Ano Escolar

Resolução de Problemas de Desisão utilizando Programação em Lógica com Restrições

David Azevedo - up201405846 and João Ferreira - up201404332

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

Resumo O objetivo deste trabalho é a criação de um programa, em Programação em Lógica com Restrições, capaz de marcar trabalhos de casa e testes para um periodo do ano, restritos a certas condições.

Ao longo deste documento será descrito o desenvolvimento do trabalho realizado, as metologias e abordagens diferentes assim como os resultados obtidos.

1 Introdução

Devido à complexidade combinatória existente no planeamento de um Ano Escolar, é necessária uma solução optima que permita obter da melhor forma um plano de aulas ágil e eficaz.

Este trabalho tem como objetivo, para um dado Ano Escolar, marcar os trabalhos de casa e os testes para diferentes turmas e disciplinas, dependendo de variáveis definidas pelo utilizador. Neste documento está descrito detalhadamente o problema, uma abordagem de resolução do problema e uma análise sobre a mesma. Também será discutida a estratégia de pesquisa, os predicados de visualização e finalmente uma conclusão geral sobre o trabalho.

2 Descrição do Problema

O problema de otimização escolhido foi o Ano Escolar, neste problema é proposto que, o programa desenvolvido, seja capaz de marcar trabalhos de casa e testes para um número indefinido de turmas e de semanas. Em relação aos trabalhos de casa, para cada turma existe um dia livre e o número de TPC só pode ser metade do número de ocorrências da disciplina, não podendo existir mais que um número determinado de TPC's no mesmo dia. Em relação aos testes, cada disciplina tem dois testes, decorrendo num número de semanas especifico (meio e fim do período) e o número máximo de testes é dois por semana, sendo que não existem testes em dias consecutivos.

3 Abordagem

Num primeiro ato de abordagem ao problema, o problema foi dividido em dois subproblemas, a marcação dos trabalhos de casa e a marcação dos testes.

Para a marcação dos trabalhos de casa, foram realizados dois perdicados que resolvem o problema de dois pontos de vista diferente.

Para o primeiro predicado dos trabalhos de casa, foi utilizada uma estrutura de modo a limitar as variáveis utilizadas por disciplina, foi criada uma lista que contém os dias da semana, cada dia contem listas por disciplina com comprimento igual ao número de semanas, para indicar, em que semanas existem ou não TPCs, naquela disciplina, daquele dia da semana. A razão pela qual foi optada esta estrutura, foi numa tentativa de otimizar o espaço para representar os dias em que é possivél marcar trabalhos de casa para certa disciplina.

Para o segundo predicado dos trabalhos de casa, foi utilizada uma lista, onde cada index, indica a disciplina, e o elemento contido nesse index é um array de tamanho igual ao número de ocorrências, ao longo do período de aulas. Esta estrutura permite ter em atenção a totalidade de ocorrências de uma dada disciplina, permitindo distribuir de melhor forma os trabalhos de casa.

No problema dos testes, optamos por escolher uma estrutura que segue o seguinte formato : Uma lista de listas, onde cada lista representa uma turma, a lista da turma por sua vez tem uma lista para cada disciplina que por sua vez tem uma lista de dois elementos, a data do primeiro teste e a data do segundo (dia total, ou seja, semana * 5 + dia da semana). Achamos que esta seria uma boa estrutura uma vez que é organizada, direta e facilita a realização da maioria das restrições.

3.1 Variáveis de Decisão

Em relação aos trabalhos de casa, as variáveis de decisão são:

Para o primeiro predicado:

1. Dia Livre de Domínio [1,5], que representa um dos dias úteis da semana.
2. Lista de TPCs por disciplina de Domínio[0,1], que representa se existe ou não trabalho de casa em determinada semana, determinada pela sua posição.

A lista de TPCs por disciplina, é um conceito simples, mas ligeiramente difícil de explicar, como foi descrito na secção anterior, cada disciplina de um determinado dia, tem associada uma lista, de tamanho igual ao número de semanas de aulas, em que cada posição da lista representa o número da semana e o seu conteúdo indica se existe ou não trabalho de casa, 1 em caso positivo 0 em caso negativo.

Para o segundo predicado:

1. Dia Livre de Domínio [1,5], que representa um dos dias úteis da semana.

2. Lista de Dias com TPC por disciplina de Domínio[0, DiasSemana], que representa os dias em que existe trabalhos de casa, um valor maior que zero indica o dia do TPC.

Em relação aos testes, existem 2 tipos de variaveis de dominios, as variaveis para o primeiro teste e as variaveis para o segundo, no inicio da resolução do problema é calculado o intervalo para os primeiros testes e os segundos, que são representados como [Min1,Max1,Min2,Max2], estes valores representa (como dia total), o inicio e o fim de cada ronda de testes.

1. Dia Teste 1 de Domínio [Min1,Max1], que representa o primeiro intervalo dos testes.
2. Dia Teste 2 de Domínio [Min2,Max2], que representa o segundo intervalo de testes.

3.2 Restrições

Na prespetiva dos trabalhos de casa, as restrições aplicadas são:
Para o primeiro predicado:

1. Para cada disciplina de um determinado dia, o número de trabalhos de casa tem que ser igual à metade do número de semanas de aulas.
   Utilizando a estrutura descrita nas últimas secções, esta restrição faz-se simplesmente em prolog, utilizando o predicado count(1, Lista, #=, Semanas/2) em que Lista representa, por disciplina de um determinado dia, as semanas em que existe TPC e Semanas o número total de semanas.
2. Para cada dia, o número de trabalhos de casa não pode superar uma variável definida pelo utilizador.
   Para a implementação desta restrição, foram criados vários predicados com o intuito de, somar os n-ésimos elementos de um conjunto de listas, e verificar se esse valor é inferior ou igual ao número máximo de TPC's por dia.

Para o segundo predicado:

1. Para cada lista de TPC's por disciplina, o número de ocorrências do número zero (indicativo de que não existem trabalhos de casa) tem que ser igual à metade do número de ocorrêcias da disciplina, ao longo do ano (Esta razão pode ser presonalizada pelo utilizador). Cada elemento desta lista é restrito para os dias em que ocorre a disciplina associada, como pode ser observado na seguinte condição:

$$\begin{aligned} E &\subseteq TPCList \\ (E \bmod 5) &\subseteq Ocurrs \end{aligned} \tag{1}$$

   Onde $TPCList$ é a referida, lista de TPC's por disciplina, E é um elemento dessa lista e $Ocurrs$ é uma lista que contém os dias da semana em que hà essa disciplina.

2. Para cada dia o número de trabalhos de casa não pode superar um máximo. Esta restrição foi implementada numa prespetiva de tarefas, neste caso, cada tarefa tem duração e custo únitários e pretende-se descobrir os tempos iniciais de cada tarefa de modo a que nunca ultrapasse os recursos disponíveis : (limite máximo de tpc's).

$$comulative(Tasks, [limit(N_{TPCs})]) \tag{2}$$

Para isso foi utilizado o predicado comulative, acima representado, que recebe um conjunto de tasks e como parâmetero options, foi utilizado limit(número máximo de tpcs).

Nos Testes, as restrições são:

1. Ter no máximo apenas dois testes por semana.
2. Não ter teste em dias consecutivos (nem sobrepostos).
3. Os testes realizados pelas diferentes turmas a uma mesma disciplina devem ser o mais próximos possível.
4. A Turma ter aula da disciplina no dia do teste.

Para a primeira restrição, a solução elaborada consiste em converter os dias nas semanas correspondentes através do predicado mod/2, a restrição é feita sobre os valores dessa lista, indicando que nenhum valor pode ocorrer mais do que 2 vezes na lista com o uso do predicado nvalue/2, a lista das semanas é ordenada sem eliminar os duplicados e para cada conjunto de 3 valores consecutivos é aplicado o predicado por forma a garantir que existem pelo menos 2 valores diferentes nesses três elementos.

Para garantir que não haviam dois testes no mesmo dia foi utilizado o predicado all_distinct/1. O predicado cumulative/2 foi utilizado para restringir os testes a não ficarem em dias consecutivos, para isto foi criado um predicado que gerava as tasks e o principio básico foi que, "cada teste começa no dia anterior é realizado no próprio dia e acaba no dia seguinte", ou seja, cada teste tem um custo de 3 dias sendo que o dia do meio é o dia em que este é realizado.

Para diminuir a distancia entre os testes de uma mesma disciplina para as diferentes turmas foram usadas duas estratégias, a primeira estratégia foi somar os dias dos testes para cada disciplina e minimizar essa forma, para isso definimos uma estrutura que é uma lista de listas onde cada lista elemento representa um teste de uma disciplina e cada um dos seus elementos representa o dia para as diferentes turmas, com esta estrutura feita e os valores somados, utilizamos a opção de minimize/2 no labeling por forma a encontrar a opção melhor. Devido ao elevado número de combinações quando temos muitas , turmas, disciplinas, semanas, etc. optámos por utilizar a nossa estrutura inicial já referida para minimizar as distância resolvendo as variavéis começando o mais a esquerda possível (leftmost), isto funciona uma vez que as disciplinas são resolvidas por onde para cada uma das turmas, sendo que a ordem é global, portanto a ordem de disciplina pela qual cada turma realiza um teste é sempre a mesma e é sempre encontrado o valor mais cedo possível para essa realização. Por vezes esta solução

não é a mais otimizada, mas sim uma das mais otimizadas, optámos por trocar este pequeno detalhe em prol de uma melhor performance para a procura em casos de dimensões superiores.

Por fim, apesar de não ser o caso na faculdade, no caso dos ensinos básico e secundário, os testes são realizados durante uma aula, o que implica que para o teste ser possível de ser realizado a turma deverá ter aula dessa disciplina nesse mesmo dia, portanto, apessar de não ser mencionado no enunciado o grupo decidiu implementar essa restrição. Esta restrição foi usada foi primeiro lugar uma vez que restringe imediatamente os dias que podem ser escolhidos. O principio aqui é : criar uma lista, com uma lista para cada disciplina que contem os dias da semana em que a turma tem essa disciplina. Com esta lista restringimos os dominios dos dias dos testes para cada disciplina convertendo cada lista no set usando o predicado list_to_fdset/2 e aplicando a restrinção ao dia - Dia in_set DomainSet.

3.3 Estratégia de Pesquisa

Foi utilizada uma chamada ao labeling por defeito em todos os predicados de procura.

4 Visualização da Solução

Em relação aos trabalhos de casa: Para o primeiro predicado: O predicado que permite a visualização dos trabalhos de casa utiliza uma determinada turma, e os respetivos trabalhos de casa e representa da seguinte forma: O nome do dia da semana, se o dia é livre ou não, seguido das disciplinas, cada disciplina indica as semanas em que existe TPC. Tal pode ser observado, com mais clareza na seguinte imagem:

```
Terca
    Ciencias
       TPC: _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ X X X X X X X X X X X X X X X
    Portugues
       TPC: _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ X X X X X X X X X X X X X X X
    Historia
       TPC: X X X X X X X X X X X X X X X _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
```

Figura 1. Representação dos TPC's de uma disciplina

Este predicado é iterado várias vezes dependendo do número de turmas existentes.

Para o segundo predicado: É iterada a solução por disciplina e é imprimida a disciplina, o conjunto de semanas em que existe teste e o(s) respetivo(s) dia(s) da semana, tal pode ser observado na imagem seguinte:

```
TPC:Matematica
 Semana 1 Quarta
 Semana 2 Sexta   Quarta
 Semana 3 Sexta   Quarta
 Semana 4 Sexta   Quarta
 Semana 5 Sexta   Quarta
 Semana 6 Sexta   Quarta
 Semana 7 Sexta   Quarta
 Semana 8 Sexta   Quarta
 Semana 9 Sexta   Quarta
 Semana 10 Sexta   Quarta
 Semana 11 Sexta   Quarta
```

Figura 2. Representação dos TPC's de uma disciplina

Para os testes o resultado é mostrado por turma, para cada disciplina o dia do primeiro e segundo teste. Os números que representam os dias são números totais desde o inicio do periodo excluindo sábados e domingos. Por forma a ser de leitura simples e concisa. A divisão do valor por 5 devolve-nos a semana correspondente sendo que o mod nos devolve o dia da semana. Na imagem é possível observar o output da solução encontrada para 2 turmas com 5 disciplinas.

```
|          TESTES          |

Turma 1:
    Disciplina 1:
    Teste 1 - Dia 18 | Teste 2 - Dia 35
    Disciplina 2:
    Teste 1 - Dia 20 | Teste 2 - Dia 41
    Disciplina 3:
    Teste 1 - Dia 22 | Teste 2 - Dia 45
    Disciplina 4:
    Teste 1 - Dia 24 | Teste 2 - Dia 37
    Disciplina 5:
    Teste 1 - Dia 28 | Teste 2 - Dia 43
Turma 2:
    Disciplina 1:
    Teste 1 - Dia 18 | Teste 2 - Dia 35
    Disciplina 2:
    Teste 1 - Dia 20 | Teste 2 - Dia 37
    Disciplina 3:
    Teste 1 - Dia 24 | Teste 2 - Dia 39
    Disciplina 4:
    Teste 1 - Dia 22 | Teste 2 - Dia 42
    Disciplina 5:
    Teste 1 - Dia 28 | Teste 2 - Dia 44
```

Figura 3. Representação dos Testes para 2 Turmas com 5 Disciplinas

5 Resultados

5.1 Trabalhos de Casa

Para comparação dos dois perdicados, foram executados para a mesma turma(cada turma com 5 disciplinas distintas) um número variável de semanas, com limite de 2 trabalhos de casa por dia. Seguem-se a seguir os resultados.

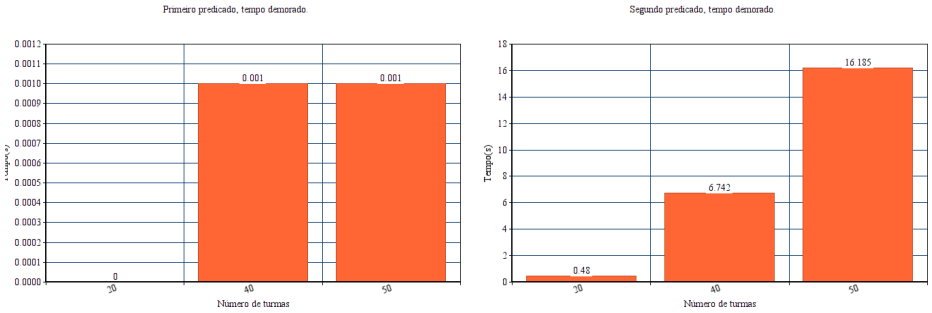

Figura 4. Testes executados em ambos predicados

Como é possível obervar o primeiro predicado teve um tempo de execução muito superior ao segundo, isto deve-se ao facto do uso de variáveis de domínio binárias e também por não se preocupar com o número de ocorrências da disciplina ao longo da semana, já que apenas marca TPC's para metade das semanas de aula de uma disciplina de um determinado dia. Em relação ao segundo predicado, por ter estas condições em conta e pelas variáveis não estarem instânciadas de forma binária, quanto maior o número de semanas, relfete-se o aumento exponencial do tempo.

5.2 Testes

Foram realizados 5 testes diferentes cada um deles com uma combinação diferente de complexidade, todos os resultados obtidos através dos métodos de estatisticas encontram-se nas imagens abaixo. Como é possível observar, o predicado que foi implementado é bastante rápido a encontrar uma solução, mesmo para problemas de grandes dimensões como é o caso da Figura 9. Desde que hajam semanas suficientes para marcar os testes todos, e os horários se encontrem bem elaborados, isto é, as disciplinas que são dadas apenas 1 vez por semana não estejam todas no mesmo dia por exemplo, o predicado encontra uma solução ótima para o problema de uma forma muito ágil e eficaz. De notar ainda que a Figura 6 demorou um pouco mais uma vez que o espaço temporal era menor (10 semanas) em relação ao número de turmas e disciplinas.

```
Time: 0.01s

Resumptions: 15144
Entailments: 3771
Prunings: 10791
Backtracks: 279
Constraints created: 328
```

Figura 5. 10 Semanas, 4 Turmas e 5 Disciplinas

```
Time: 0.03s

Resumptions: 58428
Entailments: 14676
Prunings: 41700
Backtracks: 1089
Constraints created: 1066
```

Figura 6. 10 Semanas, 13 Turmas e 5 Disciplinas

```
Time: 0.0s

Resumptions: 6219
Entailments: 1112
Prunings: 4184
Backtracks: 126
Constraints created: 392
```

Figura 7. 12 Semanas, 4 Turmas e 6 Disciplinas

```
Time: 0.01s

Resumptions: 20070
Entailments: 3605
Prunings: 13511
Backtracks: 405
Constraints created: 1274
```

Figura 8. 12 Semanas, 13 Turmas e 6 Disciplinas

```
Time: 0.02s

Resumptions: 28764
Entailments: 4266
Prunings: 19023
Backtracks: 489
Constraints created: 1368
```

Figura 9. 25 Semanas, 12 Turmas e 7 Disciplinas

6 Conclusões

Este projeto permitiu esclarecer vários conceitos relacionados com a programação em lógica com restrições, foi possivel, para o grupo, obter uma visão mais detalhada sobre este paradigma assim como a sua importância na resolução de diferentes problemas de otimização/decisão. Permitiu ainda uma melhor precepção sobre a importância da propagação das restrições de modo a melhorar o desempenho neste tipo de problemas.

Em relação aos trabalhos de casa:

Conclui-se que o primeiro predicado, devido à estrutura proposta para operar os dados foi possível otimizar a nivel de espaço o número de trabalhos de casa marcados, diminuindo consequentemente, o espaço de procura. A limitação trazida por esta estrutura é que, o plano escolar terá que ser tido em conta com semanas completas. De modo a melhorar a especificação dos trabalhos de casa, poderão ser implementadas mais variáveis, de modo a fornecer mais flexibilidade no momento da pesquisa, como por exemplo, o dia livre ser variável de semana para semana, a utilização de listas de diferença de modo a melhorar o desempenho de como é feita a concatenação de listas, entre outros.

Conclui-se que o segundo predicado, devido à sua estrutura, foi capaz de minimizar o número de dias utilizados para marcar TPC's e foi possível a adaptação do predicado comulative, referênciado anteriormente. Em comparação com o primeiro predicado, utiliza uma estrutura muito mais flexivel, fácil de iterar, melhorias poderiam passar, por melhorar a procura do dia livre, esta procura é um fator importante no momento da pesquisa por soluções, um dia bem escolhido, já facilita bastante a marcação dos TPC's, a limitação realizada pelo grupo ainda é um bocado simplista, escolhendo os melhores dias para casos simples.

Em relação aos testes:

Além do que já foi referido nos parágrafos anteriores, podemos concluir que a forma como modulámos um problema e a estrutura que utilizámos para o representar representa um peso significativo no desempenho da solução encontrada pelo solver do sicstus. Uma vez que a estrutura nos facilitou a aplicação das restrições assim como reduzir imenso o dominio das variaveis, foi possível obter-se um predicado que nos resolve um problema do dia-a-dia numa pequenissima fração de segundo. Possui uma implementação muito fácil de compreender do ponto de vista lógico e muito flexivel no ponto de vista de acrescentar mais restrições. Um dos maiores desafios neste problema assim como no problemas dos trabalhos de casa foi a geração de uma estrutura de dados em prolog que visava a fácil intrepertação e manipulação das variaveis de dominio por forma a resolver os problemas da maneira mais rápida e eficiente possível.

O grupo sente-se orgulhoso com o resultado alcançado e capaz de afirmar, que se encontra familiarizado com o paradigma da programação em lógica com restrições para a resolução de problemas complexos e de otimização.

X

Referências

1. SWI-Prolog, http://www.swi-prolog.org
2. SICStus-Prolog, https://sicstus.sics.se
3. Effective Modeling with Constraints - Roman Barták, http://link.springer.com/chapter/10.1007%2F11415763_10

Anexo

Código fonte

horarios.pl

```
:- use_module(library(clpfd)).
:- use_module(library(lists)).
:- ensure_loaded('tpcs.pl').
:- ensure_loaded('tpc2.pl').
:- ensure_loaded('utils.pl').
:- ensure_loaded('testes.pl').
:- ensure_loaded('statistics.pl').

disciplina(1,'Matematica').
disciplina(2,'Portugues ').
disciplina(3,'Historia  ').
disciplina(4,'Ciencias  ').
disciplina(5,'Artes     ').
disciplina(6,'EVT       ').
disciplina(7,'Ingles    ').

semana(1,'Segunda').
semana(2,'Terca').
semana(3,'Quarta').
semana(4,'Quinta').
semana(5,'Sexta').

%teste2.pl
nome_semana(1, 'Segunda').
nome_semana(2, 'Terca  ').
nome_semana(3, 'Quarta ').
nome_semana(4, 'Quinta ').
nome_semana(5, 'Sexta  ').

nome_discip(1,'Matematica').
nome_discip(2,'Portugues ').
nome_discip(3,'Historia  ').
nome_discip(4,'Ciencias  ').
```

```
nome_discip(5,'Artes     ').
nome_discip(6,'EVT       ').
nome_discip(7,'Ingles    ').

tpc2_disciplinas([1,2,3,4,5]).
tpc2_disciplinas2([1,2,3,4,5]).
tpc2_disciplinas3([1,2,3,4,5,6]).
tpc2_disciplinas4([1,2,3,4,5,6,7]).
tpc2_semana1([ [1,2,3] , [4,3] , [3,5] , [3,2] , [3,2,5] ]).
tpc2_semana2([ [2,1], [1] , [1,3] , [4] , [5] ]).

teste_turma1([[1,2,3],[4,2,3],[5,2,1],[3,4,2],[2,3,1]]).
teste_turma2([[3,2,1],[2,4,1],[2,2,1],[1,2,3,4,5],[2,3,1]]).

teste_turmas1([%4 turmas e 5 disciplinas Funciona nos tpcs tambem
              [[1,2,3],[4,2,3],[5,2,1],[3,4,2],[2,3,1]],
              [[3,2,1],[2,4,1],[2,5,1],[3,4,5],[2,3,1]],
              [[1,2,3],[4,2,3],[5,2,1],[2,3,1],[3,4,2]],
              [[5,2,1],[4,2,3],[1,2,3],[3,4,2],[2,3,1]]
            ]).

teste_turmas2([%13 turmas e 5 disciplinas - Funciona nos tpcs tambem
              [[1,2,3],[4,2,3],[5,2,1],[3,4,2],[2,3,1]],
              [[3,2,1],[2,4,1],[2,5,1],[3,4,5],[2,3,1]],
              [[1,2,3],[4,2,3],[5,2,1],[2,3,1],[3,4,2]],
              [[5,2,1],[4,2,3],[1,2,3],[3,4,2],[2,3,1]],
              [[1,2,3],[4,2,3],[5,2,1],[3,4,2],[2,3,1]],
              [[3,2,1],[2,4,1],[2,5,1],[3,4,5],[2,3,1]],
              [[1,2,3],[4,2,3],[5,2,1],[2,3,1],[3,4,2]],
              [[1,2,3],[4,2,3],[5,2,1],[3,4,2],[2,3,1]],
              [[3,2,1],[2,4,1],[2,5,1],[3,4,5],[2,3,1]],
              [[1,2,3],[4,2,3],[5,2,1],[2,3,1],[3,4,2]],
              [[1,2,3],[4,2,3],[5,2,1],[3,4,2],[2,3,1]],
              [[3,2,1],[2,4,1],[2,5,1],[3,4,5],[2,3,1]],
              [[1,2,3],[4,2,3],[5,2,1],[2,3,1],[3,4,2]]
            ]).

teste_turmas3([%4 turmas e 6 disciplinas
              [[1,2,3,5],[2,4,6,3],[1,5,2],[3,4,2],[4,6,1]],
              [[1,5,2],[3,4,2],[4,6,1],[1,2,3,5],[2,4,6,3]],
              [[1,2,3,5],[4,6,1],[2,4,6,3],[1,5,2],[3,4,2]],
              [[3,4,2],[4,6,1],[1,2,3,5],[2,4,6,3],[1,5,2]]
            ]).

teste_turmas4([%12 turmas e 7 disciplinas
```

```
              [[1,2,6,3,5],[2,4,6,3],[1,5,2,7],[7,3,4,2],[7,4,6,1]],
              [[7,1,5,2],[7,3,4,2],[7,4,6,1],[1,2,3,5],[2,4,6,3]],
              [[1,2,3,5],[7,4,6,1],[2,4,6,3],[1,5,2,7],[6,3,4,2]],
              [[3,4,2,6],[7,4,6,1],[1,2,3,5],[2,4,6,3],[7,1,5,2]],
              [[1,2,6,3,5],[2,4,6,3],[1,5,2,7],[7,3,4,2],[7,4,6,1]],
              [[7,1,5,2],[7,3,4,2],[7,4,6,1],[1,2,3,5],[2,4,6,3]],
              [[1,2,3,5],[7,4,6,1],[2,4,6,3],[1,5,2,7],[6,3,4,2]],
              [[3,4,2,6],[7,4,6,1],[1,2,3,5],[2,4,6,3],[7,1,5,2]],
              [[1,2,6,3,5],[2,4,6,3],[1,5,2,7],[7,3,4,2],[7,4,6,1]],
              [[7,1,5,2],[7,3,4,2],[7,4,6,1],[1,2,3,5],[2,4,6,3]],
              [[1,2,3,5],[7,4,6,1],[2,4,6,3],[1,5,2,7],[6,3,4,2]],
              [[3,4,2,6],[7,4,6,1],[1,2,3,5],[2,4,6,3],[7,1,5,2]]
            ]).

trabalhoPratico2(NSemanas,NumeroTPCs) :-
    teste_turmas1(Horarios),
    write('|          TESTES          |'), nl, nl,
    resolveTestesPorFavor(Horarios,5,NSemanas),%Numero de Disciplinas
    nl , nl , write('|           TPCS           |'), nl, nl,
    get_char(_),
    tpc2_disciplinas(Disciplinas),
    calculaTpcs(Horarios,1,Disciplinas, NSemanas, NumeroTPCs,2).

trabalhoPratico22(NSemanas,NumeroTPCs) :-
    teste_turmas2(Horarios),
    write('|          TESTES          |'), nl, nl,
    resolveTestesPorFavor(Horarios,5,NSemanas),%Numero de Disciplinas
    nl , nl , write('|           TPCS           |'), nl, nl,
    get_char(_),
    tpc2_disciplinas(Disciplinas),
    calculaTpcs(Horarios,1,Disciplinas, NSemanas, NumeroTPCs,2).

trabalhoPratico23(NSemanas,NumeroTPCs) :-%TPCS 4.
    teste_turmas3(Horarios),
    write('|          TESTES          |'), nl, nl,
    resolveTestesPorFavor(Horarios,6,NSemanas),%Numero de Disciplinas
    nl , nl , write('|           TPCS           |'), nl, nl,
    get_char(_),
    tpc2_disciplinas3(Disciplinas),
    calculaTpcs(Horarios,1,Disciplinas, NSemanas, NumeroTPCs,2).

trabalhoPratico24(NSemanas,NumeroTPCs) :-%TPCS 4.
    teste_turmas4(Horarios),
    write('|          TESTES          |'), nl, nl,
    resolveTestesPorFavor(Horarios,7,NSemanas),%Numero de Disciplinas
```

```
    nl , nl , write('|           TPCS           |'), nl, nl,
    get_char(_),
    tpc2_disciplinas4(Disciplinas),
    calculaTpcs(Horarios,1,Disciplinas, NSemanas, NumeroTPCs,2).
```

testes.pl

```
resolveTestesPorFavor(Horarios,NDisciplinas,NSemanas) :-
    solution(Horarios,NDisciplinas,NSemanas,Testes), !,
    printResults(Testes,1,1).

solution(Horarios,NDisciplinas,NSemanas,Testes) :-
    %Construcao dos dominios e das variaveis
    length(Horarios,NTurmas),
    buildTestes(NSemanas,NTurmas,NDisciplinas,Testes),
    Intervalo is 3 * NDisciplinas,%Intervalo da zona dos testes.
    MeioIntervalo is div(Intervalo,2),
    DiasTotais is NSemanas * 5,
    MetadeDias is div(DiasTotais,2),
    Min1 is MetadeDias - MeioIntervalo,
    Max1 is MetadeDias + MeioIntervalo,
    Min2 is DiasTotais - Intervalo,
    Max2 is DiasTotais, %Ultimo Dia
    Min1 > 1, Min2 > 1, %Falha mais rapido se nao for possivel
    solveTurma(Testes,Horarios,Min1,Max1,Min2,Max2),%Aplicacao das restricoes
    /*%Calcular distancia do uma disciplina entre turmas (deprecated)
    constroiEstrutura(Testes,NDisciplinas,DisciplinaTeste),
    getSum(DisciplinaTeste,Somas),
    sum(Somas,#=,DistTotal),% A logica e se a soma de todos os valores for a menor possivel eles vao ficar t
    */
    flat(Testes,Nivel1), flat(Nivel1,Vars),
    reset_timer,
    labeling([],Vars),%Solver
    print_time,
    fd_statistics.

% Recebe uma lista de listas e devolve uma lista em que cada elemento é a soma dos elementos de cada uma das
getSum([],[]) :- !.
getSum([Disciplina|Ds],Somas) :-
    getSum(Ds,D1),
    sum(Disciplina,#=,Soma),
    Somas = [Soma|D1].

/*Passa da estrutura de testes para uma estrutura que facilita a minimizacao dos testes de turmas diferentes
```

```
constroiPorDisciplinaAux([],_,_,[]) :- !.
constroiPorDisciplinaAux([Turma|Ts],IndexD,IndexT,Res) :-
    constroiPorDisciplinaAux(Ts,IndexD,IndexT,R1),
    getDataTurmaPorDisciplinaTeste(Turma,IndexD,IndexT,Data),
    Res = [Data|R1].

constroiPorDisciplina(_,I1,I2,_,[]) :- I1 is I2 + 1, !.
constroiPorDisciplina(Testes,IndexD,NDisciplinas,IndexT,Res) :-
    NewIndex is IndexD + 1,
    constroiPorDisciplina(Testes,NewIndex,NDisciplinas,IndexT,R1),
    constroiPorDisciplinaAux(Testes,IndexD,IndexT,R2),%Devolve a lista por disciplina
    Res = [R2|R1].

constroiEstrutura(Testes,NDisciplinas,Res) :-
    constroiPorDisciplina(Testes,1,NDisciplinas,1,Testes1),
    constroiPorDisciplina(Testes,1,NDisciplinas,2,Testes2),
    append(Testes1,Testes2,Res).

getDataTurmaPorDisciplinaTeste(Turma,IndexDisciplina,IndexTeste,Data) :-
    nth1(IndexDisciplina,Turma,Testes),
    nth1(IndexTeste,Testes,Data).
/*************************************************************************/

/*Resolucao para todos os testes de todas as turmas*/
solveTurma(Testes,Horarios,Min1,Max1,Min2,Max2) :-
    teste1PorTurma(Testes,Min1,Max1,Testes1),
    teste2PorTurma(Testes,Min2,Max2,Testes2),
    solveTurmaAux(Testes1,Horarios),
    solveTurmaAux(Testes2,Horarios).

solveTurmaAux([],[]) :- !.
solveTurmaAux([Testes|Ts],[Horario|Hs]):-
    solveTesteForTurma(Testes,Horario),
    solveTurmaAux(Ts,Hs).

/***********Resolve para uma turma***********/
solveTesteForTurma(Testes,Horario):-
    length(Testes,Ndis),
    calculaOcorrencias(Horario, _, OcurrDias, Ndis),
    diaDisciplina(Testes,OcurrDias),
    all_distinct(Testes),
    semTestesSeguidos(Testes),
    restringe2PorSemana(Testes).

%Resolve o problema de o teste ter de ser marcado num dia em que haja aula da disciplina
```

```
diaDisciplina([],[]) :- !.
diaDisciplina([Teste|Ts],[Disciplina|Ds]) :-
    list_to_fdset(Disciplina,DomainSet),
    Dia in_set DomainSet,
    Teste mod 5 #= Dia mod 5,
    diaDisciplina(Ts,Ds).

%Predicado que restringe a não haver testes em dias seguidos
semTestesSeguidos(Testes) :-
    buildTasks(Testes,1,Tasks),
    cumulative(Tasks,[limit(5)]).

%Constroi as tasks para o predicado semTestesSeguidos/1
buildTasks([],_,[]) :- !.
buildTasks([T|Ts],Index,Res) :-
    I1 is Index + 1,
    buildTasks(Ts,I1,R1),
    Tdepois #= T + 1,%A resposta passa por se o mod for 1 ou 0 (Segunda e sexta)
    Tantes #= T - 1,%e entao Tantes e Tdepois serao igual a T
    Res = [task(Tantes,_,Tdepois,5,Index)|R1].

%Verifica se a cada tres valores exitem pelo menos 2 diferentes e garante assim o max de 2 testes por semana
checkTrios([_,_]) :- !.
checkTrios([A,B,C|Ts]) :-
    nvalue(2,[A,B,C]),
    checkTrios([B,C|Ts]).

%Restringe os testes a 2 por semana
restringe2PorSemana(Testes) :-
    converteSemana(Testes,Res),
    msort(Res,Sorted),
    checkTrios(Sorted).

%Converte uma lista de dias totais numa lista de semanas
converteSemana([],[]) :- !.
converteSemana([DiaTotal|Ds],Res) :-
    converteSemana(Ds,R1),
    DiaTmp #= DiaTotal - 1,
    Tmp #= DiaTmp div 5,
    Sem #= Tmp + 1,
    Res = [Sem|R1].
/************************************************/

/*Devolve uma lista por turma para o 1 testes com as datas dos testes por disciplina*/
teste1PorTurma([],_,_,[]) :- !.
```

```
teste1PorTurma([Turma|Ts],MinBound,MaxBound,Res) :-
    teste1PorTurmaAux(Turma,MinBound,MaxBound,Testes),
    teste1PorTurma(Ts,MinBound,MaxBound,R1),
    Res = [Testes|R1].

teste1PorTurmaAux([],_,_,[]) :- !.
teste1PorTurmaAux([[A,_]|Ds],MinBound,MaxBound,Res) :-
    domain([A],MinBound,MaxBound),
    teste1PorTurmaAux(Ds,MinBound,MaxBound,R1), Res = [A|R1].
/*****************************************/

/*Devolve uma lista por turma para o 2 testes com as datas dos testes por disciplina*/
teste2PorTurma([],_,_,[]) :- !.
teste2PorTurma([Turma|Ts],MinBound,MaxBound,Res) :-
    teste2PorTurmaAux(Turma,MinBound,MaxBound,Testes),
    teste2PorTurma(Ts,MinBound,MaxBound,R1),
    Res = [Testes|R1].

teste2PorTurmaAux([],_,_,[]) :- !.
teste2PorTurmaAux([[_,A]|Ds],MinBound,MaxBound,Res) :-
    domain([A],MinBound,MaxBound),
    teste2PorTurmaAux(Ds,MinBound,MaxBound,R1), Res = [A|R1].
/*****************************************/

%Estrutura numero 1 para os testes
buildTestesAux([],_).
buildTestesAux([D|Ds],NSemanas) :-
    D = [_,_],
    MaxDias is NSemanas * 5,
    domain(D,1,MaxDias),
    buildTestesAux(Ds,NSemanas).
buildTestesDisc(_,_,[]).
buildTestesDisc(NSemanas,NDisciplinas,[L|Ls]) :-
    length(L,NDisciplinas),
    buildTestesAux(L,NSemanas),
    buildTestesDisc(NSemanas,NDisciplinas,Ls).
buildTestes(NSemanas,NTurmas,NDisciplinas,Res) :-
    length(Res,NTurmas),
    buildTestesDisc(NSemanas,NDisciplinas,Res).

/*HELPERS*/
%Sort que nao elimina duplicados
msort(Keys, KeysS) :-
   keys_pairs(Keys, Pairs), % pairs_keys(Pairs, Keys)
   keysort(Pairs, PairsS),
```

```
   pairs_keys(PairsS, KeysS).

keys_pairs([], []).
keys_pairs([K|Ks], [K-_|Ps]) :-
   keys_pairs(Ks, Ps).

pairs_keys([], []).
pairs_keys([K-_|Ps],[K|Ks]) :-
   pairs_keys(Ps, Ks).


/*PRINTERS*/

printDisciplina(Testes,NDisciplina) :-
    format("    Disciplina ~w:", [NDisciplina]), nl,
    format("    Teste 1 - Dia ~w | Teste 2 - Dia ~w",Testes), nl, !.

printTurma([],_,_) :- !.
printTurma([Disciplina|Ds],NTurma,NDisciplina) :-
    printDisciplina(Disciplina,NDisciplina),
    D1 is NDisciplina + 1, !,
    printTurma(Ds,NTurma,D1).

printResults([],_,_) :- !.
printResults([Turma|Ts],NTurma,NDisciplina) :-
    format("Turma ~w:",[NTurma]), nl,
    printTurma(Turma,NTurma,NDisciplina),
    N1 is NTurma + 1, !,
    printResults(Ts,N1,NDisciplina).
```

tpcs.pl

```
:- use_module(library(clpfd)).
:- use_module(library(lists)).

testeSemana([[1,2],[3,4],[5,4],[3,2,4],[1,2,3]]).

%analisar turma
turmaTPCs([Turma|Ts], Semanas, MaximoTPCs):-
  turmaTPCs1([Turma|Ts], 1, Semanas, MaximoTPCs).

turmaTPCs1([], _, _, _).
turmaTPCs1([Turma|Ts], N, Semanas, MaximoTPCs):-
  write('------------------'), nl,
```

```
  write('Turma: '), write(N), nl, !,
  tpcs(Turma, Semanas, MaximoTPCs, TPCs, DiaLivre), !,
  printTPCResult(Turma, TPCs, DiaLivre),
  get_char(_),
  N2 is N + 1, !,
  turmaTPCs1(Ts, N2, Semanas, MaximoTPCs).



%Adicionar razao de tpc por disciplina.
tpcs(Semana, NS, NTPC, TPCs, DiaLivre):-
  domain([DiaLivre],1,5),
  domain([RTPC],NTPC,NTPC),
  constroiLabels(Semana, NS, DiaLivre, TPCs),
  limitarTPCs(TPCs, RTPC, NS, DiaLivre),
  flat(TPCs,  ResultMid),
  flat(ResultMid, Results),
  labeling([],Results).

%
%Limitar os TPCs a dois
limitarTPCs3([],_,0).

limitarTPCs3([Disciplina| Ds], Semana, R):- !,
  nth1(Semana, Disciplina, N1),
  R #= R1 + N1,
  limitarTPCs3(Ds, Semana, R1).

limitarTPCs2(_, _, S, NS):- S > NS, !.
limitarTPCs2(Dia, NTPC, Semana, NS):- Semana =< NS, !,
  R #=< NTPC,
  limitarTPCs3(Dia, Semana, R),
  S2 is Semana + 1,
  limitarTPCs2(Dia, NTPC, S2, NS).

limitarTPCs1([], _, _, _).
limitarTPCs1([Dia|Ds], NTPC, NS, DiaLivre):-
  limitarTPCs2(Dia, NTPC, 1, NS),
  limitarTPCs1(Ds, NTPC,NS,DiaLivre).


limitarTPCs(Semana, NTPC, NS, DiaLivre):-
  limitarTPCs1(Semana, NTPC, NS, DiaLivre).
```

```
%bla
constroiLabels2([_ | Ds], NS, livre, [A | As]):-
  length(A, NS),
  domain(A, 0, 0),
  constroiLabels2(Ds, NS, livre, As).

constroiLabels2([_ | Ds], NS, ocupado, [A | As]):-
  length(A, NS),
  domain(A, 0, 1),
  NK is div(NS,2),
  count(1, A, #=, NK),
  constroiLabels2(Ds, NS, ocupado, As).

constroiLabels2([], _, _, []).


constroiLabels1([Dia | Ds], N, NS, DiaLivre, [A|As]):- N \= DiaLivre, !,
  length(Dia, Disciplinas), length(A, Disciplinas),
  constroiLabels2(Dia, NS, ocupado, A),
  N1 is N + 1,
  constroiLabels1(Ds, N1, NS, DiaLivre, As).

constroiLabels1([Dia | Ds], N, NS, N, [A|As]):- !,
  length(Dia, Disciplinas), length(A, Disciplinas),
  constroiLabels2(Dia, NS, livre, A),
  N1 is N + 1,
  constroiLabels1(Ds, N1, NS, N, As).

constroiLabels1([], _, _, _, []).

constroiLabels(Semana, NS, DiaLivre, Array):-
  length(Array, 5),
  constroiLabels1(Semana, 1, NS, DiaLivre, Array).

%prints

printTPCResultTPC([]).
printTPCResultTPC([1|Es]):-
  write('X '),
  printTPCResultTPC(Es).

printTPCResultTPC([0|Es]):-
  write('_ '),
  printTPCResultTPC(Es).
```

```
printTPCResultAux2([],[]).
printTPCResultAux2([Dis| Ds], [Tpc| Ts]):-
  disciplina(Dis,DisNome),
  write('    '), write(DisNome), nl,
  write('    '), write('   TPC: '),
  printTPCResultTPC(Tpc), nl,
  printTPCResultAux2(Ds,Ts).

printDisciplinas([]).
printDisciplinas([Dis| Ds]):-
  disciplina(Dis,DisNome),
  write('    '), write(DisNome), nl,
  printDisciplinas(Ds).

printTPCResultAux1([],[],_,_).
printTPCResultAux1([Dia|Ds], [_|TDs], Ns, Ns):- !,
  semana(Ns, Semana),
  write(Semana), write(': Dia Livre'),nl,
  N2 is Ns + 1,
  printDisciplinas(Dia),
  printTPCResultAux1(Ds, TDs, Ns, N2).

printTPCResultAux1([Dia|Ds], [TDia|TDs], _, Ns):-
  semana(Ns, Semana),
  write(Semana), nl,
  N2 is Ns + 1,
  printTPCResultAux2(Dia, TDia),
  printTPCResultAux1(Ds, TDs, Ns, N2).

printTPCResult(Turma, TPCs, DiaLivre):-
  printTPCResultAux1(Turma, TPCs, DiaLivre, 1).
```

tpc2.pl

```
calculaTpcs([],_,_,_,_,_) :- !.
calculaTpcs([Turma|Ts],NTurma, Disciplinas, Semanas, NumeroTPCs,MaxTpcs):-
    write(' Turma '), write(NTurma), nl,
    N1 is NTurma + 1, !,
  pre_marca_tpcs(Turma, Disciplinas, Semanas, NumeroTPCs,MaxTpcs, _, _), get_char(_),
  calculaTpcs(Ts,N1, Disciplinas, Semanas, NumeroTPCs,MaxTpcs).

pre_marca_tpcs(Semana, Disciplinas, Ns, NTPC,MaxTpcs, TPCs, DiaLivre):-
  length(Disciplinas, Ndis),
  calculaOcorrencias(Semana, Ocorrencias, OcorrenciasDias, Ndis),
```

```
%  write('Ocorrencias'), write(Ocorrencias), nl,
%  write('ODia'), write(OcorrenciasDias), nl,
%  desenvolveOcorrencias(OcurrDias, OcorrenciasDias, Ns, Ndis).
  marca_tpcs(Ocorrencias, OcorrenciasDias, Ns, NTPC, TPCs, DiaLivre,MaxTpcs),
%  write('Resultado'), nl,
  print_tpc1(Semana, TPCs, DiaLivre).


%desenvolveOcorrencias([O | Os], [R | Rs], Ns, Ndis):-


marca_tpcs(Ocorrencias, OcorrenciasDias, NS, NTPC, TPCs, DiaLivre,MaxTpcs):-
  Dias is NS * 5,
  limitDia(OcorrenciasDias, Res),
  list_to_fdset(Res, Set),
  DiaLivre in_set Set,
  criarListagemPDisciplina(NS, Dias, DiaLivre, Ocorrencias, OcorrenciasDias, TPCs,MaxTpcs),
%write('FIM'). testeA(Semana, Disciplinas, Ocorrencias, OcorrenciasDias, NS, NTPC, TPCs, DiaLivre):-
  %criarRectangulos(TPCs, Rects),
  criarTasks(TPCs, Tasks),
  cumulative(Tasks, [limit(NTPC)]),
  %disjoint2(Rects, [ bounding_box([0,0], [Dias, NTPC]) ]),%wrap(0,Dias,0, NTPC)]),
  flat(TPCs, Result),
  K = [DiaLivre | Result], !,
  labeling([], K).


limitDia1([],A, A).
limitDia1([[K] | Rs],Dia, T2):-member(K, Dia), !,
  select(K, Dia, T1),
  limitDia1(Rs,T1, T2).

limitDia1([_ | Rs],Days, T2):- !,
  limitDia1(Rs, Days, T2).


limitDia(Ocurrencias, Res):-
  limitDia1(Ocurrencias, [1,2,3,4,5], Res).


adicionaRects([L | []], Rs, [A | Rs]):-
  L #\= 0 #<=> M,
  domain([Temp], 1,2),
  A = f(L, M, Temp, M).
```

```
adicionaRects([L | Ls], Rs, [R | T1]):-
  adicionaRects(Ls, Rs, T1),
  L #\= 0 #<=> M,
  domain([Temp], 1,2),
  R = f(L, M, Temp, M).


criarRectangulos([], []).
criarRectangulos([TPC | Ts], Rects):-
  criarRectangulos(Ts, R1),
  adicionaRects(TPC, R1, Rects).

%asdasd

adicionaTasks([L | []], Rs, [A | Rs]):-
  L #\= 0 #<=> M,
  length(Rs, ID),
  A = task(L, M, _, M, ID).

adicionaTasks([L | Ls], Rs, [R | T1]):-
  adicionaTasks(Ls, Rs, T1),
  L #\= 0 #<=> M,
  length(Rs, ID),
  R = task(L, M, _, M, ID).


criarTasks([], []).
criarTasks([TPC | Ts], Rects):-
  criarTasks(Ts, R1),
  adicionaTasks(TPC, R1, Rects).

%asdd

aplicarDominio([], _, _).
aplicarDominio([V | Ls], DomainSet, DiaLivre):-
  DiasDaSemana in_set DomainSet,
  (V #=< 0) #\/ ( (V #> 0) #/\
  ((V mod 5) #= DiasDaSemana mod 5 ) #/\ ((V mod 5) #\= DiaLivre mod 5) ),
  aplicarDominio(Ls, DomainSet, DiaLivre).

criarListagemPDisciplina(_, _, _, [], [], [],_).
criarListagemPDisciplina(NumeroSemanas, Dias, DiaLivre, [Ocurr | Os], [OcurrD | Ocurrs], [TPC | Ts],MaxTpcs)
  criarListagemPDisciplina(NumeroSemanas, Dias, DiaLivre, Os, Ocurrs, Ts,MaxTpcs),
  Total is Ocurr * NumeroSemanas,
  length(TPC, Total),
```

```
  list_to_fdset(OcurrD, Set),
  domain(TPC, 0, Dias),
  aplicarDominio(TPC, Set, DiaLivre),
  Mid is div(Total,MaxTpcs),
  MidP1 is Mid + 1,
  nvalue(MidP1, TPC),
  count(0, TPC, #=, Mid).



calculaOcorrencias3(Ls, N, Dia, 1, Vs, RVs):- member(N, Ls), RVs = [Dia | Vs].
calculaOcorrencias3(Ls, N, _, 0, Vs, Vs):- \+ member(N, Ls).

calculaOcorrencias2([], _, 6, 0, []).
calculaOcorrencias2([Dia|Ds], N, Day, V, VD):-
  calculaOcorrencias2(Ds, N, Dia2, V2, VD2),
  Day is Dia2 - 1,
  calculaOcorrencias3(Dia, N, Day, R, VD2, VD),
  V is V2 + R.

calculaOcorrencias1(_, [], [], Ndis, N):- Ndis < N, !.
calculaOcorrencias1(Semana, Ocurr, OcurrDias, Ndis, N):- N =< Ndis, !,
  N2 is N + 1,
  calculaOcorrencias2(Semana, N, _, V, VD),
  calculaOcorrencias1(Semana, O2, OD2, Ndis, N2),
  OcurrDias = [VD|OD2],
  Ocurr = [V|O2].


calculaOcorrencias(Semana, Ocurr, OcurrDias, Ndis):-
  calculaOcorrencias1(Semana, Ocurr, OcurrDias, Ndis, 1).

%print tpcs

translate(0, 5).
translate(N, N).


print_discp([]).
print_discp([D | Ds]):-
  nome_discip(D, N),
  write('    '), write(N), nl,
  print_discp(Ds).

print_horario([], _).
```

```
print_horario([S | Ss], Ns):-
  nome_semana(Ns, SS),
  write(SS), nl,
  print_discp(S),
  N2 is Ns + 1,
  print_horario(Ss, N2).

print_tpcPorDisciplina2([],_).
print_tpcPorDisciplina2([0 | Vs], S):- !,
  print_tpcPorDisciplina2(Vs,S).
print_tpcPorDisciplina2([N | Vs], S):-
  Kappa = div(N, 5),
  K1 is Kappa + 1,
  K1 \= S, !,
  nl, write(' Semana '), write(K1), write(' '),
  Mod = mod(N, 5),
  Mod2 is Mod,
  translate(Mod2,Mod3),
  nome_semana(Mod3, Nome2),
  write(Nome2), write(' '),
  print_tpcPorDisciplina2(Vs, K1).

print_tpcPorDisciplina2([N | Vs], S):-
  Mod = mod(N, 5),
  Mod2 is Mod,
  translate(Mod2,Mod3),
  nome_semana(Mod3, Nome2),
  write(Nome2), write(' '),
  print_tpcPorDisciplina2(Vs, S).


print_tpcPorDisciplina([],_).
print_tpcPorDisciplina([TPC | Ts], Ns):-
  nome_discip(Ns, Nome), nl, nl,
  write('TPC:'), write(Nome),
  print_tpcPorDisciplina2(TPC, 0),
  N2 is Ns + 1,
  print_tpcPorDisciplina(Ts,N2).


print_tpc1(Semana, TPCs, DiaLivre):-
  print_horario(Semana, 1),
  nome_semana(DiaLivre, NomeDiaSemana),
  write('Dia Livre: ') , write(NomeDiaSemana), nl,
  get_char(_),
```

```
  print_tpcPorDisciplina(TPCs, 1) , nl.
```

utils.pl

```
flat([],[]).
flat([T|Ts], Rs):-
  flat(Ts, R1),
  append(T, R1,Rs).
```

statistics.pl

```
:- use_module(library(clpfd)).

problem(Vars) :-
length(Vars,10),
domain(Vars,1,100),

all_distinct(Vars),

Vars = [V|Vs],
maximum(V,Vars),
sum(Vs,#=,V),

reset_timer,
labeling([],Vars),
% labeling([down],Vars),
print_time,
fd_statistics.



reset_timer :- statistics(walltime,_).
print_time :-
statistics(walltime,[_,T]),
TS is ((T//10)*10)/1000,
nl, write('Time: '), write(TS), write('s'), nl, nl.
```