Estudo Heurístico Baseada no Algoritmo de Coloração de Grafos Aplicado ao Problema de Atribuição de Horários e Salas de Aula para Disciplinas

Gabriel Carvalho de Araujo, *Universidade Federal de Roraima, UFRR*, Hermino Barbosa de Freitas Júnior, *Universidade Federal de Roraima, UFRR*

***Abstract*—Este estudo tem como objetivo aplicar dois algoritmos para solucionar o problema de alocação de horários de disciplina do Departamento de Ciência da Computação da Universidade Federal de Roraima. Foi usado um algoritmo guloso para encontrar uma solução aproximadamente próxima e a técnica de força bruta para encontrar as permutações possíveis da grade curricular semestral.**

***Index Terms*—grafo, coloração, complexidade, NP-completo.**

I. Introdução

Para matemática, um grafo é considerado como um conjunto de objetos em que alguns pares de objetos são conectados por *links*. Os objetos interconectado são geralmente chamados de vértices e os *links* conectando os pares de vértices são chamados de vértices. Os grafos pode ser usados para modelar um conjunto muito extenso de problema do mundo real. O problema de coloração vértices de um grafo é um dos problema mais famoso no campo da teoria dos grafos.

A coloração de vértices de um grafo $G(V, E)$ é uma atribuição de cores de modo que nenhum dos vértices adjacentes recebam a mesma cor [5]. Nesse sentido o número mínimo de cores necessárias para tal coloração é chamado de número cromático de G, sendo denotado por $\chi(G)$. Na Figura 1 temos um exemplo de coloração de grafo, onde o grafo G possui 10 vértices e 21 arestas e ainda mostra o grafo colorido com 5 cores diferentes. Nesse sentido pode dizer que a coloração do grafo é adequada, porque todos os pares de vértices unidos por arestas foram atribuídos com cores diferentes.

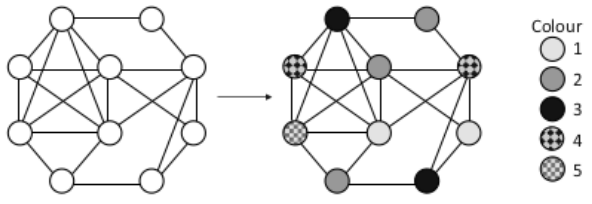

Fig. 1. Um pequeno gráfico e correspondente 5-color. Adaptado de 6

Para determinar o número cromático de um grafo é preciso saber que esse é problema da classes de problemas *NP-difícil* 2, para isso pesquisadores procuram aproximações desses problema para as suas aplicações. Com isso naturalmente surgiu uma variedade de aplicações, como alocação de registradores, cronograma ou programação de exames. Essa aplicações podem ser formuladas como problemas de coloração de grafos, bastando apenas encontrar uma coloração de grafo aproximadamente ótima, ou seja, uma coloração do grafo com um número pequeno de cores mas não ótima.

Diante disso, é possível resolver uma serie de problema com coloração de grafo um que é muito comum ser tratado com esse abordagem é o problema sobre oferta de disciplinares de um curso. A primeira vista não parece um problema custoso, mas na verdade trata-se de um problema NP-Completo, pois o mesmo estabelece diversas variáveis, tornado-o muito custoso. Sabemos então, que é procedimento comum que os departamentos de universidades, ficam a cargo de criarem a oferta de disciplinas que sejam compatíveis com os horários dos professores no semestre e sem que haja choques horários entre as disciplinas. Esse procedimento muitas vezes exige uma logística, que poderia ser evitada com algum software ou algoritmo que gere automaticamente os horários para as disciplinas.

Por este motivo, nesse trabalho será implementado um algoritmo para coloração de grafo, afim de encontrar uma coloração adequada para o problema de horários da grade disciplinar semestral do Departamento de Ciência da Computação (DCC) da Universidade Federal de Roraima (UFRR). Será desenvolvido um algoritmo que gera as que gera os horários baseados nas cores atribuídas nas disciplinas. Lembramos que o departamento conta com poucos professor, nesse sentido faz necessário um solução que otimize esse processo de decisão de oferta das disciplinas.

Para isso este trabalho será dividido na seção 2 que descreve os trabalhos relacionados. Seção 3 onde descreve o método proposto. Finalmente a seção 4 onde será dada as considerações finais.

II. Trabalhos Relacionados

No trabalho de 5 é proposto um algoritmo de tempo polinomial randomizado que colore um grafo com 3 cores e n vértices com no $\min\{min = O(\Delta^{\frac{1}{3}} \log^{\frac{1}{2}} \Delta \log n), O(n^{\frac{1}{4}} \log^{\frac{1}{2}} n)\}$ cores, onde Δ é o grau máximo de qualquer vértice. Além

disso, fornece a relação de aproximação, conhecida em termos de n. Isso mostra o primeiro resultado de aproximação não trivial em função do grau máximo. Ainda 5 cita que o resultado pode ser generalizado para grafos k-coloráveis para obter uma coloração usando no $\min\{O(\Delta^{\frac{1-2}{k}} \log^{\frac{1}{2}} \Delta \log n), O(n^{\frac{1-3}{k+1}} \log^{\frac{1}{2}} n)\}$ cores.

Em vez de 5 atribuir cores aos vértices de um grafo, o mesmo considerou atribuir vetores unitários. Assim, para capturar a propriedade de um coloração, os vetores de vértices adjacentes devem ser diferentes de uma maneira natural. O vetor k-cor definido, possui o papel hipotético de "k-coloração fracionária" que teria uma abordagem de programação linear. Nesse sentido 5 dá a seguinte definição:

Definição: dado um grafo $G(V, E)$ com n vértices, um número real de $k \geq 1$, um vetor k-colorável de G é uma atribuição de vetores unitários v_i do espaço $\mathbb{R}^n$ para cada vértice $i \in V$, tal que para quaisquer dos vértices adjacentes i e j é o produto de seus vetores satisfaz a desigualdade:

$$(v_i, v_j) \leq -\frac{1}{k-1} \tag{1}$$

O relaxamento da coloração vetorial foi resolvido usando programação semidefinida. Logo para [5] resolver o problema usando a seguinte definição auxiliar:

Definição: dado um grafo $G(V, E)$ com n vértices, uma matriz k-coloração do grafo é uma matriz M semidefinida positiva simétrica $n \times n$, como $m_{ii} = 1$ e $m_{ij} \leqslant -1/(k-1)$ $if\{i, j\} \in E$.

Além de 5, 4 propôs uma versão de algoritmo guloso que fornece um algoritmo de aproximação $O(n/\log n)$ cores para colorir um grafo k-colorável. Já 7 obteve resultado melhor que 4 propondo um algoritmo com complexidade $O(n^{1-\frac{1}{(k-1)}})$. Posteriormente, outros algoritmos de tempo polinomial foram fornecidos 1 que tem complexidade $O(n^{\frac{3}{8}} \log^{\frac{8}{5}} n)$ para colorir um grafo com 3 cores e n-vértice. Esse resultado generaliza para colorir um grafo k-colorável com cores $O(n^{1-\frac{1}{(k-\frac{4}{3})}} \log^{\frac{8}{5}} n)$. 3 forneceu um algoritmo de tempo polinomial usando um número de cores que está dentro de um fator de $O(n(\log\log n)^2/\log^3 n)$.

III. Método Proposto

Nesta seção será demonstrado como foi desenvolvido este estudo. Primeiramente analisou se as variáveis mais importantes e necessárias para modelar uma solução adequada ao cenário DCC. Nesse sentido, é importante encontrar uma solução aproximada para este problema, de forma que os horários dos professor se adéquem a sua rotina. Assim, o responsável por organizar os horários do DCC, por exemplo, o coordenador do curso poderá está alimentando o algoritmo e verificando quais das alternativas geradas que melhor beneficia o corpo docente do DCC. Posto isso, nas próximas seções discutiremos a entrada para o algoritmo.

Este estudo utilizou a linguagem C para implementar os algoritmo. Dessa forma no Código **??** vemos os *struct's* construídos para implementação do algoritmo. No *struct* Disciplina armazenará as variáveis código da disciplina como o nome da disciplina, a carga horária, os pré requisitos da disciplina, o professor, a turma, por exemplo se a turma é do 1º, 3º, 5º ou 7º semestre; quantidade de alunos a mesma possui, os recursos que a disciplina utiliza, isto é, se necessita de projetor, quadro iterativo ou laboratório; o número de conexões, o horário em que ela é dada e a conexão como outra disciplina. Já o *struct* Disciplinas, cria uma lista da disciplinas e a quantidade de matérias que o semestre possui. No *struct* Sala é armazenado as informações sobre as salas, como o código, a capacidade que a mesma tem e os recursos disponíveis nela. O *struct* Salas cria uma lista de salas e quantidade de salas que estão disponíveis.

```
typedef struct Disciplina{
  char codigo[7];
  char nome[60];
  int cargahoraria;
  char prerequisitos[18];
  char docente[30];
  int turma;
  int qtd_alunos;
  int recursos[3];
  int nConexoes;
  int horario;
  struct Disciplina **conexoes;
} Disciplina;

typedef struct Disciplinas{
  Disciplina **listadisciplinas;
  int quantidade;
} Disciplinas;

typedef struct Sala{
  char codigo[5];
  int capacidade;
  int recursos[3];
} Sala;

typedef struct Salas{
  Sala **listasalas;
  int quantidade;
} Salas;
```

Fig. 2. Struct's utilizados para implementação do algoritmod e coloração de grafos.

Na Figura 3 é possível ver um exemplo hipotético de instância da primeira entrada do algoritmo, relativa as disciplinas ofertadas pelo DCC no segundo semestre de um ano. Na tabela, é possível observar atributos como o código é identificação da disciplina, o nome da disciplina, a carga horária, os pré-requisitos, o professor que leciona a disciplina, as turmas e qual semestre as mesma está, o número de alunos, se a disciplina necessita de quadro iterativo, se a disciplina necessita de projetor e se a aula deve ser em um laboratório.

Ainda nesse sentido, exploramos informações sobre as sala de aula disponíveis, para assim verificar se a mesma atende o requisito de uma disciplina. Dessa forma, foi construído a tabela na Figura 4, para ser a segunda entrada do algoritmo. Assim, cada sala possui um código de identificação, uma capacidade de alunos, se a mesma possui um quadro iterativo, se possui um projetor ou se a sala é um laboratório.

As relações ou ligações dos nós no algoritmo, serão dadas

CÓDIGO	NOME	C.H	REQUISITO	PROF.	TUR./SEM	ALN	Q. INTER	PROJETOR	LAB
CSC04	M. E T. DO TRAB. CIENT.	60	-	-	1	41	0	1	0
DCC103	INTR. A SIST. DE COMP.	60	-	PROF. 1	1	41	0	1	0
DCC104	LÓGICA PROPOSICIONAL	60	-	PROF. 2	1	41	0	0	0
DCC105	ALGORITMOS	60	-	PROF. 3	1	41	0	1	1
DCC106	ELETRICIDADE BÁSICA	60	-	-	1	41	1	1	0
MAT100	PRÉ-CÁLCULO	60	-	-	1	41	0	1	0
AD310	FORMAÇÃO PROF. DO ADM.	60	-	-	3	14	0	1	0
DCC301	ARQ. E ORG. DE COMPUTADORES	60	DCC204	PROF. 4	3	20	0	1	0
DCC302	ESTRUTURAS DE DADOS I	90	DCC205	PROF. 5	3	23	1	1	1
DCC305	PROGR. ORIENTADA A OBJ.	60	DCC205	PROF. 5	3	23	1	1	1
MAT05	CALC. DIF. E INTEGRAL II	90	MAT01/MAT04	-	3	38	0	0	0
MAT06	ÁLGEBRA LINEAR I	90	MAT04	-	3	25	1	0	0
DCC502	BANCO DE DADOS I	60	DCC405	PROF. 6	5	24	0	1	1
DCC507	REDES DE COMPUTADORES II	60	DCC407	PROF. 7	5	20	1	1	1
DCC508	FUNDAMENTOS DA COMPUTAÇÃO	60	DCC405/MA302	PROF. 3	5	25	1	1	0
DCC509	ENGENHARIA DE SOFTWARE II	60	DCC402	PROF. 2	5	30	0	1	0
DCC510	PROGRAMAÇÃO EM BAIXO NÍVEL	60	DCC301	-	5	15	0	1	1
DCC511	LÓGICA DE PREDICADOS	60	DCC104	-	5	14	1	1	0
DCC703	COMPUTAÇÃO GRÁFICA	60	DCC305/MAT04	PROF. 8	7	8	1	1	1
DCC703	ARQ. E TEC DE SISTEMAS WEB	60	DCC407/DCC502	PROF. 6	7	22	1	1	1

Fig. 3. Disciplinas ofertadas no primeiro semestre do ano letivo, com o recursos necessário para atender seus requisitos.

CÓDIGO	CAPACIDADE	Q. ITERATIVO	PROJETOR	LAB
501	50	0	1	0
502	60	0	1	1
503	50	0	1	0
504	20	1	1	1
505	30	0	1	0
506	30	1	1	0
507	30	0	1	1
508	20	1	1	0

Fig. 4. Variáveis utilizadas para construção do Algoritmo.

com base nas seguintes informações:

- Se um par de nós pertencem a mesma turma, ou seja, se os pertencem por exemplo a turma do 3º semestre.
- As disciplinas lecionadas pelo mesmo professor.
- As disciplinas que requerem os mesmos recursos em sala de aula.

Foi predefinido para cada horário uma cor. Na Figura 5 é possível ver um exemplo das cores e sua nomenclatura usada para identificá-la. Assim, foi verificado se naquela coloração (horário) os requisitos da disciplina fora estabelecidos. Nesse sentido, a coloração do grafo atenderá as seguintes restrições:

	SEGUNDA	TERÇA	QUARTA	QUINTA	SEXTA
8:00 - 10:00	R	B	R	B	-
10:00 - 12:00	G	Y	G	Y	-
14:00 - 16:00	2R	2B	2R	2B	-
16:00 - 18:00	2G	2Y	2G	2Y	-

Fig. 5. Horários disponível com suas respectivas cores.

- Se os nós vizinhos foram possuem cores diferentes.
- Se todos os requisitos (e.g sala com projetor, sala com capacidade para 30 alunos, sala como quadro iterativo, etc) foram atendidos, por aquela coloração.

Como dito anteriormente os dados de entrada serão dados pelo responsável por montar o calendário disciplinar semestral. Dado os dados de entrada em um arquivo de texto *Comma Separated Values* (CSV), para ser lido pelo algoritmo, dessa forma o algoritmo será responsável por tratar esse dados submetidos e gerar um resultado.

IV. Resultados e Discussões

Para os experimentos realizados no estudo, primeiramente foi feito uma análise usando uma instância hipotética, onde dada uma entrada de disciplinas, o grafo foi colorido usando a heurística de grau. Dessa forma obtemos o grafo apresentado na Figura 6, onde cada cor representa um horário. No grafo apresentado os nós foram relacionados de acordo com as turmas do respectivo semestre e pelas disciplinas que possuem o mesmo professor. Essa relação é importante para evitar choque horários caso o aluno tenha reprovado e também evitar que o professor pegue duas disciplinas no mesmo horário.

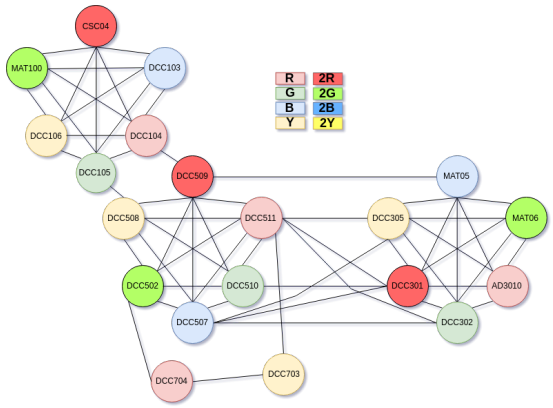

Fig. 6. Grafo gerado a partir de uma instância de disciplinas hipotética.

Nesse sentido, é necessário explicar o que cada cor apresentada no grafo significa. Logo na Figura 5 temos as cores por cada horário. Cada disciplina possui duas aulas por semana, por isso as cores se repetem apenas duas vezes. E ainda, as sextas feira são dedicadas para os professores tirarem dúvidas dos alunos em sua sala.

```
main.exe
HORARIO: r | DISCIPLINA: DCC104 | SALA: 502
HORARIO: r | DISCIPLINA: MAT006 | SALA: 506
HORARIO: r | DISCIPLINA: DCC507 | SALA: 504
HORARIO: r | DISCIPLINA: DCC703 | SALA: 507
HORARIO: g | DISCIPLINA: DCC105 | SALA: 501
HORARIO: g | DISCIPLINA: DCC301 | SALA: 504
HORARIO: g | DISCIPLINA: DCC509 | SALA: 505
HORARIO: g | DISCIPLINA: DCC704 | SALA: 507
HORARIO: b | DISCIPLINA: CSC004 | SALA: 502
HORARIO: b | DISCIPLINA: DCC302 | SALA: 504
HORARIO: b | DISCIPLINA: DCC510 | SALA: 507
HORARIO: y | DISCIPLINA: DCC103 | SALA: 502
HORARIO: y | DISCIPLINA: DCC305 | SALA: 504
HORARIO: y | DISCIPLINA: DCC502 | SALA: 507
HORARIO: R | DISCIPLINA: DCC106 | SALA: 503
HORARIO: R | DISCIPLINA: MAT005 | SALA: 505
HORARIO: R | DISCIPLINA: DCC508 | SALA: 506
HORARIO: G | DISCIPLINA: MAT100 | SALA: 502
HORARIO: G | DISCIPLINA: ADM310 | SALA: 505
HORARIO: G | DISCIPLINA: DCC511 | SALA: 506
```

Fig. 7. Saída do algoritmo de coloração.

Agora vejamos na Figura 7 o resultado gerado pelo algoritmo dado a instância hipotética de disciplinas. É possível

ver que o algoritmo conseguiu selecionar apenas 6 cores das oito apresentadas anteriormente. Como isso o algoritmo gera o grafo apresentado na Figura 8. É possível ver também que o mesmo atendeu todos os requisitos das disciplinas, fazendo com que os professores tenham aulas em horários diferentes.

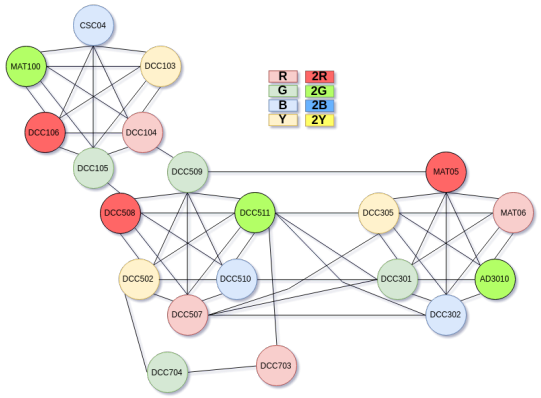

Fig. 8. Solução encontrada pelo algoritmo dada uma instância hipotética de disciplinas.

Comparando a Figura 6 e a Figura 8 é possível perceber que houve uma pequena variação na coloração do grafo e que tudo ocorreu de maneira correta. As disciplinas foram coloridas com no máximo 6 cores e os horários ficaram distribuídos como na Figura 9. Com isso conseguimos gerar um horário para a oferta disciplinar semestral do DCC de forma automática é alocar cada disciplina em sala, sem haver choques de horários.

	SEGUNDA	TERÇA	SEGUNDA	TERÇA	SEXTA
8:00 - 10:00	DCC104 - SALA 502 MAT06 - SALA 506 DCC507 - SALA 504 DCC703 - SALA 507	CSC204 - SALA 502 DCC302 - SALA 504 DCC510 - SALA 507	DCC104 - SALA 502 MAT06 - SALA 506 DCC507 - SALA 504 DCC703 - SALA 507	CSC204 - SALA 502 DCC302 - SALA 504 DCC510 - SALA 507	-
10:00 - 12:00	DCC105 - SALA 501 DCC301 - SALA 504 DCC509 - SALA 505 DCC704 - SALA 507	DCC103 - SALA 502 DCC305 - SALA 504 DCC502 - SALA 507	DCC105 - SALA 501 DCC301 - SALA 504 DCC509 - SALA 505 DCC704 - SALA 507	DCC103 - SALA 502 DCC305 - SALA 504 DCC502 - SALA 507	-
14:00 - 16:00	MAT05 - SALA 505 DCC106 - SALA 503 DCC508 - SALA 506	-	MAT05 - SALA 505 DCC106 - SALA 503 DCC508 - SALA 506	-	-
16:00 - 18:00	MAT100 - SALA 502 AD3010 - SALA 505 DCC511 - SALA 506	-	MAT100 - SALA 502 AD3010 - SALA 505 DCC511 - SALA 506	-	-

Fig. 9. Horários como os resultados gerados pelo algoritmo.

Neste estudo, foi realizado os calculo de complexidade de algumas funções que executam mais instruções dada a entrada, logo temos:

- Função **lerDisciplinas**(): nessa função possuímos o seguinte somatório $\sum_{i=0}^{n} + \sum_{i=0}^{n} \sum_{j=0}^{i} 1$, assim obtemos o seguinte resultado $n^2 + 5n + 4$.
- Função **colereGrafo**(): nessa função foi calculado o somatório $\sum_{i=0}^{n} G_i + \sum_{j=0}^{n} G_i + G_j$ e obteve-se o $n^2 + n + G_i n + G_i$.
- Impressão do Grafo: resulta no somatório $\sum_{i=0}^{9} \sum_{j=0}^{n} \sum_{k=0}^{m} 1$ igual à $9(n + n \times m)$.

V. Considerações Finais

Concluímos que gerar uma coloração para disciplinas de uma grade curricular e possível, mas não se trata de um problema trivial, que possa ser resolvido como pouco esforço. Nesse estudo ainda foi considerado que a complexidade do algoritmo proposto é $O(n/\log n)$, em que k é o número de cores e n é o número de disciplinas. Nos casos de testes, foram usadas um pequena instância de entrada já que o DCC oferta no máximo 22 disciplinas obrigatórias por semestre. Os estudo não considerou as disciplinas optativas, pois as mesmas mudam a todo semestre.

Lembrando que o DCC conta com pouco professores e por isso, torna se um desafio organizar horários para as disciplinas, devido aos choques de horários que ocorrem ou pela falta de possibilidade do professor dar aula no horário requerido. Nesse sentido, esse estudo concluiu com êxito o seu objetivo de desenvolver um algoritmo para resolver o problema de logística existente no DCC ao elaborar a oferta disciplinar semestral. A solução estará à disposição para o departamento, caso o mesmo deseje usar-lá.

References

[1] A. Blum. New approximation algorithms for graph coloring. *Journal of the ACM (JACM)*, 41(3):470–516, 1994.
[2] M. R. Garey. Computers and intractability; a guide to the theory of np-completeness. Technical report, 1979.
[3] M. M. Halldórsson. A still better performance guarantee for approximate graph coloring. *Information Processing Letters*, 45(1):19–23, 1993.
[4] D. JOHNSON. Worst case behavior of graph coloring algorithms. In *Proceedings of the 5th Southeast Conference on Combinatorics, Graph Theory, and Computing, 1974*, pages 513–527, 1974.
[5] D. Karger, R. Motwani, and M. Sudan. Approximate graph coloring by semidefinite programming. *Journal of the ACM (JACM)*, 45(2):246–265, 1998.
[6] R. Lewis. *A guide to graph colouring*, volume 7. Springer, 2015.
[7] A. Wigderson. Improving the performance guarantee for approximate graph coloring. *Journal of the ACM (JACM)*, 30(4):729–735, 1983.