Instituto Federal da Paraíba
Campus Campina Grande.
Curso: Bacharelado em Engenharia de Computação
Disciplina: Análise e Técnicas de Algoritmos
Professora: Ruan Delgado
Discentes: Rafael Guimarães
Ramon Lino

Relatório de Projeto Final

Algoritmo de alocação de professores para disciplinas

Campina Grande
Dezembro 2020

1. Introdução

Nesse relatório encontram-se notas, análise e roteiro de solução do projeto de Alocação otimizada e justa de professores para disciplinas do IFPB. Esse projeto foi proposto pela disciplina *Análise e técnica de algoritmos,* ministrada pelo professor Ruan Delgado Gomes.

2. Descrição do Projeto

O projeto consiste na implementação de um software para alocação de professores a disciplinas. Esse software e os algoritmos utilizados podem ser genéricos com relação a cursos e professores, mas deve-se tentar criar casos de teste relevantes no contexto da Coordenação da Área de Informática (COAIN) do IFPB Campina Grande. A COAIN possui atualmente 5 cursos: Engenharia de Computação, Telemática, Técnico Integrado em Informática, Técnico Subsequente em Informática e Técnico Subsequente em Manutenção e Suporte em Informática. Algumas disciplinas são oferecidas em múltiplos grupos (olhar o HIFPB para obter essa informação). A equipe conta atualmente com 35 professores efetivos e 4 substitutos. Mais informações sobre os cursos e equipe de professores podem ser obtidas em `coain.vhost.ifpb.edu.br`.

O desafio na alocação de professores a disciplinas é conseguir alocar os professores mais capacitados para os assuntos específicos, mas ao mesmo tempo atender a requisitos de equalização de trabalho, bem como requisitos com relação à limitação de carga horária máxima por professores. À princípio, a carga horária (CH) de sala de aula máxima de um professor do IFPB é de 20h. No entanto, devido à necessidade de os professores participarem de outras atividades, como reuniões, trabalhos em comissões, orientações de alunos etc, em geral busca-se alocar no máximo 16h ou 18h por professor. Professores que possuem cargos de gestão (ex: coordenador de curso, coordenador de estágio, coordenador de monitorias etc) possuem CH máxima de 12h. Professores em cargos de direção (ex: diretor de ensino ou diretor geral) usualmente não lecionam disciplinas, pois dedicam 100% do tempo ao trabalho de gestão. No entanto, como esses valores podem variar dependendo do contexto, seria interessante ter flexibilidade para configurá-los na hora de executar o software.

Com relação à justiça na alocação de carga de trabalho aos professores, não pode-se apenas considerar a CH semanal de aulas, uma vez que existem disciplinas com níveis de dificuldade diferentes (exigem mais horas de trabalho de preparação), bem como turmas com muito mais alunos que outras (que exigem mais horas de trabalho de acompanhamento e correções de atividades e provas). Dessa forma, o ideal seria que a métrica para determinar a carga de trabalho de um professor levasse em consideração todos esses fatores e o algoritmo conseguisse equalizar a carga de trabalho com base nessa métrica mais holística.

Como parte do projeto, vocês devem definir as métricas e formalizar a função objetivo e as restrições deste problema de otimização, que tem como objetivo maximizar o nível de aptidão médio dos professores alocados às disciplinas, mas ao mesmo tempo atender às restrições de CH impostas e também tentar garantir o máximo de justiça possível na alocação de carga de trabalho. Além disso, seria interessante que o software fosse o mais parametrizável possível, para que pudesse ser usado em diferentes contextos e também que permita ao usuário avaliar diferentes possibilidades. Finalmente, os algoritmos utilizados devem possuir complexidade computacional adequada para executar rapidamente, mesmo considerando entradas que contenham centenas de professores e até milhares de disciplinas. Como parte do projeto, além da descrição dos algoritmos utilizados, deve-se também demonstrar a complexidade deles. Como sugestão, aconselho documentar todas as ideias exploradas, de modo a fundamentar a escolha dos algoritmos da versão final. Quanto mais informações úteis forem geradas sobre o processo de desenvolvimento da solução, melhor.

3. Overview de Requisitos

De acordo com a descrição do projeto podemos destacar alguns pontos que devem estar presentes na lógica do desenvolvimento do algoritmo.

I. Alocar os professores mais capacitados para os assuntos específicos
II. Atender a requisitos de equalização de trabalho
III. Atender requisitos com relação à limitação de carga horária máxima por professor.
IV. Coordenadores e Diretores possuem carga horária diferenciada.
V. Na alocação, não devemos considerar apenas a carga horária como fator de justica.
VI. Definir as métricas e formalizar a função objetivo

4. Desenvolvimento

4.1 Descrição entrada

Para satisfazer as restrições impostas, pela descrição do problema, o algoritmo recebe como entrada, os seguintes parâmetros:

- **D** Quantidade de disciplinas
- **P** Quantidade de professores
- **CHM** Carga horária máxima dos professores
- **CHC** Carga horária da Coordenação
- **CHG** Carga horária dos cargos de Gestão

Dessa forma a carga horária ficará parametrizável.

Em seguida serão disponibilizados **D** disciplinas da seguinte forma:

Primeira Linha: o teremos o **nome** Disciplina**;**
Segunda Linha: teremos **C, A, IR** separados por um espaço em branco

Sendo:

- **C** Quantidade de **Créditos** ou **Carga Horária**
- **A** Quantidade de **Alunos**
- **IR** Índice de Reprovação

Em seguida serão recebidas **P** linhas com contendo **D** inteiros, O professor $\mathbf{P_i}$ será identificado até o final da execução do programa pelo índice **i.** Cada inteiro $\mathbf{d_i}$ representa a aptidão do professor em uma disciplina **i.**

Em seguida o programa recebe como entrada **NC** com o número de professores coordenadores e em seguida **NC** linhas com o número de cada professor. (1 <= $\mathbf{NC_i}$ <= P) para definir o professor como coordenador.

Depois disso, temos **ND** com o número de professores em cargo de direção e em seguida **NC** linhas com o número de cada professor. (1 <= $\mathbf{ND_i}$ <= P) para definir o professor como coordenador.

4.2 Tabela Match

A aptidão do professor para cada disciplina pode variar de 1 a 5, e possui a seguinte classificação.

1. Inapto
2. Pouco Apto
3. Médio Apto
4. Muito Apto
5. Especialista

Com base nessa classificação nosso algoritmo cria inicialmente uma lista de adjacência, com professores com aptidão maior que > 3 com objetivo de reduzir o espaço de busca e consequentemente a complexidade em alguns pontos da solução proposta.

4.3 Estimando nível de dificuldade da disciplina

Um dos maiores desafios do projeto é a estimativa do quão trabalhosa se torna a disciplina para um professor, uma vez que vários fatores podem tornar uma disciplina mais trabalhosa do que outras e não só isso, como também o trabalho de uma disciplina pode distinguir de outra em diferentes parâmetros. Tomando como exemplo duas disciplinas: Programação Orientada a Objeto (POO) e Introdução a Engenharia de computação (IEC), temos duas disciplinas com a quantidade de aluno relativamente alta, no entanto para POO há uma dificuldade maior da parte dos alunos para assimilar o conteúdo da disciplina, o que não demanda em IEC, com isso é necessário que o professor tenha mais empenho e atenção aos alunos para suprir essa necessidade que geralmente se tem da parte dos alunos, então, POO, de maneira implícita, requer mais trabalho do professor do que IEC requer.

Diante da problemática achamos melhor definir uma métrica geral para cada disciplina que define o quão trabalhosa a disciplina tende a ser para o professor. Essa métrica está definida pelo atributo cargaDeTrabalho - **CT** na classe **Disciplina.**

- cargaDeTrabalho (**CT**) - Esse atributo é composto por outros atributos que já compõem a disciplina, como: Créditos (**C**), Quantidade de Alunos (**NA**) e Índice de Reprovação (**IR**). Para cada atributo que compõe a carga de trabalho existe um peso, que pode ser parametrizável em implementações posteriores, para nossa implementação está definido em 1/3 o peso para cada atributo escolhido (**C**, **NA** e **IR**), então para definir a carga de trabalho é feito uma média ponderada:
  - (**C** * 33,3 + **NA** *33,3 + **IR** * 33,3) / 100 = cargaDeTrabalho.
- Para cada peso, alternativamente poderia definir o grau de importância que um atributo teria no quanto uma disciplina gera trabalho ao professor, seria necessário apenas

aumentar ou diminuir o peso que cada atributo tem na média final. Outros atributos também poderiam ser adicionados para definir a carga de trabalho da disciplina.

4.4 Definindo quais disciplinas serão alocadas inicialmente.

Para começar, as primeiras disciplinas receberão uma estimativa do número de professores aptos para ministrá-la. Com base nesses dados, as disciplinas serão ranqueadas de modo que as primeiras disciplinas a serem alocadas, serão aquelas que têm a menor quantidade de professores aptos. Essa operação é linear na quantidade de disciplinas, aproveitando os dados da **Tabela Match.**

4.5 Ranqueamento de Professores por disciplina

Nesse ponto do algoritmo será feito um ranqueamento dos professores com base em três parâmetros em ordem decrescente de prioridade.

1. Aptidão (maior)
2. Carga Horária Disponível (maior)
3. Carga de Trabalho (menor)

Desse modo, temos que a aptidão será o primeiro parâmetro de escolha, em seguida a carga horária disponível em terceiro lugar teremos a carga de trabalho (com prioridade inversa) ou seja, o professor com menor carga de trabalho será escolhido.

5 Pseudocódigo Algoritmo de Alocação

Nosso algoritmo de alocação de escolhas consiste no uso das estruturas de dados fila de prioridade para fazer o ranqueamento do professor com base nos parâmetros citados anteriormente de maneira, que cada ranqueamento será realizado com o estado do professor sendo alterado.

Com isso tempos que: se um professor é alocado para uma disciplina, sua carga horária é de crescida a carga de trabalho é incrementada, ambos conforme disciplina.

Vejamos esse exemplo:

```
Administração de Serviços
Prof:   11      CH: 20      A:  2       CT: 0.00
Prof:   3       CH: 16      A:  2       CT: 11.22
Prof:   8       CH: 16      A:  2       CT: 12.54
Prof:   9       CH: 20      A:  1       CT: 0.00
Prof:   10      CH: 20      A:  1       CT: 0.00
Prof:   7       CH: 20      A:  1       CT: 0.00
Prof:   4       CH: 20      A:  1       CT: 0.00
Prof:   5       CH: 20      A:  1       CT: 0.00
Prof:   2       CH: 20      A:  1       CT: 0.00
Prof:   0       CH: 16      A:  1       CT: 14.52
Prof:   6       CH: 12      A:  1       CT: 20.00
```

Fig. 1: Ranqueamento de Disciplinas

Os três professores no topo do Ranking possuem a mesma aptidão, no entanto nesse momento da execução do programa, o professor 11 era o único que estava com carga horária disponível de 20 horas, portanto para essa disciplina ele será o escolhido.

Veja também que os outros dois professores, em segundo e terceiro lugar “empatam” em carga horária, ou seja, os dois possuem a mesma carga horária disponível, no entanto a carga de trabalho do professor 3 é menor que a do professor 8 e por isso ele ficou como segunda opção para aquela disciplina em questão. Abaixo disponibilizamos o pseudocódigo do algoritmo de alocação.

```
Aloca-Disciplinas():

  PriorityQueue disciplinas;
  // Ordenar disciplinas quantidade de prof. aptos crescente.

  List<Priority Queue<Professores>> professoresQueue;
  // ordena os professores, de acordo com os parâmetros citados em 4.5

  Array professores;
  // isso é um atributo de classe usado para controle de estados da classe.


  while ( disciplinas != vazio )  ):

      // extrair disciplina
      disciplina ⇇ disciplina.top();

      if (tableMatch[disciplina.index].size() == 0) {

            for (each professor in professores) {
                  if (professor.getAptidaao() == 2)
                        inserir professor em professoresQueue;
            }

      }

      else {

            for (todos professors in tableMatch[disciplina.index]) {
              inserir professor em professoresQueue.
            }

      }
```

```
        // Extrair o melhor professor do ranking.
        if ( ! professoresQueue.empty() )
              p ⇐ professoresQueue.top();

              // atualizar estado do professor.
              (...)

              // atualizar tabela de alocação
              (...)


        else {

              Array disciplinaSemProfessores.add( disciplina );

        }




        delete professoresQueue;

}
```

6 Análise do Algoritmo

De modo objetivo podemos inferir que inicialmente nós temos uma ordenação das disciplinas por quantidade de professores aptos usando o método de ordenação HeapSort com complexidade $O(d \bullet log(d))$ sendo d a quantidade de disciplinas.

Em seguida temos que, para cada disciplina retirada da fila vamos ranquear professores que estão na tabela match e com carga horária disponível. No pior caso temos que essa operação será $O(p \bullet log(p))$ mas muito provavelmente bem menor que p.

Caso não tenha professores na tabela match será feita uma busca linear na quantidade de professores montando uma fila de prioridade de professores com a aptidão == 2. $O(p + p \bullet log(p))$

Na média, o algoritmo de alocação executa em $O(d \bullet p \bullet log(p))$

7 Resultados

Alguns casos de teste, no primeiro caso de teste não tão grande, fizemos 12 disciplinas sendo alocadas para 4 professores e tivemos o seguinte resultado:

Entrada 1

```
13 4
20 8 20
IEC 2 30 5
AEP 4 20 20
LAEP 4 25 20
SDI 4 40 15
MEE 2 40 5
ED 4 35 25
LED 4 25 15
SDII 4 25 20
TDG 4 20 25
POO 4 30 30
LPOO 4 30 10
OAC 4 30 20
IA 4 27 18
1 5 5 3 1 5 5 1 5 4 4 5 3
4 5 5 1 1 4 4 1 5 5 5 1 2
5 2 2 5 5 2 2 5 2 4 4 5 1
5 5 5 1 1 4 4 1 5 5 5 1 5
0
0
```

Saída 1

```
Prof. 0:  CH: 8    CT: 53.46
A: 5               OAC
A: 5               ED
A: 5               LED

Prof. 1:  CH: 8    CT: 51.81
A: 5               TDG
A: 5               AEP
A: 5               POO

Prof. 2:  CH: 10   CT: 51.15
A: 5               MEE
A: 5               SDII
A: 5               SDI

Prof. 3:  CH: 6    CT: 59.07
A: 5               IA
A: 5               LAEP
A: 5               IEC
A: 5               LPOO
```

Podemos observar nesse caso de teste que a grande maioria das disciplinas foram alocadas com aptidão próxima de 5 e não houve uma diferença absurda de carga de trabalho entre os professores.

Entrada 2

Para esse teste, o arquivo está disponível no repositório do projeto em tests/T5.txt

Saída 2

```
        - Results -

Prof. 0: CH: 4   CT: 71.28
A: 3             AdmDeSistemasOperAbertos
A: 3             AnáliseETécDeAlgoritmos
A: 5             PadrõesDeProjeto
A: 5             InformaticaBásica

Prof. 1: CH: 0   CT: 50.00

Prof. 2: CH: 4   CT: 50.49
A: 4             LabDeSistemasAbertos
A: 5             OrientaçãoAObjetosJava
A: 5             EstruturaDeDadosC

Prof. 3: CH: 0   CT: 42.24
A: 5             EstatísticaAplicada
A: 4             ProjetoemTelemática
A: 5             InteligenciaArtificial
A: 5             BancoDeDados
A: 2             AdministraçãoDeServiços

Prof. 4: CH: 10  CT: 33.00
A: 4             MetodologiaDaPesquisaCientifica
A: 5             AlgoritmosI

Prof. 5: CH: 12  CT: 34.32
A: 5             TeoriadaComputação
A: 4             ProjetoEmEngenhariaDeComp.

Prof. 6: CH: 0   CT: 64.88
A: 3             SistemasDigitaisI
A: 3             SistemasDigitaisII
A: 2             TecnologiaDeRedesLocais

Prof. 7: CH: 20  CT: 0.00
```

Prof. 8: CH: 0 CT: 52.14
A: 3 TeoriaDosGrafos
A: 5 ProgramaçãoIII(Concorrente,Redes,Web)
A: 5 DesenvolvimentoWeb
A: 4 AnáliseEProjetoDeSistemas
A: 2 MétodosNuméricos

Prof. 9: CH: 4 CT: 38.28
A: 3 SistemasOperacionais
A: 3 RedesDeComputadores
A: 4 GerênciaDeProjetos
A: 5 TestesDeSoftware

Prof. 10: CH: 8 CT: 36.96
A: 3 AdmDeSistemasOperProprietários
A: 2 AdministraçãoDeSistemas
A: 2 LabSistemasOperacionais

Prof. 11: CH: 4 CT: 57.75
A: 3 ArquiteturaDeComputadores
A: 4 OrientaçãoAObjetosC++
A: 5 EstruturaDeDadosPython

Foi possível adquirir nesse caso de teste aptidão média alta e também médio percentual de justiça em termos de carga de trabalho.

Conclusão

Durante a execução do projeto foram realizados os mais diversos testes tentamos criar um algoritmo que fazia todas as permutações, testando as possibilidades, Nessa primeira tentativa o tempo de execução explodiu (e o caso de teste não chegava nem perto da realidade de uma Instituição de Ensino Federal) Em seguida pensamos em uma estratégia que embora não verifica todas as possibilidades, na média, busca o melhor resultado, considerando existem 3 parâmetros de ranqueamento. Dessa forma conseguimos também atender um requisito muito importante que é o tempo de execução do algoritmo.

De maneira geral, a abordagem gulosa para a resolução do problema atendeu de maneira **satisfatória,** priorizando sempre alocar os professores mais aptos para ministrar a disciplina e distribuindo de maneira uniforme a carga de trabalho para cada professor, considerando a ideia inicial de uma definição da carga de trabalho (**CT**) para cada disciplina. Mas vale destacar que essa estratégia possui limitações como por exemplo, disponibilizar diferentes resultados para o mesmo conjunto de disciplinas e professores, uma vez que a estratégia busca alocar de maneira eficiente e ótima os professores e para casos onde não há professores aptos, nosso arranjo na implementação acaba alocando professores com aptidão não tão ideal. Concluímos então que a estratégia, mesmo com suas limitações, atende de maneira satisfatória a justiça na distribuição das disciplinas para os professores.