Universidade Federal de Sergipe

Disciplina: Inteligência Artificial (IA)
Turma: 01
Professor: Carlos Alberto Estombelo Montesco

Aluno: Everton Santos de Andrade Júnior
Matrícula: 202100011379
Grupo:02

Trabalho Final de IA - Turma CSP

Contexto do Problema de Turma CSP

Neste trabalho formalizamos um problema de satisfação de restrições (CSP) com respeito a escolha de disciplinas em um dado período de uma universidade.

Considere um estudante de Ciência da Computação (CC) da Universidade Federal de Sergipe (UFS).
Esse estudante não faz estágio, mas ele faz iniciação científica (PIBIC), isso limita as horas diárias que ele tem para participar das aulas, e ainda estudar fora de sala.

Para esse aluno o único turno que ele realmente pode frequentar a universidade é o turno vespertino, portanto a escolha de disciplinas desse aluno deve considerar apenas as ofertas de turmas no horário da tarde. Além disso, o aluno deseja cursar pelo menos duas matérias que são do *Perfil Básico*, que seriam as matérias não opcionais. E ainda deseja cursar pelo menos 1 matéria do Departamento de Matemática (DMA), já que o aluno pretende aprender mais matemática.

Um dia tem 24 horas, esse aluno precisa de 10 horas para dormir e se alimentar no dia. Sobram 14 horas, mas como ele faz PIBIC, que são 20 horas semanais,dividindo essas horas semanais em 2h diárias de dedicação ao PIBIC (sabemos que 2h por dia é igual a 17 horas na semana, mas esse aluno considera isso suficiente, e promete dar atenção maior ao PIBIC nas férias). Assim sobram 12 horas diárias, mas considerando viagem de ida e volta, realisticamente, são 10 horas diárias.

Então temos 10 horas diárias para distribuir entre as possíveis turmas escolhidas por esse aluno. Considerando de segunda-feira a sexta-feira, em média a cada 15 horas de carga horária equivale a 1 hora de aula presencial na semana.

O aluno afirma que precisa de um tempo de estudo a mais para cada matéria, além das aulas na UFS, então para ser justo ele decidiu calcular a partir da carga horária que pegar. O aluno achou justo 1 hora de estudo extra semanal para cada 15 horas de carga horária.

Então, a cada 15 horas de carga horária, o aluno precisa, ao total, de 2 horas toda semana. Lembre-se que o aluno possui tempo livre $l = 10 \cdot 5$ horas por semana (considerando seg. à sex.) e horas semanais de estudo (incluindo participação presencial e estudo próprio) por carga horária, dado por, $\frac{h}{c_h}$.

Portanto, a carga horária máxima c_{hmax} que esse aluno deve pegar é dada pela equação:

$$c_{hmax} = \frac{c_h}{h} l \tag{1}$$

Nesse caso $c_h = 15, \quad h = 1.5, \quad l = 10 \cdot 5$ portanto:

$$c_{hmax} = \frac{15}{2} 50 = 375$$

O aluno também acha ineficiente cursar muitas matérias ao mesmo tempo, pois não consegue trocar de contexto tão rápido, então ele pretende cursar no máximo 7 disciplinas para o próximo período (2022.2).

Definição do Problema em Variáveis, Domínios e Restrições

Dessa maneira, o Problema de Satisfação de Restrições fica definido da maneira abaixo, considerando as variáveis V, conjunto dos domínios D e restrições C

$V = \{E_1, ..., E_7\}$ onde cada $E_i : i = 1, 2, .., 7$ representa uma turma a ser **escolhida**.

$D = \{D_1, ..., D_7\}$

$C = \{C_1, ...C_k, C_{sum}, C_{diff(1)}, ..., C_{diff(7)}, C_{req(1)}, ..., C_{req(7)}, C_{DMA}, C_{perfil}\}$

Todas essas restrições serão definidas mais abaixo neste texto.

$n =$ nº de turmas ofertadas

$D_i = \{null, T_1, T_2, ..., T_n\}$ é conjunto de todas as turmas ofertadas neste período, exemplo de turma $T =$ COMP045 no horário 24T12 (Seg. e Qua. no primeiro e segundo horário da tarde) com professor Alberto. Consideramos $null$ um turma especial que representa que nenhuma turma foi atribuída a uma escolha, e a escolha null é uma escolha válida.

Mas todas as escolhas $null$ não seriam uma opção de solução? Não, porque temos a garantia de uma quantidade mínima e máxima de carga horária definida no Projeto Pedagógico do Curso (PPC) que exige uma carga horária mínima do aluno em um dado período, o que garante que vamos realmente escolher, ao menos, algumas disciplinas (se a oferta permitir). Segue o trecho do artigo que se encontra no PPC:

§2º O aluno poderá cursar uma carga horária mínima de 240 (duzentas e quarenta) horas e uma carga horária máxima de 480 (quatrocentas e oitenta) horas por semestre.

Vamos combinar essa a restrição do PPC de CC junto com carga horária máxima que do aluno da seguinte maneira:

Seja $c'_{hmax} = max(480, c_{hmax})$ lembrando que c_{hmax} é o máximo permitido pelo aluno e 480 o máximo permitido pelo PPC.

Seja $c_{hmin} = 240$ e $sum : \mathbb{P}(V) \to \mathbb{R}$ uma função que soma a carga horária de cada um dos elementos de um subconjunto de variáveis V e retorna esse somátorio. ($\mathbb{P}(V)$ é o *powerset* de V)
Observação: A turma $null$, possui carga horária de 0 horas.

Portanto a restrição n-ária C_{sum} é definida da seguinte maneira:

$C_{sum} = \langle (E_1, ..., E_7), sum(\{E_1, ..., E_7\}) \geq c_{hmin} \wedge sum(\{E_1, ..., E_7\}) \leq c'_{hmax} \rangle$

Dado um relação $conflita : V \times V \rightarrow \{true, false\}$, que pode ser encarado como uma função que pega a atribuição de duas variáveis e retorna $false$ ou $true$. Onde $false$ representa que duas turmas escolhidas não conflitam o horário e não possuem a mesma disciplina, já $true$ indica que conflitam ou possuem a mesma disciplina.

Exemplo 1: $conflita(E_1 = T_1, E_2 = T_1) = 1$, pois as turmas são exatamente a mesma e portanto tem horários conflitantes.

Exemplo 2: $conflita(E_1 = null, E_2 = T_1) = 0$ pois um escolha de vazia nunca irá conflitar com qualquer turma, portanto não há choque de horários.

Com essa relação em mãos podemos definir as k restrições de conflito dado abaixo (k depende de quantas turmas, no máximo, pode-se escolher) :

$p = 1, 2, ..., k$

$C_p = \langle (E_i, E_j), \quad conflita(T_i, T_j) = false \rangle$ considerando $i \neq j$

Consideramos que o aluno já cursou o primeiro e segundo período de CC, segue o código das matérias:

MAT0151, MAT0150, MAT0057, COMP0480, COMP0393, que são do 1º período,

MAT0152, MAT0078, FISI0260, FISI0264, COMP0410, COMP0334, que são do 2º período.

Visto que, existem matérias que possuem pré-requisitos, então tome β como o conjunto de todas as matérias cursadas por esse aluno, que é constante durante o problema. Além da a função $satisfaz_\beta : V \rightarrow \{true, false\}$ que retorna $true$ se a variável atribuída tem seus pré-requisitos satisfeitos pelo conjunto β.

Assim, faremos as seguintes restrições unária $C_{req(i)}$ para garantir pré-requisitos consistentes e $C_{diff(i)}$ para garantir que a matéria não é repetida, ou seja, que o aluno já cursou essa matéria.

$C_{diff(i)} = \langle (E_i), \quad E_i.disciplina \notin \beta \rangle$ Considere que $E.disciplina$ é a disciplina de determinada turma atribuída.

$C_{req(i)} = \langle (E_i), satisfaz_\beta(E_i) = true \rangle$

E também definimos as restrições globais de incluir matéria do DMA, e o perfil de pelo menos 2 serem básico, considere a função, $sum\_perfil\_básico$ que recebe um conjunto de Variáveis e retorna o número de turmas que possuem perfil básico.

$C_{perfil} = \big\langle (E_1, ..., E_7), sum\_perfil\_básico(\{E_1, ..., E_7\}) \geq 2 \big\rangle$

Seja β_{DMA} o conjunto de todas as turmas ofertadas pelo DMA, então podemos definir a restrição de DMA como:

$C_{DMA} = \langle (E_1, ..., E_7), \exists i : E_i \in \beta_{DMA} \rangle$ para $i = 1, 2, ..., 7$ Ou seja, existe, entre as escolhas do aluno, alguma turma que é ofertada pelo DMA.

Codificação

Nesta sessão mostro alguns trechos da codificação das restrições e partes alteráveis desse trabalho e no final um recorte do *output* dos resultados encontrados.

Primeiramente, como mostrado no livro AIMA, uma restrição unitária pode ser satisfeita ao restringir o domínio das variáveis, alcançando a consistência. Então no código, fazemos uma filtragem para

retirar as mateiras iguais as matérias *cursadas* pelo aluno, além de, também, remover as matérias que o aluno não possui pré-requisito para cursar, da maneira abaixo:

```java
ArrayList<Turma> turmas_restringidas = new ArrayList<Turma>();

for (int i = 0; i < all_turmas.length; i++) {
  boolean disciplina_permitida = true;
  for (int j = 0; j < cursadas.length; j++) {
    Turma t =  all_turmas[i];
    Disciplina d  = t.getDisciplina();
    // Checando se a turma é de alguma disciplina já cursada pelo discente
    boolean iguais = d.equals(cursadas[j]);
    boolean satisfaz_prerequisito = d.satisfazPreReq(cursadas);
    // Se as turmas são iguais a que ele cursou
    // Ou ele não satisfaz o prerequisito então
    // a disciplinas nao é permitida
    if (iguais || ! satisfaz_prerequisito) {
      disciplina_permitida = false;
      break;
    }
  }

  if(disciplina_permitida) {
    turmas_restringidas.add(all_turmas[i]);
  }
}
```

Segue abaixo a maneira foi codificado as restrições para evitar conflito de horários, seguindo fielmente as restrições definidas nesse trabalho.

```java
for (int i = 0; i < variables.length; i++) {
  for (int j = 0; j < variables.length; j++) {
    if (i == j)
      continue;

    restrictions.add(
      new BasicConstraint(
        new String[]{variables[i], variables[j]},
        vals -> {
          Turma t1 = (Turma) vals[0];
          Turma t2 = (Turma) vals[1];

          boolean conflict = t1.conflita(t2);
          // Só permitimos se NÃO der conflito
          return conflict == false;
        }
      )
    );
  }
}
```

Para garantir que a quantidade carga horária pego por esse aluno respeita a restrição de carga horária máxima dele, e ao mesmo tempo respeite as horas mínimas por período informado pelo PPC, temos a formulação da restrição C_{sum} em forma de código, a seguir:

```java
Constraint min_max_carga_horario = new BasicConstraint(
  variables,
  (vals) -> {
    int sum = 0;

    for (Object val : vals) {
      sum += ((Turma) val).getDisciplina().getCargaHoraria();
    }

    if (sum ≥ ch_min && sum ≤ ch_max){
      return true;
    }
    else{
      return false;
    }

  }
);
```

Para satisfazer a requisição do aluno de possuir pelo menos uma matéria do DMA:

```
Constraint dma_constraint = new BasicConstraint(
  variables,
  (vals) -> {

    for (Object val : vals) {
      boolean dma_exist = ((Turma) val).getDisciplina().getDepartamentoCode().equals("DMA");
      if (dma_exist){
        return true;
      }
    }
    return false;
  }
);
```

Além da restrição de duas matérias do perfil básico:

```
Constraint duas_perfil_basico = new BasicConstraint(
  variables,
  (vals) -> {
    int basico = 0;
    for (Object val : vals) {
      boolean is_basico = ((Turma) val).getDisciplina().perfil == Disciplina.Perfil.Basico;
      if (is_basico){
        basico += 1;
      }
    }
    if (basico > 2) {
      return true;
    }
    else {
      return false;
    }
  }
);
```

Escolha do Algoritmo

Como TurmaCSP como definido neste texto é um problema que possui representação de estado fatorada (estruturada) e que existem várias soluções e poucas restrições binárias, foi decidido que **Backtracking** seria um algoritmo adequado e suficiente.

Backtracking é finalizado assim que encontrar uma atribuição consistente e completa que atenda todas as restrições, pois o Backtracking aproveita a estrutura dos estados definidos dessa maneira, acaba economizando em memória e não perde tempo visitando estados que fere alguma restrição, e, também, não depende de heurística específica para o problema. Já o algoritmo de busca generalizada, com ou sem informação, não toma proveito da representação estruturada dos estados, mas o Backtracking sim.

Além disso, o Backtracking é eficiente em memória, pois mantém apenas uma representação de atribuição das variáveis e vai passando essa representação às suas chamadas recursivas e à medida que alguma atribuição retorna como falha, essa mesma representação é modificada para um estado consistente novamente. Então não há cópias dessa representação.

Também, considerando que o problema possui várias configurações válidas logo de cara, certos pré-processamentos não restringem o espaço de busca. Por exemplo, o AC3 busca consistência de arco então apenas considera apenas restrições binárias, então das restrições binárias, uma opção de turma T_j só se torna inviável à medida que escolhemos alguma outra turma T_i para sequer haver a possibilidade de conflito. Mas no começo ainda não escolhemos nenhuma turma, portanto não reduz o número de valores possíveis inicialmente.

E usar o AC3 ou outra heurística durante o backtracking não iria reduzir o tempo computacional, pois, o tempo usado para testar esses algoritmos, para esse problema, é o basicamente o mesmo tempo que o Backtracking leva para conferir as restrições. Testes foram feitos, e a implementação mais simples, às vezes, é adequada, como foi nesse nosso problema.

Outro motivo que o backtracking sozinho é suficiente é a presença de algumas restrições globais, tal que usando k-consistência também acabaria tendo complexidade computacional parecida com o backtracking ao testar as restrições definidas, já que o problema de CSP, ainda, é NP-Completo.

A estrutura desse problema não é de árvore, e não está separado em componentes, os arcos conectam todos os nós, e, portanto, usar um TreeCSP solver ou tentar encontrar uma atribuição em componentes separados também não é viável.

Mas vale informar, que um algoritmo ***Local Search***, também é apropriado. Ou seja, poderíamos começar com uma atribuição de turmas para cada variável $E_1, \ldots, E_7$, provavelmente essa atribuição inicial possui diversas restrições violadas, então procurando localmente variações que resultam em uma solução, como existem muitas soluções em nesse problema TurmaCSP, **rapidamente** encontraria uma solução. Porém, caso aumentássemos muitos as restrições, e cada vez mais adicionarmos preferências para esse código, a chance de começarem uma atribuição local viável que possa modificar localmente e encontrar uma solução se torna cada vez menor, para evitar isso preferimos continuar com o Backtracking.

Codificação - Valores ajustáveis

Valores que podem ser mudados no código são as matérias que ele cursa, localizado no arquivo TurmaCSP.java, se ele faz PIBIC, Estágio, ou PIBITI, pode ser alterado lá também, cada atividade retira horas semanais do saldo de horas desse aluno.

```
Disciplina[] disciplinas_cursadas = new Disciplina[]{
  // Periodo 1
  Disciplina.disciplinaFromCode("MAT0151"),
  Disciplina.disciplinaFromCode("MAT0150"),
  Disciplina.disciplinaFromCode("MAT0057"),
  Disciplina.disciplinaFromCode("COMP0480"),
  Disciplina.disciplinaFromCode("COMP0393"),
  // Periodo 2
  Disciplina.disciplinaFromCode("MAT0152"),
  Disciplina.disciplinaFromCode("MAT0078"),
  Disciplina.disciplinaFromCode("FISI0260"),
  Disciplina.disciplinaFromCode("FISI0264"),
  Disciplina.disciplinaFromCode("COMP0410"),
  Disciplina.disciplinaFromCode("COMP0334"),
};

boolean
  PIBIC   = true,
  ESTAGIO = false,
  PIBITI  = false;

Estudante e = new Estudante(
  disciplinas_cursadas,
  Horario.Turno.Vespestino,
  PIBIC,PIBITI,ESTAGIO
);
```

As horas de aula presenciais não podem ser alteradas, pois não se pode mudar como a UFS estrutura essas aulas, mas pode mudar quantas horas extra semanal por carga horária, que no nosso exemplo foram 1 hora extra semanais a cada 15 horas de carga horária. Possível alterar a ida e volta (para PIBIC ou para a UFS digamos). Nessas linhas de código:

```
   void
); ufs.cc.Estudante.setHorasSemanaisExtraPorCargaHoraria(flo
   horas_extra, float carga_horaria)
//                                                    ria
e.setHorasSemanaisExtraPorCargaHoraria(1.f,15.0f);
e.setHorasDeViagemIda(1.0f);
e.setHorasDeViagemVolta(1.0f);
```

Também podem ser alterados a quantidade de turma ofertadas, tanto quanto a quantidade de turmas máximo que esse aluno se permite pegar:

```
public static void run() {
  final int N_TURMAS_PARA_ESCOLHA = 7;
  final int N_TURMAS_ORFERTADAS   = 45;
```

Se quiser que o output do programa saia em um arquivo chamado *output-file.txt*

Mude para $true$ essa a variável chamada *SEND_TO_OUTPUT_FILE* no arquivo ./ufs/Main.java assim:

```
public class Main {
  // Deixe a variavel abaxio como 'true' para mandar printar em arquivo txt no lugar do console.
  static final boolean SEND_TO_OUTPUT_FILE = true;
```

Output do trabalho

Imagem chamda *output.png* do console em format png disponível no envio deste trabalho

- ---------- TurmaCSP ----------

> Aluno:
> PIBIC:true ESTAGIO:false PIBITI:false
> Horas Diarias Disponiveis para Estudar = 10,000000
> Horas de Estudo Diario (incluso horario de aula + estudo fora de aula) por
> Carga Horária = 0,133333
> <<

CARGA_HORARIO_MAXIMA_DO_ALUNO = 375
CARGA_HORARIO_MAXIMA_PPC = 480

Turmas Ofertadas:
2022.2 FISI0264 60 Laboratório de Física1*
Seg 13:00-14:50 Qua 13:00-14:50 Estombelo

2022.2 COMP0424 60 Aprendizagem de Máquina
Seg 15:00-16:50 Qua 15:00-16:50 Leonardo Nogueira

2022.2 COMP0480 30 Seminários em Computação
Ter 13:00-14:50 Beatriz Trinchao

2022.2 COMP0308 120 Atividades Complementares
Seg 17:00-18:50 Ter 17:00-18:50 Qua 17:00-18:50 Qui 17:00-18:50 Leonardo Nogueira

2022.2 COMP0438 60 Engenharia de Software I

Ter 15:00-16:50 Qui 15:00-16:50 Rene Pereira

2022.2 COMP0424 60 Aprendizagem de Máquina
Ter 17:00-18:50 Qui 17:00-18:50 Rafael

2022.2 COMP0334 60 Programação Imperativa**
Sex 15:00-16:50 Sex 17:00-18:50 Leonardo Nogueira

2022.2 COMP0455 60 Banco de Dados I
Ter 15:00-16:50 Qui 15:00-16:50 Beatriz Trinchao

2022.2 COMP0334 60 Programação Imperativa**
Sex 15:00-16:50 Sex 17:00-18:50 Estombelo

2022.2 COMP0472 60 Sistemas Operacionais
Sex 15:00-16:50 Sex 17:00-18:50 Tarcisio Rocha

2022.2 MAT0153 60 Cálculo C
Seg 13:00-14:50 Qua 13:00-14:50 Evilson

2022.2 COMP0472 60 Sistemas Operacionais
Ter 17:00-18:50 Qui 17:00-18:50 Rene Pereira

2022.2 COMP0485 120 Trabalho de Conclusão de Curso I
Seg 13:00-14:50 Ter 13:00-14:50 Qua 13:00-14:50 Qui 13:00-14:50 Alberto Costa

2022.2 COMP0484 120 Prática Orientada em Computação II *
Seg 13:00-14:50 Ter 13:00-14:50 Qua 13:00-14:50 Qui 13:00-14:50 Beatriz Trinchao

2022.2 COMP0417 30 Fundamentos de Sistemas Embarcados
Sex 15:00-16:50 Rafael

2022.2 COMP0485 120 Trabalho de Conclusão de Curso I
Seg 13:00-14:50 Ter 13:00-14:50 Qua 13:00-14:50 Qui 13:00-14:50 Rene Pereira

2022.2 COMP0439 60 Engenharia de Software II
Ter 13:00-14:50 Qui 13:00-14:50 Alberto Costa

2022.2 COMP0443 60 Interface Humano Computador
Ter 13:00-14:50 Qui 13:00-14:50 Rafael

2022.2 COMP0409 60 Linguagens Formais e Computabilidade
Ter 13:00-14:50 Qui 13:00-14:50 Alberto Costa

2022.2 COMP0427 60 Inteligência Artificial
Ter 17:00-18:50 Qui 17:00-18:50 Rene Pereira

2022.2 COMP0432 60 Processamento de Imagens
Sex 15:00-16:50 Sex 17:00-18:50 Rafael

2022.2 COMP0398 60 Programação para Web**
Sex 15:00-16:50 Sex 17:00-18:50 Alberto Costa

2022.2 COMP0397 60 Programação Paralela e Concorrente**
Sex 15:00-16:50 Sex 17:00-18:50 Evilson

2022.2 COMP0417 30 Fundamentos de Sistemas Embarcados
Qua 17:00-18:50 Rene Pereira

2022.2 COMP0405 60 Estruturas de Dados**
Sex 15:00-16:50 Sex 17:00-18:50 Rene Pereira

2022.2 FISI0260 60 Física 1
Ter 15:00-16:50 Qui 15:00-16:50 Bruno Otavio

2022.2 COMP0419 60 Prática em Sistemas Digitais*
Sex 15:00-16:50 Sex 17:00-18:50 Rafael

2022.2 COMP0455 60 Banco de Dados I
Sex 15:00-16:50 Sex 17:00-18:50 Bruno Otavio

2022.2 COMP0484 120 Prática Orientada em Computação II *
Seg 15:00-16:50 Ter 15:00-16:50 Qua 15:00-16:50 Qui 15:00-16:50 Leonardo Nogueira

2022.2 COMP0308 120 Atividades Complementares
Seg 15:00-16:50 Ter 15:00-16:50 Qua 15:00-16:50 Qui 15:00-16:50 Rene Pereira

2022.2 COMP0483 180 Prática Orientada em Computação I*
Ter 17:00-18:50 Qua 17:00-18:50 Leonardo Nogueira

2022.2 COMP0419 60 Prática em Sistemas Digitais*

Ter 13:00-14:50 Qui 13:00-14:50 Rene Pereira

2022.2 COMP0455 60 Banco de Dados I
Seg 13:00-14:50 Qua 13:00-14:50 Bruno Otavio

2022.2 FISI0260 60 Física 1
Seg 13:00-14:50 Qua 13:00-14:50 Tarcisio Rocha

2022.2 COMP0308 120 Atividades Complementares
Seg 15:00-16:50 Ter 15:00-16:50 Qua 15:00-16:50 Qui 15:00-16:50 Tarcisio Rocha

2022.2 COMP0463 60 Laboratório de Redes de Computadores*
Ter 17:00-18:50 Qui 17:00-18:50 Evilson

2022.2 MAT0153 60 Cálculo C
Seg 13:00-14:50 Qua 13:00-14:50 Rafael

2022.2 COMP0393 60 Programação Funcional**
Seg 13:00-14:50 Qua 13:00-14:50 Bruno Otavio

2022.2 COMP0417 30 Fundamentos de Sistemas Embarcados
Seg 17:00-18:50 Evilson

2022.2 MAT0057 60 Fundamentos Elementares dade Matemática
Ter 15:00-16:50 Qui 15:00-16:50 Estombelo

2022.2 COMP0483 180 Prática Orientada em Computação I*
Qua 17:00-18:50 Qui 17:00-18:50 Rafael

2022.2 COMP0419 60 Prática em Sistemas Digitais*
Ter 13:00-14:50 Qui 13:00-14:50 Estombelo

2022.2 COMP0484 120 Prática Orientada em Computação II *
Seg 15:00-16:50 Ter 15:00-16:50 Qua 15:00-16:50 Qui 15:00-16:50 Rafael

2022.2 COMP0415 60 Arquitetura de Computadores
Seg 15:00-16:50 Qua 15:00-16:50 Rafael

2022.2 COMP0424 60 Aprendizagem de Máquina
Seg 13:00-14:50 Qua 13:00-14:50 Rene Pereira

Turmas Possiveis para o aluno: considerando (pre-requisito e nao repetir turma já cursada):
2022.2 COMP0308 120 Atividades Complementares
Seg 17:00-18:50 Ter 17:00-18:50 Qua 17:00-18:50 Qui 17:00-18:50 Leonardo Nogueira

2022.2 MAT0153 60 Cálculo C
Seg 13:00-14:50 Qua 13:00-14:50 Evilson

2022.2 COMP0417 30 Fundamentos de Sistemas Embarcados
Sex 15:00-16:50 Rafael

2022.2 COMP0409 60 Linguagens Formais e Computabilidade
Ter 13:00-14:50 Qui 13:00-14:50 Alberto Costa

2022.2 COMP0432 60 Processamento de Imagens
Sex 15:00-16:50 Sex 17:00-18:50 Rafael

2022.2 COMP0417 30 Fundamentos de Sistemas Embarcados
Qua 17:00-18:50 Rene Pereira

2022.2 COMP0405 60 Estruturas de Dados**
Sex 15:00-16:50 Sex 17:00-18:50 Rene Pereira

2022.2 COMP0419 60 Prática em Sistemas Digitais*
Sex 15:00-16:50 Sex 17:00-18:50 Rafael

2022.2 COMP0308 120 Atividades Complementares
Seg 15:00-16:50 Ter 15:00-16:50 Qua 15:00-16:50 Qui 15:00-16:50 Rene Pereira

2022.2 COMP0419 60 Prática em Sistemas Digitais*
Ter 13:00-14:50 Qui 13:00-14:50 Rene Pereira

2022.2 COMP0308 120 Atividades Complementares
Seg 15:00-16:50 Ter 15:00-16:50 Qua 15:00-16:50 Qui 15:00-16:50 Tarcisio Rocha

2022.2 MAT0153 60 Cálculo C
Seg 13:00-14:50 Qua 13:00-14:50 Rafael

2022.2 COMP0417 30 Fundamentos de Sistemas Embarcados
Seg 17:00-18:50 Evilson

2022.2 COMP0419 60 Prática em Sistemas Digitais*

Ter 13:00-14:50 Qui 13:00-14:50 Estombelo

2022.2 COMP0415 60 Arquitetura de Computadores
Seg 15:00-16:50 Qua 15:00-16:50 Rafael

2022.2 VAZIA 0 Disciplina Nula
<<

> Disciplinas Cursadas anteriormente pelo aluno:
> MAT0151 60 Cálculo A
> MAT0150 60 Vetores e Geometria Analítica
> MAT0057 60 Fundamentos Elementares dade Matemática
> COMP0480 30 Seminários em Computação
> COMP0393 60 Programação Funcional**
> MAT0152 60 Cálculo B
> MAT0078 60 Álgebra Linear I
> FISI0260 60 Física 1
> FISI0264 60 Laboratório de Física1*
> COMP0410 60 Lógica para Computação
> COMP0334 60 Programação Imperativa**
> <<

Variaveis : [E1, E2, E3, E4, E5, E6, E7]
|Dominios| = 7
|Dominios[0]| = 16
|Dominios[1]| = 16
|Dominios[2]| = 16
|Dominios[3]| = 16
|Dominios[4]| = 16
|Dominios[5]| = 16
|Dominios[6]| = 16
Tuplas da Relacao de Restricao = {(E1,E2,E3,E4,E5,E6,E7,),(E1,E2,E3,E4,E5,E6,E7,),
(E1,E2,E3,E4,E5,E6,E7,),(E1,E2,),(E1,E3,),(E1,E4,),(E1,E5,),(E1,E6,),(E1,E7,),(E2,E1,),(E2,E3,),
(E2,E4,),(E2,E5,),(E2,E6,),(E2,E7,),(E3,E1,),(E3,E2,),(E3,E4,),(E3,E5,),(E3,E6,),(E3,E7,),(E4,E1,),
(E4,E2,),(E4,E3,),(E4,E5,),(E4,E6,),(E4,E7,),(E5,E1,),(E5,E2,),(E5,E3,),(E5,E4,),(E5,E6,),(E5,E7,),
(E6,E1,),(E6,E2,),(E6,E3,),(E6,E4,),(E6,E5,),(E6,E7,),(E7,E1,),(E7,E2,),(E7,E3,),(E7,E4,),(E7,E5,),
(E7,E6,), }
|Restricoes| = 45

Resultado Final de Atribuições para para $E_1, \ldots, E_7$

==========

Após aplicar BacktrackingSearch.apply(csp)
É Solução ? : Sim

Atribuicoes Encontradas

> E1 = 2022.2 COMP0308 120 Atividades Complementares
> Ter 17:00-18:50 Qua 17:00-18:50 Qui 17:00-18:50
> E1.Perfil = Basico; E1.Departamento = DCOMP

> E2 = 2022.2 MAT0153 60 Cálculo C
> Seg 13:00-14:50 Qua 13:00-14:50 Evilson
> E2.Perfil = Basico; E2.Departamento = DMA

> E3 = 2022.2 COMP0417 30 Fundamentos de Sistemas Embarcados
> Sex 15:00-16:50 Rafael
> E3.Perfil = Basico; E3.Departamento = DCOMP

> E4 = 2022.2 COMP0409 60 Linguagens Formais e Computabilidade
> Ter 13:00-14:50 Qui 13:00-14:50 Alberto Costa
> E4.Perfil = Basico; E4.Departamento = DCOMP

> E5 = 2022.2 COMP0415 60 Arquitetura de Computadores
> Seg 15:00-16:50 Qua 15:00-16:50 Rafael
> E5.Perfil = Basico; E5.Departamento = DCOMP

> E6 = 2022.2 VAZIA 0 Disciplina Nula
> E6.Perfil = “”; E6.Departamento =

> E7 = 2022.2 VAZIA 0 Disciplina Nula
> E7.Perfil = “”; E7.Departamento =

==========