“Escola de Línguas”

Resolução de Problema de Optimização usando Programação em Lógica com Restrições

Jorge Reis (ei08053), Miao Sun (ei08162)

{FEUP-PLOG, Turma 3MIEIC1, Grupo 01}

Resumo. Este projecto teve como objectivo principal desenvolver um programa em Programação em Lógica com Restrições (PLR) que permitisse optimizar a abertura de novos cursos numa escola de línguas, baseado na valência dos professores contratados. Utilizou-se o sistema de desenvolvimento SICStus Prolog, que inclui um módulo de resolução de restrições sobre domínios finitos. Conseguiu-se assim, através do programa, chegar a um resultado óptimo num curtíssimo espaço de tempo. Outro objectivo deste trabalho foi a continuação da aprendizagem da linguagem Prolog e a familiarização com o paradigma da programação em lógica com restrições.

Keywords: Prolog, Programação em Lógica com Restrições, clp(FD), escola de línguas

1 Introdução

Realizado no âmbito da disciplina de Programação em Lógica leccionada no 3º ano do Mestrado Integrado em Engenharia Informática e Computação, este trabalho tem como principal objectivo o desenvolvimento de uma pequena aplicação capaz de optimizar a abertura de novos cursos de línguas numa escola e o escalonamento dos professores disponíveis na mesma, obedecendo a determinadas restrições e critérios, aplicando os conhecimentos adquiridos ao longo das aulas da disciplina.

A escolha deste problema deveu-se essencialmente ao interesse suscitado pela optimização para um caso que podia perfeitamente enquadrar-se num problema real. No entanto, este trabalho constituiu efectivamente um desafio uma vez que ainda estavam a ser desenvolvidas competências de representação de problemas de forma declarativa, ao contrário da programação funcional, mais utilizada até então, e o trabalho foi desenvolvido enquanto a matéria ia sendo leccionada e num curto espaço de tempo.

Ao longo deste artigo será descrito pormenorizadamente o problema, será explicada a abordagem feita, assim como resultados e conclusões obtidas.

2 Descrição do Problema

O problema proposto consiste numa escola de línguas, que pretende alargar a sua oferta de cursos, tendo em conta o lucro que daí se poderá obter.

A escola oferece cursos de Espanhol, Francês e Inglês. Para tal, tem três professores, com estatutos diferentes. A prof. Annette é natural de França, mas também fala fluentemente Português, Espanhol e Alemão. O prof. Bóris é proveniente da Rússia mas teve que se refugiar em Itália, França, Espanha e Portugal por estar a ser procurado pelo KGB. Como tal, fala fluentemente as línguas desses países. Por último, o prof. Charles é americano e estava em Cuba de férias durante o período da Guerra Fria e da crise dos mísseis. Dessa forma, é versado também em Espanhol, Russo e Italiano (só porque gosta de ver filmes italianos).

A escola tem ainda duas funcionárias administrativas necessárias para o registo e manutenção dos cursos de línguas. Possui ainda duas salas, que são usadas para ministrar os cursos. Devido à valência dos professores, a direcção da escola pretende agora alargar o número de cursos para incluir Italiano, Alemão e Russo.

- Cada curso tem que ter um mínimo de 4h por semana e um máximo de 8h.
- Um professor tem que dar no mínimo um curso, mas pode dar no máximo dois cursos.
- A prof. Annette é a mais nova, pelo que ganha 25€ por hora de aula. Como está em início de carreira pode trabalhar até um máximo de 40h por semana.
- O prof. Bóris é o mais velho, mas por ordens médicas só pode trabalhar um máximo de 20h por semana. O seu vencimento é de 40€ à hora.
- O prof. Charles teve recentemente um filho e só poderá trabalhar um máximo de 30h por semana. O seu vencimento é de 30€ à hora.
- A manutenção de um curso requer cerca de 15h semanais a uma funcionária administrativa.
- Cada funcionária administrativa trabalha 40h por semana e aufere 15€ por hora. No entanto, poderá realizar até um máximo de 5h extra por semana, com um custo de 25€ à hora. Por outro lado, pode-se contratar uma funcionária adicional em part-time, que ganha 10€ à hora e trabalha um máximo de 10h por semana.

- Cada sala só pode ter uma aula de cada vez.
- Cada aluno paga mensalmente o equivalente a 10€ por cada hora de aula.
- Cada curso tem a duração de um ano.

Ao abrir os novos cursos, a escola pretende maximizar o lucro, reduzindo os custos operacionais e admitindo que cada curso preencherá todas as vagas disponíveis (15 para cada curso).

Assim, o objectivo é definir o problema como um problema de satisfação de restrições, e resolver com PLR, de modo a que seja possível resolver problemas desta classe com diferentes parâmetros, isto é, deverá ser possível resolver problemas com mais ou menos línguas ou número de vagas diferentes por curso, com restrições parametrizáveis relativas aos cursos ministráveis por professor, tempos de contratação, salas, etc.

2.1 Análise do Problema Proposto

Após análise e discussão do problema com os professores da disciplina tomou-se, em concordância, algumas decisões por incoerência do enunciado.

No problema, cada professor apresenta um número máximo de horas semanais de trabalho diferente, nomeadamente 40h, 20h e 30h. Porém, uma das restrições impostas é que nenhum professor poderá leccionar mais do que 2 cursos. Assim, o número máximo de horas que consideramos para cada professor foi 16h, tendo em conta que cada curso tem um máximo de 8h semanais. Também não impusemos nenhuma restrição a nível de salas, pois não faria muito sentido no problema proposto e principalmente na sua dimensão.

Então, sabendo que cada aluno paga uma propina horária e tendo em conta o vencimento de professores e funcionárias, pretende-se obter o escalonamento de cursos segundo professor e número de horas que ofereça um maior lucro semanal.

3 Ficheiros de Dados

Foram criados 3 ficheiros com dimensões de dados diferentes e cada um com vários casos particulares no que diz respeito a número de cursos e número de vagas. Os ficheiros (dados1.pl, dados2.pl e dados3.pl) estão incluídos no cimo do ficheiro do programa, bastando descomentar o que se pretende utilizar.

No ficheiro dados1 há vários casos, tendo em conta as informações do enunciado. No "*caso1()*", temos apenas uma resolução para as 3 línguas que existiam inicialmente na escola, e no "*caso2()*" temos para as 6 línguas que a escola pretende incluir. Assumindo sempre os 3 professores fornecidos e as 15 vagas impostas no enunciado. Nos casos seguintes e restantes ficheiros procuramos demonstrar que o programa é capaz de resolver problemas diferentes e de dimensões maiores.

Assim, no ficheiro 2 existem mais línguas e mais professores (com valências diferentes), e como tal restrições diferentes para limites horários de funcionárias. Depois tem-se vários casos de teste, que contemplam mais ou menos línguas diferentes, e números de vagas também diferentes. O mesmo acontece no ficheiro 3, sendo que este tem uma dimensão de dados bastante maior.

4 Variáveis de decisão

As variáveis de decisão usadas consistem em duas listas, NProf e HCurso, correspondendo respectivamente ao número do professor e as horas semanais para cada curso. Em ambas o índice de cada elemento corresponde a um curso, como tal são criadas com os predicados: length(NProf,N) e length(HCurso,N), em que N é o número de cursos pretendidos. Para a lista NProf, o domínio definido é de 1 a NP, em que NP é o número de professores da escola, e para HCurso é de 4 a 8, pois cada curso tem entre 4 e 8 horas. Os predicados domain(NProf,1,NP) e domain(HCurso,4,8) simbolizam estas restrições de domínio.

5 Restrições

No desenvolvimento da aplicação em Prolog, recorreu-se a um módulo de resolução de restrições sobre domínios finitos, tendo sido utilizadas diversas restrições no sentido de obter a solução pretendida.

Em suma, foram aplicadas as seguintes restrições:

- Cada curso terá que ter obrigatoriamente entre 4 e 8 horas por semana;
- Cursos de Línguas que cada professor pode leccionar;
- Cada professor terá que leccionar pelo menos um curso, e no máximo 2;
- O número de horas extra que cada funcionária poderá realizar é 5h semanais (10h no total das duas), e a contratação de uma funcionária em part-time impõe um limite de 10h.

Inicialmente são definidos os tamanhos das listas HCurso e NProf, com o número de cursos para o caso dado, e os domínios dos valores destas variáveis. Assim, começa-se por restringir, para cada curso, apenas os professores que têm habilitações para os leccionar, através do predicado *aplica_prof(IndispProfCurso,IndCurso,NProf)*.

De seguida, através do predicado *verifica_nr_cursos_prof(NProf,NP)*, aplica-se a restrição que define que cada professor será responsável por pelo menos um curso, e no máximo 2.

Ao longo do programa são utilizadas várias outras restrições, como:

- **Restrições de pertença**, por exemplo os domínios das horas extra das funcionárias;
- **Restrições aritméticas**, como o predicado *sum*;
- **Restrições proposicionais**, incluindo restrições materializáveis, e restrições combinatórias.

6 Estratégia de Pesquisa

Como forma de pesquisa da melhor solução para o problema dado, utilizou-se o predicado de optimização *maximize(LucroSemanal)* que nos permite encontrar a solução que apresenta um Lucro Semanal mais elevado após atribuição de valores possíveis às variáveis do problema. Porém, como forma de melhorar a performance do programa a nível temporal, antes deste *labeling* final, fez-se um *labeling* às variáveis que representam as horas extra das funcionárias administrativas e as horas da possível funcionária part-time, utilizando o predicado *minimize*, o que permite obter em primeiro lugar, para o número de cursos dado, o resultado para um menor gasto relativamente à manutenção dos cursos, ou seja, uma despesa menor com funcionários.

Para além disto, foram testadas quatro estratégias de pesquisa no que diz respeito à ordenação de variáveis: *ff*, *ffc*, *min* e *max*. Como se pode constatar na secção 8 do presente artigo, embora para casos de pequena dimensão praticamente não seja observada diferença temporal na obtenção de resultados entre a utilização de todas estas heurísticas, para casos de grandes dimensões observa-se uma melhoria muito significativa com a utilização da opção de ordenação *min*, que leva a que seja tratada primeiro a variável com menor *lower bound*. Assim sendo, esta foi a estratégia final implementada no programa.

7 Visualização da Solução

Para resolver um problema utilizando o programa, deve-se atribuir a uma variável "Caso", um dos predicados de inicialização de casos presentes no ficheiro escolhido (por exemplo, *caso1(Caso)*). Depois chama-se o predicado *solve(Caso)*, que iá resolver o problema para o caso escolhido.

Para visualizar a solução obtida para o problema proposto, após o "*labeling*" ocorre o output do Lucro Semanal para a solução óptima do mesmo. De seguida é invocado o predicado *print_plano(HCurso,NProf,ListaProfs,IndCurso)*, que irá, de modo recursivo, apresentar no ecrã o nome do curso, o professor que administra e o número de horas, para todos os cursos do caso dado. Por fim, apresenta o número de horas

extra das funcionárias administrativas e o número de horas de uma possível funcionária em part-time (0h caso não seja necessária).

São também apresentadas estatísticas de execução, através do predicado *fd_estatistics*.

Como exemplo, de seguida é apresentado o output do programa para o caso base do enunciado. Assim, com 3 professores e 6 cursos de línguas, pretendemos obter o resultado em termos de distribuição de professores, horas por curso e horas extras de funcionários, caso necessário, que nos permitem obter um lucro mais elevado.

```
| ?- caso2(Caso), solve(Caso).

Maximo Lucro Semanal: 4380

Plano:
 Curso de Espanhol
    Professor: Annete
    Numero Horas Semanais: 8
 Curso de Frances
    Professor: Boris
    Numero Horas Semanais: 8
 Curso de Ingles
    Professor: Charles
    Numero Horas Semanais: 8
 Curso de Italiano
    Professor: Charles
    Numero Horas Semanais: 8
 Curso de Alemao
    Professor: Annete
    Numero Horas Semanais: 8
 Curso de Russo
    Professor: Boris
    Numero Horas Semanais: 8

 Horas Extra Necessarias das Funcionarias Administrativas: 0
 Horas Necessarias de Funcionaria Part-time: 10

Solutions in 0.107 seconds.

Resumptions: 6122
Entailments: 1548
Prunings: 5533
Backtracks: 105
Constraints created: 59
```

Fig. 1. Visualização da solução em modo texto para o problema proposto.

8 Resultados

Como foi exposto na secção 6, foram efectuados diversos testes ao programa com diferentes estratégias de pesquisa. Estes testes incidiram principalmente sobre o tempo de execução, pelo que procuramos uma estratégia que nos possibilitasse obter o resultado óptimo no menor espaço de tempo.

De seguida são apresentadas tabelas com alguns resultados de desempenho com estratégias de pesquisa diferentes no *labeling* da variável **LucroSemanal,** e para casos também de dimensões diferentes.

C \ Op	normal	ff	ffc	min	max
Caso 1	0.140 s	0.137 s	0.141 s	0.139 s	0.142 s
Caso 2	0.216 s	0.212 s	0.217 s	0.215 s	0.206 s
Caso 3	0.168 s	0.168 s	0.164 s	0.166 s	0.167 s
Caso 4	0.192 s	0.198 s	0.187 s	0.193 s	0.194 s
Caso 5	0.222 s	0.216 s	0.218 s	0.219 s	0.213 s

Tabela 1. Desempenhos para casos do ficheiro dados1

C \ Op	normal	ff	ffc	min	max
Caso 1	0.175 s	0.177 s	0.180 s	0.193 s	0.185 s
Caso 2	0.232 s	0.208 s	0.226 s	0.219 s	0.221 s
Caso 3	0.326 s	0.295 s	0.291 s	0.287 s	0.310 s
Caso 4	0.362 s	0.357 s	0.376 s	0.322 s	0.381 s

Tabela 2. Desempenhos para casos do ficheiro dados2

C \ Op	normal	ff	ffc	min	max
Caso 1	1.456 s	1.451 s	1.446 s	0.984 s	1.433 s
Caso 2	1.285 s	1.303 s	1.294 s	0.605 s	1.304 s
Caso 3	15.128 s	15.366 s	15.435 s	6.330 s	15.355 s
Caso 4	63.816 s	65.328 s	65.570 s	37.261 s	64.959 s
Caso 5	126.76 s	130.35 s	130.39 s	91.500 s	129.93 s
Caso 6	128.51 s	129.54 s	132.32 s	27.752 s	130.09 s
Caso 7	442.08 s	449.31 s	445.74 s	83.798 s	453.49 s

Tabela 3. Desempenhos para casos do ficheiro dados3

Como se pode constatar, com a utilização da opção "*min*", obtêm-se resultados de desempenho significativamente mais baixos para os casos com mais dados, o que permite para estes casos de grandes dimensões obter um resultado em tempo útil.

Os resultados apresentados foram obtidos num computador com a seguinte configuração: CPU Intel core 2 Duo P8700 @ 2.53GHz, 4GB de RAM e Windows 7 64bits.

9 Conclusões e Perspectivas de Desevolvimento

Após análise da realização deste trabalho, concluiu-se que foram cumpridos os objectivos propostos. O programa mostra ser capaz de obter a solução mais eficiente para o problema proposto, obedecendo aos critérios impostos e maximizando o Lucro Semanal da escola de línguas. Note-se ainda que o desenvolvimento de um código genérico que permitisse resolver problemas com dimensões e restrições diferentes foi uma das prioridades, constituindo assim uma vantagem da aplicação.

A nível de desenvolvimento, este trabalho apresenta boas perspectivas, pois pode facilmente ser moldado para resolver problemas semelhantes, mesmo com critérios e restrições diferentes que se enquadrem em situações ainda mais reais complexas.

Por fim, este trabalho foi de grande importância para por em prática os conhecimentos teóricos leccionados nas aulas e que nos permitiu adquirir familiaridade com os paradigmas de programação em lógica com restrições bem como aprofundar a prática em linguagem Prolog.

Bibliografia

1. Swedish Institute of Computer Science. *SICStus Prolog User's Manual*. Outubro 2012. Disponível online em http://www.sics.se/isl/sicstus.html.

2. SICStus support. SICStus Prolog. http://www.sics.se/sicstus/docs/4.0.4/html/sicstus/Enumeration-Predicates.html Dezembro 2012.

3. Marriot, Kim. *Programming with constraints*. MIT Press, 1998.

4. Luís Paulo Reis, Henrique Lopes Cardoso. Materiais da Disciplina de Programação em Lógica, Dezembro 2012

Anexos

Código Fonte

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

% Ficheiros de teste.
% Descomentar o que se pretende utilizar:
:- compile('dados1.pl').
%:- compile('dados2.pl').
%:- compile('dados3.pl').

solve(Caso):-
          statistics(walltime,[Start,_]),
          nl, escola(Caso),
          statistics(walltime,[End,_]),
          Time is End - Start,nl,
          format('Solutions in ~3d seconds.~n',[Time]),nl,
          fd_statistics,nl.

escola(Caso):-
    length(Caso,N), length(HCurso,N), length(NProf,N),
    profs(ListaProfs), length(ListaProfs,NP),
    preco_prof(PrecoProf), % lista com vencimento por
hora de cada professor
    domain(HCurso,4,8), % lista com as horas de cada cur-
so (indice)
    domain(NProf,1,NP),  % lista c professor p cada curso
(indice)

    indisp_prof_curso(IndispProfCurso),
    sep_caso(Caso,[],IndCurso1,_TotalVaga),
    reverse(IndCurso1,IndCurso),
    aplica_prof(IndispProfCurso,IndCurso,NProf),

    verifica_nr_cursos_prof(NProf,NP),
    length(LucroPorCurso,N),
    lucro_por_curso(HCurso,NProf,Caso,PrecoProf,
[],LucroPorCurso),
```

```
    sum(LucroPorCurso,#=,LucroDosCursos), % LucroDosCur-
sos é o lucro total, ou seja, somatório do lucro de todos
os cursos

    max_hextra(MaxHEx),max_hpt(MaxHPt),
    HExtra in 0..MaxHEx, HPt in 0..MaxHPt,
    (N*15) #=< 2*40 + HExtra + HPt,
    CustoFunc #= (2*40*15 + HExtra*25 + HPt*10),

    LucroSemanal #= LucroDosCursos - CustoFunc,
    append(HCurso,NProf,L1),
    labeling([minimize(CustoFunc)],[HExtra,HPt]),
    labeling([min,maximize(LucroSemanal)],L1),

    write('Maximo Lucro Semanal: '), write(LucroSemanal),
nl, nl,
    write('Plano:'),nl,
    print_plano(HCurso,NProf,ListaProfs,IndCurso), nl,
    write(' Horas Extra Necessarias das Funcionarias
Administrativas: '), write(HExtra),nl,
    write(' Horas Necessarias de Funcionaria Part-time:
'), write(HPt),nl.



%IndCurso - lista dos índices dos cursos,
%TotalVaga - Somatório das vagas de tudos cursos
sep_caso([],IndCurso,IndCurso,0).
sep_caso([NCurso-Vaga|Resto],Acc,IndCurso,TotalVaga):-
    sep_caso(Resto,[NCurso|Acc],IndCurso,Acum),
    TotalVaga is Acum+Vaga.


% verifica se nr de cursos de cada professor está entre 1
e 2
verifica_nr_cursos_prof(_,0).
verifica_nr_cursos_prof(NProf,N):-
    count(N,NProf,#=,Count),Count#=1 #\ Count#=2,
    N1 is N-1,
    verifica_nr_cursos_prof(NProf,N1).

% predicado auxiliar de aplica_prof/3
aplica_prof_aux([],_).
aplica_prof_aux([IndispProf1|T],NProf1):-
    NProf1 #\= IndispProf1, aplica_prof_aux(T,NProf1).
```

```
% restringe, para cada curso, apenas os professores com
habilitações para os leccionar
aplica_prof(_IndispProfCurso,[],_NProf).
apli-
ca_prof(IndispProfCurso,[Ind1|Resto],[NProf1|NProfResto])
:-
    nth1(Ind1,IndispProfCurso,IndispProf),
    aplica_prof_aux(IndispProf,NProf1),
    aplica_prof(IndispProfCurso,Resto,NProfResto).

% Deixou de ser necessário:
/*
horas_por_professor(_,_,_HProf,0).
horas_por_professor(HCurso,NProf,HProf,NP):-
    horas_aux(HCurso,NProf,NP,Total),
    nth1(NP,HProf,Total),
    NP1 is NP-1,
    horas_por_professor(HCurso,NProf,HProf,NP1).
*/

horas_aux([],[],_,0).
horas_aux([HCurso1|HResto],[NProf1|NResto],NP,Total):-
    (NP #= NProf1) #<=> A,
horas_aux(HResto,NResto,NP,Acum),
    Total #= Acum+HCurso1*A.

% atribui, na lista LucroPorCurso, o lucro para cada cur-
so (excluindo ainda custos de manutenção)
lucro_por_curso([],[],[],_,LucroPorCurso,LucroPorCurso).
lucro_por_curso([HCurso1|HResto],[NProf1|NResto],[_-
Vaga|CResto],PrecoProf,Acc,LucroPorCurso):-
    nth1(NProf1,PrecoProf,Salario),
    L1 #= HCurso1*Vaga*10 - Salario*HCurso1,

  lucro_por_curso(HResto,NResto,CResto,PrecoProf,[L1|Acc]
,LucroPorCurso).

% imprime o plano para cada curso: Professor responsável
e número de horas semanais
print_plano([],[],_,_).
print_plano([H|Hs],[P|Ps],ListaProfs,[Ind1|Inds]):-
    write(' Curso de '), curso(Ind1,Curso), write(Curso),
nl,
    nth1(P,ListaProfs,NomeP),
```

```
    write('    Professor: '), write(NomeP), nl,
    write('    Numero Horas Semanais: '), write(H), nl,
    print_plano(Hs,Ps,ListaProfs,Inds).
```

Ficheiro de Teste dados1.pl

```
% Cursos =[Espanhol,Frances,Ingles,Italiano,Alemao,Russo]
%              1       2      3       4       5      6

profs(['Annete','Charles','Boris']).

curso(1,'Espanhol').
curso(2,'Frances').
curso(3,'Ingles').
curso(4,'Italiano').
curso(5,'Alemao').
curso(6,'Russo').

/*
curso_prof([1,2,5,7],    %Prof. Annette
       [1,3,4,6],    %Prof. Charles
       [1,2,4,6,7]). %Prof. Boris
*/

%prof_curso([[1,2,3],[1,3],[2],[2,3],[1],[2,3],[1,3]]).

indisp_prof_curso([[],[2],[1,3],[1],[2,3],[1],[2]]).

preco_prof([25,30,40]).

max_hextra(10).
max_hpt(10).

% Curso-Vaga
caso1([1-15,2-15,3-15]).
caso2([1-15,2-15,3-15,4-15,5-15,6-15]).
caso3([1-13,2-15,3-11,4-14]).
caso4([1-1,2-1,3-1,4-1,5-1]).
caso5([1-1,2-1,3-1,4-1,5-1,6-1]).
```

Ficheiro de Teste dados2.pl

```
% Cursos = [Espanhol,Frances,Ingles,Italiano,Alemao,
Russo,Portugues,Chines,Coreano]

profs(['Annete','Charles','Boris','Jorge','Miao']).

curso(1,'Espanhol').
curso(2,'Frances').
curso(3,'Ingles').
curso(4,'Italiano').
curso(5,'Alemao').
curso(6,'Russo').
curso(7,'Portugues').
curso(8,'Chines').
curso(9,'Coreano').

/*
curso_prof(  [1,2,5,7],    %Prof. Annette
             [1,3,4,6],    %Prof. Charles
             [1,2,4,6,7],  %Prof. Boris
             [1,3,7,9],    %Prof. Jorge
             [3,7,8,9]).   %Prof. Miao
*/

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

preco_prof([25,30,40,35,35]).

max_hextra(30).
max_hpt(30).

% Curso-Vaga
caso1([1-15,2-15,3-15,5-15,8-15]).
caso2([1-13,3-11,4-8,5-14,7-10,8-15]).
caso3([2-9,3-8,4-5,5-10,6-2,7-9,8-12,9-9]).
caso4([1-1,2-7,3-12,4-6,5-2,6-3,7-10,8-12,9-10]).
```

Ficheiro de Teste dados3.pl

```
profs(['Annete','Charles','Boris','Jorge','Miao','Mario',
'Alexandre','Afonso','Rui']).

curso(1,'Espanhol').
curso(2,'Frances').
curso(3,'Ingles').
curso(4,'Italiano').
curso(5,'Alemao').
curso(6,'Russo').
curso(7,'Portugues').
curso(8,'Chines').
curso(9,'Coreano').
curso(10,'Croata').
curso(11,'Finlandês').
curso(12,'Romeno').
curso(13,'Grego').
curso(14,'Turco').
curso(15,'Sueco').
curso(16,'Latim').

/*
curso_prof([1,3,5,12],                %Annete
           [1,3,4,6,9],               %Charles
           [2,4,8,10,11],             %Boris
           [5,13,15,16],              %Jorge
           [3,7,14,15,16],            %Miao
           [1,2,4,6,7,9,13],          %Mario
           [3,4,7,9,10,11],           %Alexandre
           [3,8,9,12,13,14,15],       %Afonso
           [2,5,6,8,10,11,12,14,16]). %Rui
*/

indisp_prof_curso([[3,4,5,7,8,9],
                   [1,2,4,5,7,8],
                   [3,4,6,9],
                   [1,4,5,8,9],
                   [2,3,5,6,7,8],
                   [1,3,4,5,7,8],
                   [1,2,3,4,8,9],
                   [1,2,4,5,6,7],
                   [1,3,4,5,9],
                   [1,2,4,5,6,8],
```

```
                    [1,2,4,5,6,8],
                    [2,3,4,5,6,7],
                    [1,2,3,5,7,9],
                    [1,2,3,4,6,7],
                    [1,2,3,6,7,9],
                    [1,2,3,6,7,8]]).

preco_prof([80,90,85,75,95,80,85,75,100]).

max_hextra(100).
max_hpt(60).

% Curso-Vaga
caso1([1-15,3-15,5-15,7-12,8-15,11-9,13-13,15-12,16-15]).
caso2([1-23,4-18,5-14,7-20,8-15,9-10,12-25,14-23,
15-20,16-22]).
caso3([2-19,3-18,4-15,5-10,7-19,8-12,9-9,11-6,12-12,
13-14,14-15]).
caso4([2-7,3-12,4-6,6-13,8-12,9-10,10-3,11-14,12-5,
13-6,15-8,16-9]).
caso5([2-7,3-12,4-6,6-13,8-12,9-10,10-3,11-14,12-5,
13-6,14-3,15-8,16-9]).
caso6([2-17,3-22,4-9,5-21,6-5,8-16,9-20,10-17,11-10,
12-15,13-19,14-23,15-18,16-10]).
caso7([1-15,2-11,3-25,4-19,5-13,6-15,8-15,9-17,10-12,
11-14,12-20,13-16,14-15,15-24,16-20]).
```