variable Neighborhood Search* (VNS) é uma metaheurística na qual trocas sistemáticas de estruturas de vizinhança são realizadas com o objetivo de percorrer uma parcela maior do espaço de busca, tentando escapar dos ótimos locais para a obtenção da solução ótima global. No presente trabalho, esta estratégia é aplicada juntamente ao procedimento de *Path Relinking* para a solução do Problema de Coloração de Vértices. Este problema consiste na busca do número mínimo de cores possíveis para se colorir os vértices de um grafo de modo que vértices adjacentes não tenham a mesma cor, apresentando diversas aplicações práticas. Os resultados revelaram a eficiência do método ao encontrar soluções de boa qualidade em baixo tempo computacional para instâncias de tamanho moderado.

PALAVARAS CHAVE. Coloração de Grafos, Metaheurísticas, Variable Neighborhood Search, Path Relinking.

Área principal – Metaheurísticas, Teoria e Algoritmos em Grafos

ABSTRACT

The Variable Neighborhood Search (VNS) is a metaheuristic in which systematic exchanges of neighborhood structures are made in order to cover a larger portion of the search space, trying to escape from local optimal points to obtain the global optimum solution. In the present work, this strategy is applied with Path Relinking procedure to solve the Vertex Coloring Problem. This problem consists in finding the minimum number of possible different colors to color graph's vertices so that adjacent vertices don't have the same color. It has many practical applications, and the results showed the efficiency of the method to find good quality solutions at low computational time for moderate size instances.

KEYWORDS. Graph Coloring, Metaheuristics, Variable Neighborhood Search, Path Relinking.

Main area – Metaheuristics, Theory and Algorithms in Graphs

1. Introdução

Os estudos sobre o Problema de Coloração começaram em 1852, com a conjectura proposta por Francis Guthrie de que qualquer mapa desenhado em uma folha de papel podia ser colorido com apenas quatro cores, de forma que países fronteiriços não tivessem a mesma cor (Campello e Maculan, 1994). Na época, o matemático Augustus De Morgan tentou sem sucesso provar matematicamente esta conjectura, que só foi confirmada em 1976 por Kenneth Appel e Wolfang Haken (ver Appel e Haken, 1976; Appel e Haken, 1977; Appel, Haken e Koch, 1977), através de uma técnica de prova por computador.

Através de resultados já comprovados, Appel e Haken desenvolveram provas de que o problema geral pode ser reduzido a um número finito de casos particulares de configurações de grafos. Estes possuem certas propriedades que os permitem ser coloridos com quatro cores.

Dentre as aplicações práticas para o problema de coloração, é possível elencar os problemas de alocação de salas ou de programação de horários (*timetabling*) e *scheduling*, como pode ser visto em Silva e Silva (2009), Gardin e Hernandes (2008) e Leighton (1979), bem como o problema de alocação de canais em sistemas de telecomunicações (ver Silva, 2006).

Dado um grafo $G = (V, E)$, onde V é um conjunto de vértices e E um conjunto de arestas, o problema de coloração de um grafo G simples e não orientado consiste em colorir os vértices do grafo utilizando o menor número possível de cores, de tal forma que vértices adjacentes não recebam a mesma cor. Uma coloração que atenda a este requisito é chamada legítima, conforme pode ser visto na Figura 1.

Seja k uma coloração dada, $C_k \subseteq V$ o subconjunto de vértices com cor k para $k = 1, \ldots, n$ e $\chi(G)$ o número cromático do grafo G, considera-se que o grafo G admite uma $k - coloração$ ou é $k - colorível$ se $\chi(G) \leq k$, e $k - cromático$ quando $\chi(G) = k$.

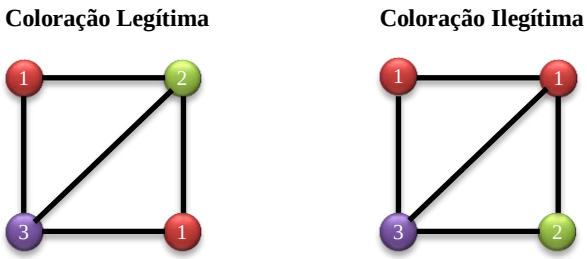

Figura 1 - Problema de Coloração de Vértices

Garey e Johnson (1979) afirmam que o problema de coloração de grafos é um problema NP-difícil, ou seja, pertence à classe de problemas que são pelo menos tão difíceis quanto qualquer problema em NP. A prova pertence a Stockmeyer (1973). Desta forma, não são conhecidos algoritmos exatos capazes de resolver tais problemas em tempo polinomial, o que traz como consequência um aumento exponencial do custo computacional para o tratamento de tais problemas à medida que cresce o tamanho das instâncias, sendo chamados então de problemas intratáveis.

Segundo Avanthay, Hertz e Zufferey (2003), os métodos exatos até então desenvolvidos podem ser aplicados apenas para problemas de tamanho relativamente pequeno (não mais de 100 vértices), explicando o porquê da aplicação de métodos heurísticos para a obtenção de um limitante superior de número cromático de um grafo.

O presente trabalho tem como objetivo aplicar a metaheurística *Variable Neighborhood Search* (VNS) com o procedimento de Path Relinking ao problema de coloração de vértices de um grafo; e apresenta a seguinte estrutura: No segundo tópico são apresentadas as técnicas utilizadas para o tratamento do problema no estudo efetuado. Na terceira seção do trabalho é descrita a metodologia utilizada no mesmo, explicitando a implementação de cada técnica utilizada. No quarto tópico são apresentados os resultados obtidos computacionalmente, e no quinto, as conclusões e recomendações para futuros trabalhos.

2. Técnicas Utilizadas

Várias são as estratégias aplicáveis para a obtenção de boas soluções viáveis no espaço de busca para o problema de coloração. Para este trabalho optou-se pela utilização do VNS em conjunção com o Path Relinking, que serão detalhados em seguida.

2.1. Variable Neighborhood Search

Os métodos convencionais de busca local para otimização combinatória em geral consistem na realização de uma sequência de alterações locais em uma solução inicial, visando a melhorias no valor da função objetivo até que um ótimo local seja encontrado. O VNS, proposto por Hansen e Mladenovic (1997), expande esta ideia de maneira simples ao alterar as estruturas de vizinhança onde são realizadas as buscas.

O método explora o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança cada vez mais distantes da solução atual e focaliza a busca em torno de uma nova solução somente se um movimento de melhoria é realizado. Portanto, a cada iteração uma solução melhorada s'' é obtida na vizinhança $N(s)$ da corrente solução s, até que não haja mais melhorias encontradas. Ele também conta com o auxílio de perturbações para ampliar a exploração de cada uma das vizinhanças. O algoritmo pode ser visto no quadro abaixo:

```
VNS (s0 = solução inicial e kmax = número de estruturas de vizinhança)
.  s ← s0 {s : solução corrente}
.  escolher um critério_de_parada
.        enquanto (critério_de_parada_não_satisfeito) faça
.        .        t ← 1 {t : tipo de estrutura de vizinhança}
.        .        enquanto (t ≤ tmax ) faça
.        .        .        gerar um vizinho qualquer s' ∈ N^t (s) {agitação}
.        .        .        s '' ← BuscaLocal(s ' )
.        .        .        se ( f (s '') < f (s) )
.        .        .        .        então s ← s '' ; t ← 1
.        .        .        .        senão t ← t + 1
.        .        .        fim-se
.        .        fim-enquanto
.        fim-enquanto
.        Retornar s
fim VNS.
```

Quadro 2 – Estrutura geral do algoritmo VNS

O VNS é dependente da estrutura de vizinhança e as vizinhanças N^t geralmente são aninhadas, embora não necessariamente. O número de vizinhanças é determinado experimentalmente.

O critério de parada pode ser determinado por uma série de fatores: número máximo de iterações, número máximo de iterações sem melhoria, tempo máximo de CPU permitido, *gap* (podendo ser este em relação ao ótimo conhecido, a algum limitante dual conhecido ou a algum limitante dual calculado), entre outros (Hansen e Mladenovic, 1997).

O trabalho de Hansen e Mladenovic (1999) evidencia diversas outras aplicações para o VNS, como o problema do caixeiro viajante, o problema de p-medianas, problemas contínuos de alocação de facilidades, e problemas de particionamento de conjuntos.

2.1. Path Relinking

O *Path Relinking* é um método proposto por Glover (para maiores detalhes, ver Glover et. al, 2000) para a busca de novas soluções, utilizando como premissa a exploração do trajeto que conecta soluções de boa qualidade.

Neste método é selecionada uma solução inicial e uma solução guia. Em seguida é criado um caminho ligando as duas, a partir da inserção dos atributos da solução guia na solução inicial, gerando assim as chamadas soluções intermediárias.

O objetivo da metodologia é explorar novas regiões do espaço de busca com potencial de obtenção de melhores soluções, dado que estas provavelmente compartilham atributos pertencentes às soluções de alta qualidade.

3. Metodologia

Nesta parte do trabalho é apresentada a função de avaliação, os passos para a obtenção da solução inicial, as estruturas de vizinhança utilizadas, a busca local e a forma como o VNS e *Path Relinking* foram implementados para a solução do problema em questão.

3.1. Função de Avaliação

Dado um inteiro k fixo, consideramos um problema de otimização chamado k-PCG (k-problema de coloração de grafo), que visa determinar uma k-coloração de G tal que o número de arestas conflitantes (arestas que possuem colorações inviáveis) é minimizado.

A função definida como o número total de arestas conflitantes ou arestas que possuem colorações inviáveis é a seguinte:

$$f(S) = \sum_{r=1}^{k} |E_r|$$

E_r representa a coleção de arcos de G com ambos os pontos finais no conjunto V_r (conjunto vértices de cor r). Portanto, o objetivo então é de minimizar o somatório de todas as arestas conflitantes, isto é, $\min f(S)$.

3.2. Solução Inicial

Grande parte dos métodos utilizados para a geração de soluções viáveis em problemas de coloração se baseia em algoritmos gulosos que buscam colorir os vértices de acordo com a ordem lexicográfica em que se encontram. Estes métodos geram soluções rápidas, mas a qualidade das mesmas está fortemente atrelada à forma como os vértices foram indexados.

Neste trabalho, as soluções iniciais foram geradas utilizando um método heurístico proposto pelo Brélaz (1979), chamado de algoritmo DSATUR. Este nome deve-se ao fato de que ele se utiliza do grau de saturação, que é o número de diferentes cores adjacentes a um dado vértice. Segundo o método, o próximo vértice a ser colorido é aquele que tem o maior número de cores diferentes já atribuídos aos seus vértices adjacentes. Vale lembrar que este algoritmo produz o ótimo global para grafos bipartidos.

```
DSATUR
. Disponha os vértices pela ordem decrescente de graus
. Colorir o vértice de grau máximo com a cor 1
.     enquanto (existirem vértices não coloridos) faça
.     .     Escolher um vértice com um grau de saturação máxima.
.     .     se (houver uma igualdade de grau de saturação)
.     .     .     Escolher qualquer vértice de grau máximo no
                  subgrafo não colorido
.     .     fim-se
.     .     Colorir o vértice escolhido com a mínima cor possível
.     fim-enquanto
fim DSATUR.
```

Quadro 2 – Algoritmo DSATUR

3.3. Estruturas de Vizinhança

Foram consideradas três estruturas de vizinhança para o algoritmo em questão, sendo as duas primeiras inspiradas no trabalho de Avanthay, Hertz e Zufferey (2003), e a terceira proposta neste trabalho.

Na primeira estrutura de vizinhança N^1, uma solução vizinha é gerada a partir de y movimentos, nos quais um dado vértice x em conflito é escolhido aleatoriamente e movido para a

melhor cor k possível. Um vértice em conflito é aquele pertencente a uma aresta conflitante, e a melhor cor possível para a realocação é aquela no qual o movimento apresenta o menor custo na função objetivo. O número y de trocas foi arbitrariamente escolhido como um número entre 1 e n (número de vértices) proporcional ao número de iterações sem melhoria realizadas no método VNS até então. Logo, as soluções s' geradas na vizinhança N^1 (s) são mais próximas de s no início da execução do algoritmo, se tornando cada vez mais diferentes à medida que método avança, aumentando o espaço de busca explorado.

Na segunda estrutura de vizinhança N^2, uma solução vizinha é gerada em duas etapas. Na primeira, um dado vértice x em conflito é escolhido aleatoriamente e movido para a melhor cor k possível. Em seguida, $y-1$ vértices em conflito pertencentes a esta mesma cor k são realocados para a melhor cor k' possível. O número y segue o mesmo padrão de N^1, sendo também proporcional ao número de iterações sem melhoria realizadas no método VNS até então. Caso o número de conflitos na cor k seja menor que y, o procedimento é reiniciado (a partir da $y$ésima iteração) com a escolha de um novo vértice conflitante. Diferente do algoritmo original, foi implementada também uma lista tabu de 10 posições, nos moldes da apresentada em Hertz e Werra (1987). O objetivo desta lista é impedir que um vértice x recém-alocado à cor k' seja realocado de volta à cor k no próximo movimento.

Por fim, na terceira estrutura de vizinhança N^3, uma solução vizinha é gerada a partir da alteração aleatória da cor de y vértices também escolhidos aleatoriamente dentre todos os vértices do grafo. O número y é obtido da mesma forma que em N^1 e N^2. Dada a grande diversidade das soluções geradas a partir desta estrutura, que se assemelha muito a um *restart* aleatório, foi utilizado um procedimento de reaproximação após a execução da busca local: n movimentos de realocação na melhor cor possível são realizados para vértices conflitantes escolhidos aleatoriamente, e em seguida uma nova busca local é executada.

3.4. Busca Local

O procedimento de busca local foi implementado tendo por base o algoritmo Tabucol proposto por Hertz e Werra (1987). Entretanto, optou-se por não utilizar uma lista tabu, e a estratégia adotada foi a de primeira melhoria. Esta escolha foi uma tentativa de tornar o método mais ágil e fruto da observação de que não ocorriam muitas ciclagens durante a execução do algoritmo.

Um vértice conflitante x é escolhido aleatoriamente e realocado também aleatoriamente a uma nova cor k. Caso o valor da função de avaliação da nova solução seja menor ou igual ao da solução antiga, ela se torna a solução corrente. Caso o valor seja menor que o da melhor solução encontrada até então pelo VNS, além se tornar a nova solução corrente, os parâmetros de iteração do VNS, iteração do Tabucol modicado e da vizinhança explorada (t) também são reiniciados.

Este procedimento se repete por m vezes, onde m o número de arestas do grafo.

3.5. Path Relinking Implementado

A variante de *Path Relinking* escolhida foi o *Backward Relinking*, na qual a melhor solução encontrada até então é utilizada como solução inicial (base) para o método, e a solução corrente ao fim da execução da busca local é a solução guia. Esta escolha deve-se ao fato de que no *Backward Relinking* a vizinhança da melhor solução encontrada até então é mais explorada, apresentando um maior potencial de descoberta de soluções de alta qualidade.

O algoritmo cria um vetor aleatório com a ordenação das posições em que as trocas devem ocorrer, e em seguida o percorre, fazendo com que para cada posição i o atributo da solução inicial receba o atributo da solução guia desta mesma posição.

3.6. O Método VNS Implementado

O método VNS implementado no trabalho gera uma k-coloração inicial através do algoritmo DSATUR e em seguida elimina uma cor, realocando os vértices retirados nas cores que resultem no menor valor possível da função objetivo. Isso fornece a solução inicial s, que é uma $(k-1)$-coloração inviável.

A partir daí, o algoritmo tenta viabilizar esta solução através da aplicação do procedimento de busca local e *Path Relinking* aos vizinhos gerados, sendo estes baseados nas estruturas de vizinhança explicitadas anteriormente. Caso uma solução gerada durante o processo tenha custo igual a zero (nenhum vértice em conflito), ela se configura como uma coloração legítima, e o valor de k é atualizado. Uma cor é novamente eliminada e o processo se reinicia. O algoritmo finaliza sua execução caso $3|V|$ iterações do VNS ocorram sem melhoria nas soluções encontradas.

VNS Implementado

```
. solucao_legitima, K ← Gerar solução inicial (DSATUR)
. se (K = 2)
.     Grafo Bipartido. solucao_legitima é a solução exata do problema
.     fim VNS Implementado.
. senão
.     s ← Eliminar a última cor alocada à solucao_legitima
.     iter_vns ← 0;
.     t ← 1;
.     enquanto (iter_vns ≤ 3.|V|) faça
.     .     Gerar uma solução s' aleatoriamente a partir da N(t) vizinhança de s
.     .     se (custo_s' < custo_s)
.     .     .     s ← s'
.     .     .     iter_busca_local ← 0;
.     .     enquanto (iter_busca_local ≤ m) faça
.     .     .     s'' ← BuscaLocal (s');
.     .     .     se (custo_s'' ≤ custo_s')
.     .     .     .     s' ← s'';
.     .     .     se (custo_s'' < custo_s)
.     .     .     .     s ←s'';
.     .     .     .     s' ← s'';
.     .     .     .     iter_vns ← 0;
.     .     .     .     iter_busca_local ← 0;
.     .     .     .     t ← 1;
.     .     .     .     se (custo_s'' = 0)
.     .     .     .     .     solucao_legitima ← s'';
.     .     .     .     .     K ← K - 1
.     .     .     .     .     se K = 2
.     .     .     .     .     .     Grafo Bipartido
.     .     .     .     .     .     fim VNS Implementado
.     .     .     .     .     senão
.     .     .     .     .     .     Eliminar última cor e Realocar vértices
.     .     .     .     .     fim se
.     .     .     .     fim se
.     .     .     senão
.     .     .     .     iter_busca_local++;
.     .     .     fim se
.     .     fim enquanto
.     .     PathRelinking(s, s1);
.     .     iter_vns++;
.     .     se (iter_vns > |V|)
.     .     .     t ← 2;
.     .     se (iter_vns > 2.|V|)
.     .     .     t ← 3;
.     .     fim se
.     fim enquanto
. fim se
fim VNS Implementado
```

Quadro 3 – Algoritmo VNS Implementado

É válido frisar que o número de iterações sem melhoria utilizado como critério de parada ($3|V|$) foi escolhido de forma a permitir cada vizinhança gerasse $|V|$ soluções, com y variando de 1 a $|V|$.

4. Resultados Computacionais

Os experimentos foram realizados em um PC com processador Intel® Core™ i5-2410M de 2.30GHz, com 6GB RAM e sistema operacional Windows® 7 x64, sendo o programa escrito em linguagem C e compilado usando o GNU GCC Compiler.

As instâncias de teste foram retiradas de ROIS (2012) e DIMACS (2012). A Tabela 1 apresenta os resultados obtidos para os testes realizados com instâncias pequenas e médias, tendo sido o algoritmo executado 100 vezes por instância. As colunas representam, respectivamente: o nome da instância; o número n de vértices do grafo; o número m de arestas do grafo; o grau de esparcidade (G.E.) do grafo, dado por $(n^2 - m)/n^2$; o número de cores K alvo considerado pelo algoritmo; o número de cores K^* ótimo para a instância; a diferença percentual entre o K alvo e a solução ótima (K^*) para a instância; o tempo médio demandado para obtenção do K alvo nas 100 execuções, em segundos; a média do tempo total de execução do algoritmo para as 100 rodadas, em segundos; e a porcentagem de soluções que atingiram um valor menor ou igual ao K alvo nas 100 execuções. Os tempos indicados como zero foram aqueles onde não foi possível realizar uma aferição precisa, dado que a execução durou menos de 1 milésimo de segundo.

Instância	*n*	*m*	*G.E.*	*K Alvo*	*K**	*Gap*	*TM Alvo*	*TM Execução*	*% Alvo*
Petersen	10	15	0,85	3	3	0%	0,0000	0,0003	100%
J_7_2_1	21	105	0,76	7	7	0%	0,0000	0,0022	100%
M_4	23	71	0,87	5	5	0%	0,0000	0,0017	100%
K7_3	35	70	0,94	3	3	0%	0,0000	0,0047	100%
Jeans	80	508	0,92	10	10	0%	0,0000	0,0321	100%
K9_4	126	315	0,98	3	3	0%	0,0000	0,1334	100%
Myciel7	191	2360	0,94	8	8	0%	0,0024	0,7340	100%
Queen15_15	225	10360	0,80	17	17	0%	2,1503	6,6039	12%
Queen15_15	225	10360	0,80	18	17	5,56%	0,1957	4,2796	99%
Queen16_16	256	12640	0,81	18	18	0%	1,9164	8,2597	13%
Queen16_16	256	12640	0,81	19	18	5,26%	0,2696	6,3950	100%

Tabela 1 - Resultados dos experimentos realizados com instâncias pequenas

A Tabela 2 apresenta os resultados obtidos para os testes realizados com instâncias maiores, tendo sido o algoritmo executado 5 vezes por instância. As colunas representam, respectivamente: o nome da instância; o número n de vértices do grafo; o número m de arestas do grafo; o grau de esparcidade (G.E.) do grafo, dado por $(n^2 - m)/n^2$; o menor número K de cores obtido nos testes; o menor número K^* de cores relatado na literatura; a diferença percentual entre K e K^*; e o tempo médio de execução do algoritmo, em segundos.

Instância	*n*	*m*	*G.E.*	*K*	*K**	*Gap*	*TM Execução*
DSJC500.5	500	62624	0,75	54	48	11,11%	161,77 (2,69 min)
DSJC1000.5	1000	249826	0,75	103	83	19,42%	3715,25 (61,9 min)
flat300_28_0	300	21695	0,76	36	28	22,22%	35,93
flat1000_76_0	1000	246708	0,75	102	76	25,49%	2904,22 (48,4 min)

Tabela 2 - Resultados dos experimentos realizados com instâncias grandes

O *Path Relinking* não trouxe grandes contribuições ao algoritmo para pequenos e

médios conjuntos de dados, tendo apresentado melhorias no valor da função objetivo (número de conflitos) em apenas 5% das execuções para a instância Queen15_15 e 6% para Queen16_16. Além disso, em nenhuma destas melhorias apresentadas o valor da função objetivo se igualou a zero, apresentando redução no número cromático do grafo em questão. Entretanto, para as instâncias maiores o método revelou sua importância, tendo contribuído para a redução do valor da função objetivo em quase todos os testes realizados.

Uma maneira de analisar a velocidade com que o algoritmo produz soluções de qualidade é através da distribuição empírica da probabilidade de se atingir um determinado valor alvo pré-definido (*K* alvo). Estes valores alvo são definidos de forma a permitir que o método sempre o alcance em um tempo viável, tendo sido esta metodologia proposta por Aiex et al. (2000) tomando como base o trabalho de Chambers (1983).

Para estas análises foram consideradas as instâncias Queen15_15 com valor alvo de 18 cores (*gap* de 5,56% em relação à solução ótima) e Queen16_16 com valor alvo de 19 cores (*gap* de 5,26% em relação à solução ótima). Estes valores de *K* alvo foram escolhidos porque, embora o algoritmo tenha encontrado soluções ótimas em tempo viável para todas as instâncias, este valor não foi atingido em todas as rodadas. Os resultados podem ser vistos nos gráficos abaixo.

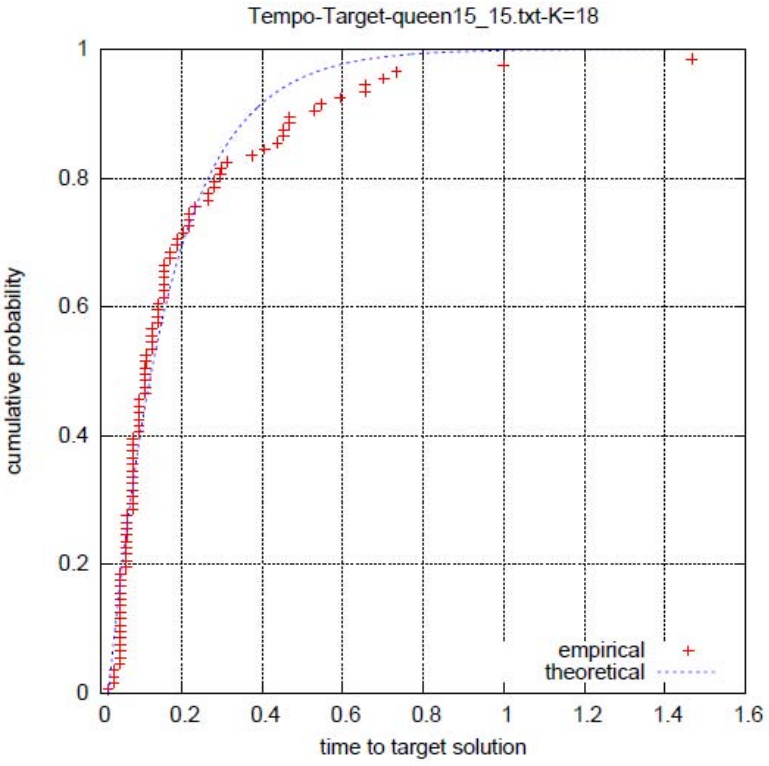

Figura 3 - Tempo para atingir o alvo - Queen15_15

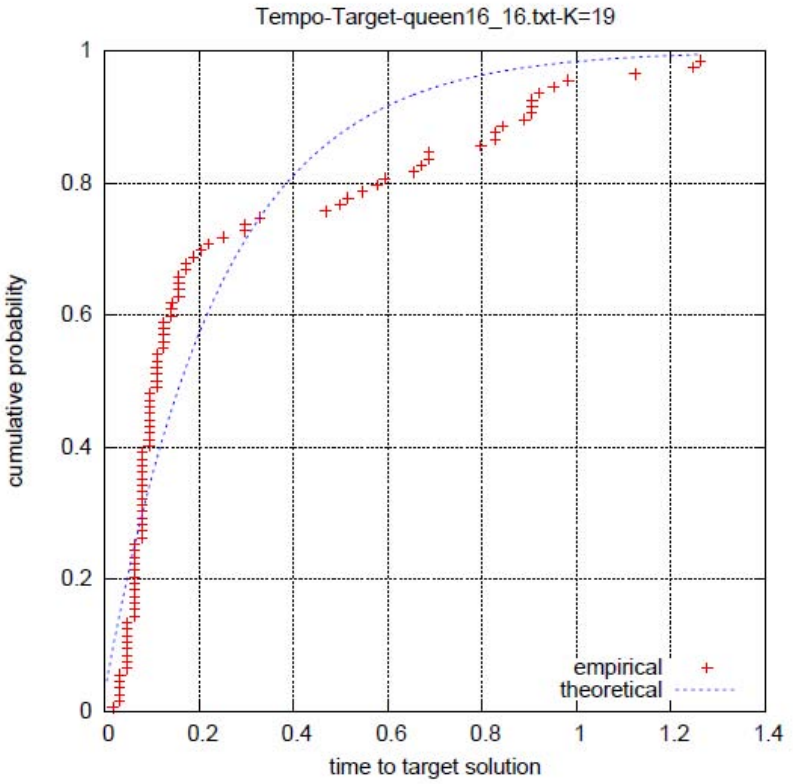

Figura 4 - Tempo para atingir o alvo - Queen16_16

5. Conclusões e Recomendações

Os resultados revelaram esta implementação de VNS como um método competitivo e eficiente para a resolução de Problemas de Coloração de Vértices em instâncias pequenas e médias. Entretanto, os *gaps* obtidos para conjuntos de dados maiores foram significativamente superiores ao de outras implementações relatadas na literatura, embora o tempo de execução também tenha sido consideravelmente inferior.

O *Path Relinking* não mostrou grande relevância para pequenos conjuntos de dados, mas obteve resultados expressivos com o aumento do tamanho das instâncias.

Soluções ótimas foram obtidas para todas as instâncias pequenas e médias, embora em algumas delas apenas para uma parcela reduzida de execuções; e soluções de boa qualidade foram obtidas em baixíssimos tempos computacionais para instâncias médias, com mais de 80% de probabilidade de que o alvo seja atingido em menos de 0,4 segundos para Queen15_15 e 0,6 segundos para Queen16_16, que são instâncias consagradas da literatura.

Como sugestão para trabalhos futuros, cabe a experimentação de novas estruturas de vizinhança, bem como outros métodos de busca local visando um ganho de robustez do algoritmo na busca de soluções ótimas.

Referências

Appel, K. e Haken, W. (1976), Every planar map is four colorable, *Bulletin of the American Mathematical Society*. Volume 82, Number 5, 711-712.

Appel, K. e Haken, W. (1977), Every Planar Map is Four Colorable Part I. Discharging, *Illinois Journal of Mathematics 21*, 429–490.

Appel, K., Haken, W. e Koch, J. (1977), Every Planar Map is Four Colorable Part II. Reducibility, *Illinois Journal of Mathematics 21*, 491–567.

Avanthay, C., Hertz, A. ,Zufferey, N. (2003). A variable neighborhood search for graph

coloring, *European Journal of Operational Research*, Switzerland;

Aiex, R. M., Resende, M. G. C. e Ribeiro, C. C. (2000). Probability distribution of solution time in grasp: An experimental investigation. *Journal of Heuristics*, 8, 200–2.

Brélaz, D. (1979). New Methods to Color the Vertices of a Graph. *École Polytechnique Fédérale de Lausanne*.

Chambers, J. M., *Graphical Methods for Data Analysis (Statistics)*, Chapman & Hall/CRC, 1983.

Campello, R. E. e Maculan, N., *Algoritmos e Heurísticas*, Editora da Universidade Federal Fluminense, Niterói, 1994.

DIMACS Graphs, *Benchmark Instances and Best Upper Bounds* (http://www.info.univ-angers.fr/pub/porumbel/graphs/ 08-2012

Gardin, E. e Hernandes, F. (2008). Aplicação da coloração em grafos fuzzy no problema de distribuição de aulas, *Revista Eletrônica Lato Sensu*, Universidade Estadual do Centro-Oeste – UNICENTRO;

Garey, M. e Johnson, D., *Computers and intractability: A guide to the theory of NP-completeness*, W.H. Freeman and Company, New York, 1979;

Glover, F.; Laguna, M.; Martí, R. (2000) Fundamentals of scatter search and path relinking. *Control and Cybernetics*, n. 42743;

Hansen, P. e Mladenovic, N. (1999). Variable Neighborhood Search: Principles and applications, *European Journal of Operation Research*, Canada;

Hansen, P. e Mladenovic, N. (1997). Variable Neighborhood Search, *Computers & Operations Research*, Volume 24, Issue 11, 1097–1100.

Hertz, A.; Werra, D. (1987). Using tabu search techniques for graph coloring. *Computing*, v. 39, n. 4, 345–351;

Leighton, F. T. (1979). A graph coloring algorithm for large scheduling problems. *Journal of Research of the National Bureau of Standards* 84, 489-506;

ROIS – Registry for Optimization Instances and Solutions, *Display Coloring Solutions* (http://mat.tepper.cmu.edu/ROIS/solutions/coloring/display_sol.php 08-2012

Silva, D. J., Silva, G. C. (2009). Heurísticas baseadas no algoritmo de coloração de grafos para o Problema de alocação de salas em uma instituição de ensino superior, *Anais do XLII Simpósio Brasileiro de Pesquisa Operacional*.

Silva, M.W.R. da. *Alocação de Canal em Redes Sem Fio IEEE 802.11 Independentes*. Dissertação de Mestrado, Universidade Federal do Rio de Janeiro (Programa de Engenharia Elétrica da COPPE/UFRJ), Rio de Janeiro, 2006.

Stockmeyer, L. (1973). Planar 3-colorability is polynomial complete, *ACM SIGACT News*. Volume 5, Issue 3, 19 - 25;