Uma implementação de Geração de Colunas para o problema da coloração de arestas

Lucas Gonzalez de Queiroz[1], **Weber Veloso Mourão**[1]

[1]Faculdade de Computação – Universidade Federal de Mato Grosso Sul (UFMS)
Caixa Postal 549 – 79.070-900 – Campo Grande – MS – Brazil

lucasgq71@gmail.com, webermourao@gmail.com

Resumo. *Este trabalho apresenta o conceito geral de desenvolver um algoritmo de geração de colunas que resolva o problema da coloração em vértices de grafos (VCP). Ou seja minimizar o numero de cores em todos os vértices, de modo que cada vertice seja associado a uma cor e cada aresta conecte vertices de cores distintas.*

1. Introdução

O problema da coloração de um grafo consiste em atribuir cores aos vértices do grafo de modo que vértices incidentes a uma mesma aresta recebam cores distintas, usando para tal o menor número de cores possível. O objetivo deste trabalho é implementar um algoritmo de geração de colunas para o VCP, utilizando a biblioteca GLPK, que é uma das bibliotecas de otimizacão com recursos para resolver problemas de Programacão Linear Inteira (PLI).

2. Heurística Inicial

Para minimização da coloração em vértices foram desenvolvidos basicamente duas funções de heurísticas, uma para o grafo inicial e outra para a clique gerada por ele.

Para desenvolvimento da Heurística desenvolveu-se um algoritmo Best First em que ao aplicar uma cor ao vértice atual, verifica-se se os vértices adjacentes que possuem as mesmas cores, atribuindo-lhes cores diferentes seguindo assim até o final do grafo.

3. Clique

Tendo como base o conceito de que conjuntos estáveis estão intimamente relacionados com cliques, ou seja um conjunto E de vértices é estável em um grafo (supondo implicitamente que o grafo não tem laços), se e somente se, E é uma clique no grafo complementar. Assim, para desenvolvimento da clique desenvolvemos o complemento do grafo e aplicamos a mesma função heurística do grafo normal, retornando assim, todas as cliques.

4. Experimentos

A máquina na qual o experimento foi realizado possui as seguintes configurações básicas:

- **16 GB de memória RAM DDR3**, 1333MHz ;
- **1 TB de HDD**;
- Sistema Operacional: **Ubuntu 16.04 LST**;
- Arquitetura: **AMD64**;

- **Processador Intel Core i5 - 5200U** com tecnologia *Turbo Boost 2.0*, que acelera o desempenho do processador, fazendo com que os núcleos do processador trabalhem mais rapidamente do que a frequência operacional nominal quando estiverem operando abaixo dos limites especificados para energia, corrente e temperatura, fazendo com que a frequência varie de 2.2 GHz até 2.7 GHz. Este processador possui cache de 3 MB, 2 cores e 4 *threads*.

Abaixos serão representados os graficos referentes a tempo, colunas geradas e quantidade de iterações feitas pelas heuristicas. Serão comparados a base, heuristica feita em classe com base e também a heuristica desenvolvida neste trabalho, possuindo o nome de ”MYheur”.

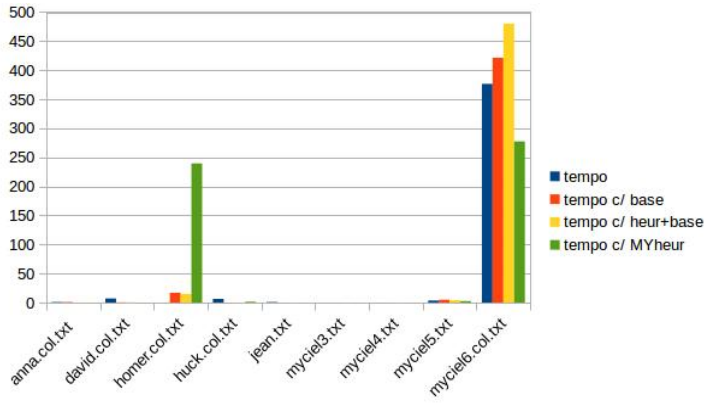

Figura 1. Tempo decorrido pelas instâncias

Neste gráfico, com estas instãncias, MYheur com base provou ser mais eficiente, utilizando menos tempo do que as outras, como por exemplo, base sozinha mostrou-se ineficiente pois demorou muito tempo na primeira instância.

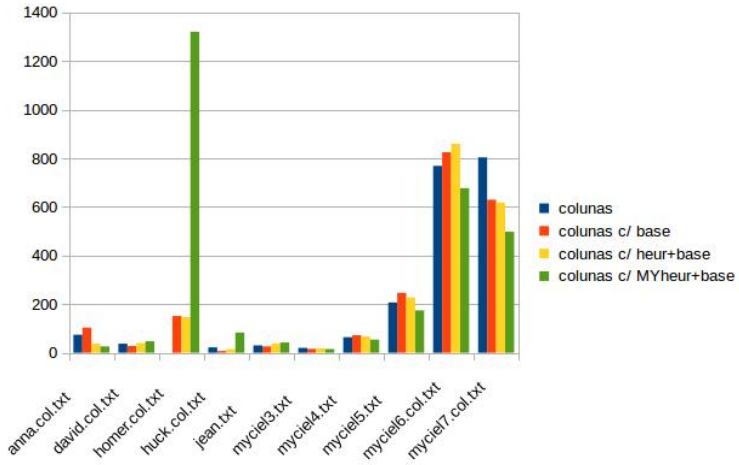

Figura 2. Colunas geradas pelas instâncias

Por fim, apresentaram-se os resultados referentes a quantidade das colunas geradas por cada heurística, onde a heuristica primal + base possuiu melhores resultados para instancias maiores e a base foi melhor em instancias menores.

A seguir, serão demonstradas tabelas para melhor visualizacão dos valores que obtivemos com os testes.

Tabela 1. Tempo Heuristicas

instancia	tempo	tempo c/ base	tempo c/ heur+base	tempo c/ MYheur
anna.col.txt	0,813818	0,932468	0,133477	0.013153
david.col.txt	7,07237	0,299976	0,264801	0.07099
homer.col.txt	*	16,9664	14,9728	239.526
huck.col.txt	6,33235	0,032554	0,037621	1.45555
jean.txt	0,892114	0,147757	0,155149	0,140072
myciel3.txt	0,006445	0,001662	0,004021	0.029615
myciel4.txt	0,099802	0,132735	0,079816	0.042966
myciel5.txt	3,50114	4,96637	4,15286	2.56289
myciel6.col.txt	376,504	421,52	480,166	277.396
myciel7.col.txt	*	*	*	1200.02
games120.col.txt	*	0,332168	0,910168	*
miles1500.col.txt	1140,36	155,853	58,3833	*
mulsol.i.1.col.txt	315,471	18,26	12,232	*
zeroin.i.1.col.txt	211,734	5,88775	1,07835	*

Tabela 2. Tabela comparativa com as colunas geradas por instância

instancia	colunas	colunas c/ base	colunas c/ heur+base	colunas c/ MYheur+base
anna.col.txt	74	103	37	26
david.col.txt	37	28	40	47
homer.col.txt	0	151	147	1321
huck.col.txt	22	7	15	83
jean.txt	30	26	37	42
myciel3.txt	20	16	18	15
myciel4.txt	64	72	67	54
myciel5.txt	207	246	227	174
myciel6.col.txt	769	825	860	677
myciel7.col.txt	804	630	618	498
games120.col.txt	1	33	44	*
miles1500.col.txt	76	35	40	*
mulsol.i.1.col.txt	103	16	109	*
zeroin.i.1.col.txt	169	21	61	*

Analisando as tabelas, podemos notar que MYheur obteve resultados ótimos, em relacão as outras heuristica, mas, também possuiu resultados ruins, principalmente para instancias maiores, onde seu desempenho e suas falhas são em número elevado.

5. Conclusão

A partir da análise dos graficos e tabelas, percebemos que, mesmo sem o desenvolvimento da heurística de pricing, há uma melhora significativa, no número de iterações e consequentemente no tempo de execução de todos os grafos. Porém quando comparados

Tabela 3. Tabela referente aos valores de Z e as falhas

instancia	falhas MYheur	falhas heur	Zlp	zlp (compact)
anna.col.txt	1	0,2368421053	11	2
david.col.txt	8	0,3658536585	11	2
homer.col.txt	22	0,1486486486	13	*
huck.col.txt	27	0,5	11	2
jean.txt	4	0,3421052632	10	2
myciel3.txt	12	0,8947368421	2,9	2
myciel4.txt	42	0,8382352941	3.24483	2
myciel5.txt	149	0,9254385965	3.55301	2
myciel6.col.txt	636	0,9558652729	3.83446	2
myciel7.col.txt	414	0,9093851133	4.67592	*
games120.col.txt	*	0,6	9	2
miles1500.col.txt	*	0,487804878	73	2
mulsol.i.1.col.txt	*	0,1	49	2
zeroin.i.1.col.txt	*	0,0483870968	49	2

com os dados disponibilizados, percebe-se que houve uma oscilação entre determinados grafos, tais quais os grafos de nome ”myciel”em que a nossa heurística mostrou-se um pouco mais rápida e o ”homer”em que houve menor desempenho. Após o desenvolvimento da heurística de pricing e analizando os resultados, notamos que houve melhora na geração de novas colunas e poucas falhas para a maioria dos grafos, porém apenas o grafo homer.txt apresentou desempenho bastante inferior, tal fato ocorre devido ao grande número de arestas e também da quantidade de vértices desse grafo, gerando assim mais falhas na heurística de pricing, consequentemente mais iterações são realizadas e mais tempo é necessário para computá-las.

Referências

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

McCormick, S. T. (1983). Optimal approximation of sparse hessians and its equivalence to a graph coloring problem. *Mathematical Programming*, 26(2):153–171.

Mehrotra, A. and Trick, M. A. (1996). A column generation approach for graph coloring. *informs Journal on Computing*, 8(4):344–354.