Grupos em Campo de Férias
Resolução de Problema de Decisão usando Programação em Lógica com Restrições

Joel Dinis, Grupos em Campo de Ferias 5, MIEIC03

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

Abstract. Este projeto insere-se no contexto da unidade curricular de Programação em Lógica usando restrições. O projeto consiste na formação de grupos com um determinado número de pessoas em atividades, sem que os grupos possuam pessoas repetidas ao longo das mesmas, e em que seja possível atribuir pelo menos uma pessoa com uma determinada característica a cada grupo formado.

Keywords: sicstus, prolog, feup, grupos

1 Introdução

Este projeto integra-se na unidade curricular de Programação em Lógica, usando restrições. Este projeto, em particular, debate-se sobre a construção de grupos para um determinado campo de férias. Neste campo de férias existem várias atividades, cada uma podendo requerer que cada grupo tenha pelo menos uma pessoa com capacidade para a executar. Para dar um exemplo prático, o grupo de férias poderá ter uma atividade que consista num jogo de futebol. Neste caso, é de esperar que cada um dos grupos tenha um jogador que seja bom a jogar futebol. Outra restrição presente no trabalho é a de, com o intuito de providenciar aos participantes uma experiência mais social, os grupos não devem ter elementos repetidos nas várias, isto é, por exemplo, se o Manuel ficou num grupo com a Maria e o João no jogo de futebol, já não deve ficar com eles nos próximos eventos para maximizar interações entre os participantes. Nas secções abaixo, irei explicar o problema, a abordagem que segui para lidar com o problema, bem como predicados de visualização e conclusões.

2 Descrição do Problema

O projeto consiste na resolução de um problema de otimização na linguagem Prolog. O objetivo é, neste caso, conseguir o maior número de elementos distintos entre os grupos, tentando assegurar a cada grupo pelo menos uma pessoa capaz de concluir a atividade. Claro que, na imagem não está explícito, mas em cada grupo de futebol, se possível, um ou mais elementos de cada grupo vão ter a

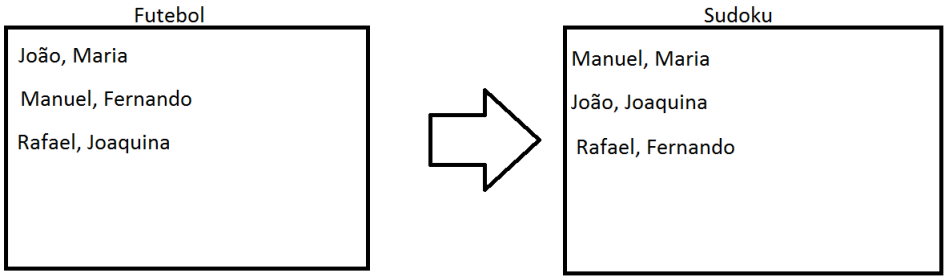

Fig. 1. Exemplo de grupos formados

habilidade de jogar futebol. Voltando atrás no exemplo dado, isto quer dizer, que no grupo do João e a Maria pelo menos um dos dois saberá jogar futebol. O mesmo acontece no grupo do Manuel e do Fernando em que pelo menos um saberá jogar futebol e assim sucessivamente. Isto, claro, referente à atividade de Jogo de Futebol. Mas, como um campo de férias tem mais atividades, neste caso, poderá ter uma atividade denominada de Sudoku por exemplo. Neste caso, e caso seja possível os grupos não deverão partilhar o mesmo grupo outra vez. Voltando ao exemplo para concretizar esta restrição, isto significa que o João não deve voltar a ficar com a Maria no mesmo grupo e vice versa. O mesmo se aplica ao Manuel e ao Fernando e assim consecutivamente. Isto, tentando que em cada novo grupo formado haja pelo menos um jogador que saiba jogar sudoku.

3 Abordagem

3.1 Variáveis de Decisão

Sendo assim, a solução poderá ser representada por uma lista simples com um tamanho de número de participantes vezes o número de atividades em que cada elemento pode representar o id de um participante e a ordem o grupo do qual participam. Nesta representação, em cada N participantes estariam representados os grupos a formar, para uma determinada atividade. Denominando por **N** o número de participantes e **S** o número de atividades, o **tamanho da solução** será representada pela seguinte fórmula:

$$TamanhoLista = N * S \tag{1}$$

O **domínio** é definido pelo número de participantes de 1 a **N**. Abaixo é dado um exemplo desta representação.

participante(Id,Nome,Habilidade).
participante(1,'Joao','futebol').
participante(2,'Maria','futebol').
participante(3,'Filipe','futebol').
participante(4,'Henrique','forte').

Na representação acima, cada participante é identificado com um Id, Nome, e Habilidade especial que possa ter. Neste caso, o João, a Maria e o Filipe jogam futebol bem, enquanto que o Henrique é forte.

atividade(Id,Nome,Número de Grupos,Habilidade).
atividade(1,'refeicao',2,'futebol').
atividade(2,'jogo de futebol',2,'forte').

Acima representado, está a forma de representação de uma atividade. Cada atividade tem um Id, um Nome, o número de grupos a formar, e o requerimento especial que esta atividade possa ter. Neste caso, e pegando num exemplo caricato imagina-se que à refeição se vão formar 4 grupos e que cada um deles terá alguém(ou pelo menos tentará) que saiba jogar futebol. Na atividade 2, tal como está representado, atividade jogo de futebol, irão fazer-se 2 grupos, em que se vai tentar que ambos tenham pelo menos uma pessoa forte.

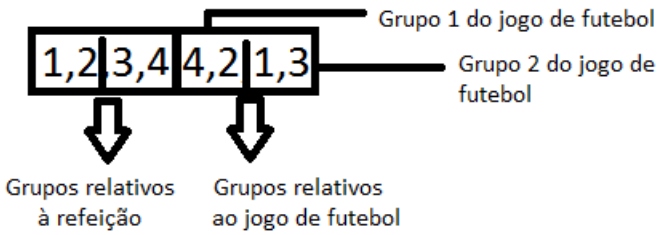

Fig. 2. Exemplo de grupos formados

Esta solução representada poderá correspnder ao problema apresentado acima, em que para a refeição as pessoas com o Id 1 e 2(João e Maria) ficam juntas no Grupo 1, e as pessoas com o Id 3 e 4(Filipe e Henrique) ficam juntas no Grupo 2. Neste caso, e por existirem pessoas suficientes que saibam jogar futebol é garantida a restrição de em cada um dos grupos formados haver alguém que saiba jogar futebol. Já na atividade jogo de futebol, onde se pretendia que houvesse pelo menos uma pessoa forte em cada equipa não foi satisfeita, por não haver número suficiente de pessoas com esta habilidade. Contudo, a restrição de não

as pessoas não se podere repetir foi satisfeita já que esta atividade possui como seus elementos no Grupo 1 as pessoas com o Id 4 e 2(Henrique e Maria) e no Grupo 2 os Ids 1 e 3(Joao e Filipe). A lista tem um tamanho igual ao **N** x **S** = 4 x 2 = 8, como já se havia explicado anteriormente. O **domínio** de N em N é de 1 a N, sendo que os Ids não se podem repetir. Em baixo fica um excerto do código que assegura o domínio das variáveis.

```
participant(L):- findall(Id,participante(Id,_,_),L).
ativities(L):- findall(Id,atividade(Id,_,_,_),L).

ndifferent(_,_,N,N).
ndifferent(L,N,Count,N3):-sublist(L,S,Count,N),all_different(S),Count1
    is Count + N,ndifferent(L,N,Count1,N3).

solver(L):- participant(P),
ativities(A),
length(P,N),
length(A,N2),
N3 #= N2 * N,
length(L,N3),
domain(L,1,N),
ndifferent(L,N,0,N3),
(...).
```

3.2 Restrições

Tal como já foi referido anteriormente, as restrições deste projeto são bastante simples de perceber. Nas várias atividades que o campo de férias irá desenvolver não deve haver repetições de conjuntos de estudantes, sendo que estas repetições deverão ser minimizadas, e em cada um dos grupos formados para uma determinada atividade, pelo menos uma pessoa deverá ter o requerimento exigido pela atividade. Para assegurar estas condições vários predicados foram construídos. No caso da primeira restrição, e para garantir que não haja repetições o algoritmo apresentado basicamente vai comparar o grupo da atividade S, a todos os grupos da atividade S + I, sendo que I é maior que 0, usando predicados do Sicstus, nomeadamente o *append* e o *nvalue*. É, essencialmente, feito um append entre o grupo a comparar com todos os outros e obriga-se a que o nvalue deste append seja de 1 à soma das duas listas(situação onde todos os elementos são distintos). Vai-se somando o valor destes nvalues de comparação a comparação e, no final, no labeling maximiza-se esta soma. Há, contudo, vários casos a ter atenção, nomeadamente quando o número de participantes não é divisível pelo tamanho imposto ao grupo. Abaixo fica um excerto que trata quer das restrições das repetições, quer da restrição da habilidade por grupo.

```
iterador(L,S,Iterador,Event,N,Sum,Sum,CurrentEvent):- Iterador #>= Event
    * N.

iterador(L,S,Iterador,Event,N,Sum,Sum,CurrentEvent):-
    length(S,P),atividade(Event,_,Largura,_),Checker is Iterador +
    Largura ,Checker > Event*N,NewLargura is Checker -
    (Event*N),sublist(L,S2,Iterador,NewLargura),atividade(Event,_,_,Habilidade),
participante(Id,_,Habilidade),member(Id,S2).

iterador(L,S,Iterador,Event,N,Sum,Sum,CurrentEvent):-
    length(S,P),atividade(Event,_,Largura,_),Checker is Iterador +
    Largura ,Checker > Event*N,NewLargura is Checker - (Event*N).

iterador(L,S,Iterador,Event,N,I,Sum,CurrentEvent):-
    length(S,P),atividade(Event,_,Largura,_),
    sublist(L,S2,Iterador,Largura),NovoIterador is Iterador +
    Largura,length(S2,P2),append(S,S2,ListaFinal),Temp is P + P2 ,Temp2
    is P + P2 -
    1,atividade(CurrentEvent,_,_,Habilidade),participante(Id,_,Habilidade),member(Id,S),
atividade(Event,_,_,Habilidade2),participante(Id2,_,Habilidade2),member(Id2,S2),Size
    in Temp2..Temp,nvalue(Size, ListaFinal),Size2 #= Size +
    I,iterador(L,S,NovoIterador,Event,N,Size2,Sum,CurrentEvent).

iterador(L,S,Iterador,Event,N,I,Sum,CurrentEvent):-
    length(S,P),atividade(Event,_,Largura,_),
    sublist(L,S2,Iterador,Largura),NovoIterador is Iterador +
    Largura,length(S2,P2),append(S,S2,ListaFinal),Temp is P + P2 ,Temp2
    is P + P2 - 1,nl,Size in Temp2..Temp,nvalue(Size, ListaFinal),Size2
    #= Size +
    I,iterador(L,S,NovoIterador,Event,N,Size2,Sum,CurrentEvent).
```

3.3 Função de Avaliação

Basicamente, para conseguir obter a solução ótima é executado um maximize da soma ao labeling. Já para garantir que cada grupo tem uma pelo menos uma pessoa com uma habilidade especial, é executado

```
atividade(CurrentEvent,_,_,Habilidade),participante(Id,_,Habilidade),member(Id,S),
    (...)
```

Se não for possível, o mesmo predicado é executado, contudo sem a parte acima mencionada.

3.4 Estratégia de Pesquisa

```
(...),
sortbyrepetition(L,0,N,1,1,0,0,Sum),
labeling([maximize(Sum),all],L),
```

Por se comparar um grupo a todos os grupos, criei uma variável Soma que basicamente graças ao labeling é maximizada, por ideia do professor. Contudo, esta ideia não se revelou eficaz, pelo motivo de que uma maior soma nem sempre fornece a melhor solução. Passando a exemplificar:

```
--------------------------------------------------------------------------------
--------          Groups for Event: refeicao
--------------------------------------------------------------------------------
--------
Group 1
Joao-futebol
Henrique-forte
--------------------------------------------------------------------------------
--------
Group 2
Maria-futebol
Bond-forte
--------------------------------------------------------------------------------
--------
Group 3
Filipe-futebol
Gordon Ramsey-cozinha
--------------------------------------------------------------------------------
--------
Group 4
Joel-cozinha
Pilinha-cozinha
--------------------------------------------------------------------------------
--------          Groups for Event: jogo de futebol
--------------------------------------------------------------------------------
--------
Group 1
Henrique-forte
Joao-futebol
--------------------------------------------------------------------------------
--------
Group 2
Bond-forte
Maria-futebol
--------------------------------------------------------------------------------
--------
Group 3
Filipe-futebol
Joel-cozinha
--------------------------------------------------------------------------------
--------
Group 4
Gordon Ramsey-cozinha
Pilinha-cozinha
--------------------------------------------------------------------------------
```

Fig. 3. Exemplo

Fazendo as contas da Soma, seria: (2+4+4+4)+(4+2+4+4)+(4+4+3+3)+(4+4+3+3) = 56 Contudo, como é possível constatar existem ainda repetições nos grupos.
A solução ótima seria a apresentada na figura abaixo.

```
--------------------------------------------------------------------------------
--------          Groups for Event: refeicao
--------------------------------------------------------------------------------
--------
Group 1
Joao-futebol
Henrique-forte
--------------------------------------------------------------------------------
--------
Group 2
Maria-futebol
Bond-forte
--------------------------------------------------------------------------------
--------
Group 3
Filipe-futebol
Gordon Ramsey-cozinha
--------------------------------------------------------------------------------
--------
Group 4
Joel-cozinha
Pilinha-cozinha
--------------------------------------------------------------------------------
--------          Groups for Event: jogo de futebol
--------------------------------------------------------------------------------
--------
Group 1
Henrique-forte
Maria-futebol
--------------------------------------------------------------------------------
--------
Group 2
Bond-forte
Joao-futebol
--------------------------------------------------------------------------------
--------
Group 3
Filipe-futebol
Joel-cozinha
--------------------------------------------------------------------------------
--------
Group 4
Gordon Ramsey-cozinha
Pilinha-cozinha
--------------------------------------------------------------------------------
```

Fig. 4. Exemplo

Contudo, fazendo as contas vem: (3+3+4+4)+(3+3+4+4)+(4+4+3+3)+(4+4+3+3) = 56. Sendo assim, não é exequível confiar na soma, pois pode não dar a solução ótima.

4 Visualização da solução

O predicado de visualização da solução é bastante simples na medida em que, tendo a lista com os valores preenchidos pelo labeling, e dada a largura de cada grupo **L**, basta separar de L em L e, quando L for igual ou maior que N, aí mudar de atividade.

```
printList(L,TamanhoGrupo,NumeroElem,N2,N,NumGrupo,NumeroElem):- N =<
    NumeroElem.

printList(L,TamanhoGrupo,NumeroElem,-1,N,NumGrupo,NumeroFinal):- N >=
    NumeroElem,write('---------------------------------------
-------------------------------------------------\n'),
write('Group '),write(NumGrupo),nl,NumGrupo1 is NumGrupo +
    1,printList(L,TamanhoGrupo,NumeroElem,0,N,NumGrupo1,NumeroFinal),!.

printList(L,TamanhoGrupo,NumeroElem,TamanhoGrupo,N,NumGrupo,NumeroFinal):-N
    >=
    NumeroElem,printList(L,TamanhoGrupo,NumeroElem,-1,N,NumGrupo,NumeroFinal),!.

printList([L|Resto],TamanhoGrupo,NumeroElem,N2,N,NumGrupo,NumeroFinal):-
    NumeroElem1 is NumeroElem + 1,N3 is N2 + 1,N >= NumeroElem1,Elem is
    L,participante(Elem,Nome,Habilidade),write(Nome-Habilidade),nl,
printList(Resto,TamanhoGrupo,NumeroElem1,N3,N,NumGrupo,NumeroFinal),!.

printGroup(_,_,NumeroEvento,_,NumeroEvento).

printGroup(L,N,NumeroEvento,NumeroElem,N2):- NumeroEvento1 is
    NumeroEvento + 1,write('-----------------------------------------
-------------------------------------------    Groups for Event:
    '),atividade(NumeroEvento1,Nome,Tamanho,_),write(Nome),write('\n'),
printList(L,Tamanho,NumeroElem,-1,N,1,NumeroFinal),length(L3,N),
append(L3,L2,L),printGroup(L2,N,NumeroEvento1,0,N2),!.
```

5 Resultados

Em geral, os resultados são bastante bons. De todas as combinações com 8 pessoas, o programa corre bastante bem, e não demora muito tempo. Deixo aqui alguns gráficos ilustrativos. De resto, o próprio programa apresenta estatísticas em *real time*, ou seja que dá para correr e ver as estatísticas depois de executado.

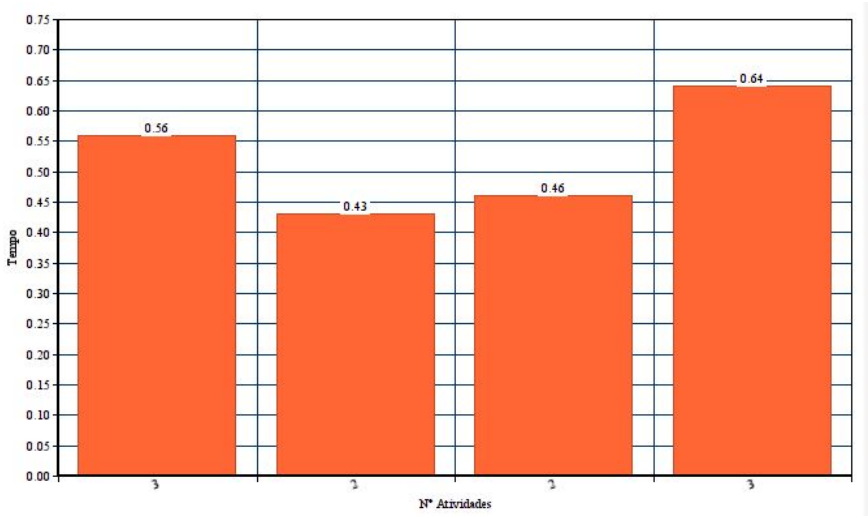

Fig. 5. Gráfico relação tempo/número de atividades

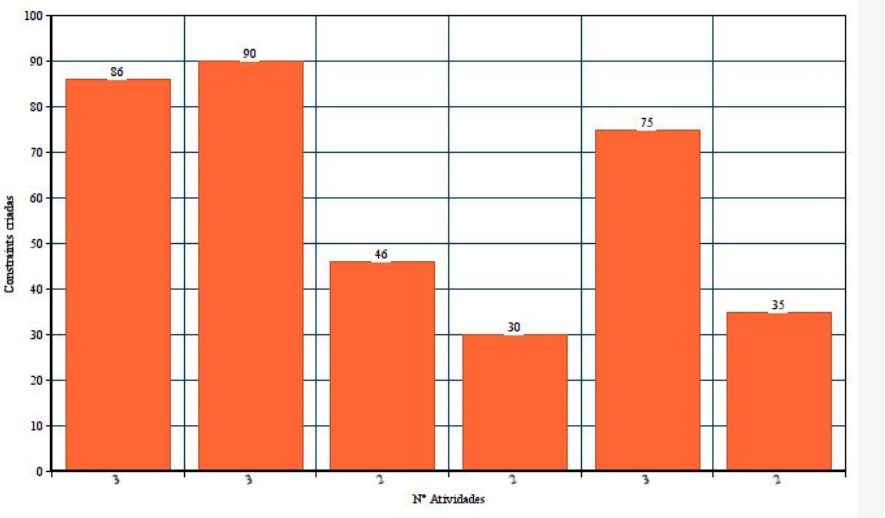

Fig. 6. Gráfico relação/constraints criadas

6 Conclusão

É de concluir o sucesso do projeto, que consegue alcançar uma solução para problemas diários num espaço de tempo relativamente curto. É um projeto com alguma utilidade, capaz de ser utilizado em problemas reais. É de concluir também

a utilidade da linguagem Prolog, que com o uso correcto se pode revelar um aliado forte na resolução de problemas. Contudo, é preciso ter cuidado com a sua utilização, pois se for incorretamente utilizada, pode criar soluções demasiado complexas para problemas simples.

References

SICStus Prolog, `https://sicstus.sics.se//`
SWI-Prolog, `www.swi-prolog.org//`
Stack Overflow, `https://pt.stackoverflow.com//`

Anexo

Código Fonte

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

%Base de Dados sobre atividades
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
atividade(1,'refeicao',3,'futebol').
atividade(2,'jogo de futebol',2,'forte').
%atividade(3,'brainstorm',3,'cozinha').

%atividade('workshop de madeira').
%atividade('sudoku challenge').
%atividade('aula de cozinha').

%Base de Dados sobre participantes
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
participante(1,'Joao','futebol').
participante(2,'Maria','futebol').
participante(3,'Filipe','futebol').
participante(4,'Henrique','forte').
participante(5,'Bond','forte').
participante(6,'Gordon Ramsey','cozinha').
participante(7,'Joel','cozinha').
participante(8,'Pilinha','cozinha').

participant(L):- findall(Id,participante(Id,_,_),L).
atividade(L):- findall(Id,atividade(Id,_,_,_),L).

%Restricao que garante que a lista de 1 a N tem IDs de participantes
    diferentes
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ndifferent(_,_,N,N).
ndifferent(L,N,Count,N3):-sublist(L,S,Count,N),all_different(S),Count1
    is Count + N,ndifferent(L,N,Count1,N3).

%printList(L,TamanhoGrupo,TamanhoGrupo,L,N,NumeroElem,NumeroElem):-nl,nl.
%printList([L|Resto],Contador,TamanhoGrupo,L2,N,NumeroElem,R):-
    NumeroElem1 is NumeroElem + 1,Contador1 is Contador + 1, NumeroElem1
    =< N,
    write(NumeroElem1),participante(L,Nome,_,_,Age,_),write(Nome-Age),nl,
%printList(Resto,Contador1,TamanhoGrupo,L2,N,NumeroElem1,R),!.
%printList(L,Contador,TamanhoGrupo,L2,N,NumeroElem,R):- NumeroElem1 is
    NumeroElem + 1,Contador1 is Contador +
    1,write(NumeroElem1),printList(L,Contador1,TamanhoGrupo,L2,N,NumeroElem1,R),!.
```

```
%printGroup([],_,_,_,_,_,_).

%printGroup(L,0,TamanhoGrupo,1,N3,N,NumeroElem):-
%write('-----------------------------------------------------------------------------------------
    Groups for Event:
    '),atividade(NumeroEvento,Nome),write(Nome),write('\n'),
%write('Group 1'),nl,nl,N4 is N3 + TamanhoGrupo,
%printList(L,0,TamanhoGrupo,L2,N,NumeroElem,R),printGroup(L2,1,TamanhoGrupo,1,N4,N,R).

%printGroup(L,Contador,TamanhoGrupo,NumeroEvento,N,N,NumeroElem):-NumeroEvento1
    is NumeroEvento + 1,
%write('-----------------------------------------------------------------------------------------
    Groups for Event:
    '),atividade(NumeroEvento1,Nome),write(Nome),write('\n'),
%write('Group 1'),nl,nl,
%printList(L,0,TamanhoGrupo,L2,N,0,R),printGroup(L2,1,TamanhoGrupo,NumeroEvento1,TamanhoGrupo,N,R).

%printGroup(L,Contador,TamanhoGrupo,NumeroEvento,N3,N,NumeroElem):-
%write('-----------------------------------------------------------------------------------------\n'),
%Contador1 is Contador + 1,write('Group '),write(Contador1),nl,nl,
%N4 is N3 + TamanhoGrupo,
%printList(L,0,TamanhoGrupo,L2,N,NumeroElem,R),
%printGroup(L2,Contador1,TamanhoGrupo,NumeroEvento,N4,N,R).

%sortbyrepetition(L,N,N,Event):-

%sortbyrepetition(L,N,N,_).
%sortbyrepetition(L,Index,N,Event):-
    atividade(Event,_,Largura),sublist(L,S,Index,Largura),Index3 is
    Index + Largura,length(S,P),Index2 is Index + N,Event2 is Event +
    1,atividade(Event2,_,Largura2),sublist(L,S1,Index2,Largura2),append(S,S1,Final),
%length(S1,P2),PDiff #= P + P2 -
    1,nvalue(PDiff,Final),sortbyrepetition(L,Index3,N,Event).

%iterador(L,S,N,Event,Final):- Event1 is Event + 1,N is Final * Event1.
%iterador(L,S,Iterador,Event,N):-
    length(S,P),atividade(Event,_,Largura),sublist(L,S2,Iterador,Largura),NovoIterador
    is Iterador + Largura,length(S2,P2),append(S,S2,ListaFinal),Size #=
    P + P2 - 1,nvalue(Size,
    ListaFinal),iterador(L,S,NovoIterador,Event,N).

%sortbyrepetition(L,Index,N,Event):-
    atividade(Event,_,Largura),sublist(L,S,Index,Largura),Event1 is
    Event + 1,Iterador is Event1 +
    N,iterador(L,S,Iterador,Event1,N).%,sortbyrepetition(L,Index,N,Event1).


%Predicado que recebe uma lista e compara a mesma com todas as outras de
    um indice ate ao fim dos indices
```

```
%garantindo a restricao de pessoas nao se repetirem em atividades
    diferentes.
%Tenta ainda colocar em cada grupo, uma pessoa com habilidade para
    concluir a tarefa dessa atividade com sucesso
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

iterador(L,S,Iterador,Event,N,Sum,Sum,CurrentEvent):- Iterador #>= Event
    * N.
iterador(L,S,Iterador,Event,N,Sum,Sum,CurrentEvent):-
    length(S,P),atividade(Event,_,Largura,_),Checker is Iterador +
    Largura ,Checker > Event*N,NewLargura is Checker -
    (Event*N),sublist(L,S2,Iterador,NewLargura),atividade(Event,_,_,Habilidade),
participante(Id,_,Habilidade),member(Id,S2).%,write(NewLargura),
    sublist(L,S2,Iterador,NewLargura),NovoIterador is Iterador +
    NewLargura,length(S2,P2),append(S,S2,ListaFinal),Size #= P + P2 - 1
    #\/ Size #= P + P2 ,nvalue(Size, ListaFinal).
iterador(L,S,Iterador,Event,N,Sum,Sum,CurrentEvent):-
    length(S,P),atividade(Event,_,Largura,_),Checker is Iterador +
    Largura ,Checker > Event*N,NewLargura is Checker -
    (Event*N).%,write(NewLargura),
    sublist(L,S2,Iterador,NewLargura),NovoIterador is Iterador +
    NewLargura,length(S2,P2),append(S,S2,ListaFinal),Size #= P + P2 - 1
    #\/ Size #= P + P2 ,nvalue(Size, ListaFinal).

iterador(L,S,Iterador,Event,N,I,Sum,CurrentEvent):-
    length(S,P),atividade(Event,_,Largura,_),
    sublist(L,S2,Iterador,Largura),NovoIterador is Iterador +
    Largura,length(S2,P2),append(S,S2,ListaFinal),Temp is P + P2 ,Temp2
    is P + P2 -
    1,atividade(CurrentEvent,_,_,Habilidade),participante(Id,_,Habilidade),member(Id,S),
atividade(Event,_,_,Habilidade2),participante(Id2,_,Habilidade2),member(Id2,S2),nl,Size
    in Temp2..Temp,nvalue(Size, ListaFinal),Size2 #= Size +
    I,iterador(L,S,NovoIterador,Event,N,Size2,Sum,CurrentEvent).
iterador(L,S,Iterador,Event,N,I,Sum,CurrentEvent):-
    length(S,P),atividade(Event,_,Largura,_),
    sublist(L,S2,Iterador,Largura),NovoIterador is Iterador +
    Largura,length(S2,P2),append(S,S2,ListaFinal),Temp is P + P2 ,Temp2
    is P + P2 - 1,Size in Temp2..Temp,nvalue(Size, ListaFinal),Size2 #=
    Size + I,iterador(L,S,NovoIterador,Event,N,Size2,Sum,CurrentEvent).

%sortbyrepetition(L,N,N,Event,CurrentEvent,Index2):-
    ativities(A),length(A,Num),CurrentEvent2 is CurrentEvent + 1,Num =
    CurrentEvent2..

% Predicados que tratam de selecionar as sublistas corretas para
    comparacao com as outras.
% Ter em atencao que este predicado e muito extenso pelo facto de
    haverem multiplas combinacoes
% para os grupos ja que a largura dos grupos pode ser ou nao multipla do
    numero de participantes
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sortbyrepetition(L,Index,N,Event,CurrentEvent,Index2,I,Sum):-
    atividades(A),length(A,Num),CurrentEvent = Num,Temp is CurrentEvent -
    1,atividade(Temp,_,Largura,_),Event1 is Event -
    1,sublist(L,S,Index2,Largura),Iterador is Temp *
    N,iterador(L,S,Iterador,Event1,N,I,T,CurrentEvent),Sum #= T.

sortbyrepetition(L,Index,N,Event,CurrentEvent,Index2,I,Sum):-
    atividades(A),length(A,Num),Num = 1.

sortbyrepetition(L,Index,N,Event,CurrentEvent,Index2,I,Sum):-
    atividade(Event,_,Largura,_),mod(N,Largura) =:= 0,Index2 \=
    0,atividades(A),length(A,Num),Temp is Num*N , Temp - Index2 =:= N.

%sortbyrepetition(L,Index,N,Event,CurrentEvent,Index2):-
    atividades(A),length(A,Num),Num =
    Event,atividade(CurrentEvent,_,Largura), mod(N,Largura) \= 0,
    CurrentEvent2 is CurrentEvent +
    1,atividade(CurrentEvent2,_,Largura),sublist(L,S,Index2,Largura),Event1
    is CurrentEvent2 + 1,Iterador is Index + N*CurrentEvent2
    ,iterador(L,S,Iterador,Event1,N),write(Iterador),
%sortbyrepetition(L,Iterador,N,Event1,CurrentEvent2,Index2).

sortbyrepetition(L,Index,N,Event,CurrentEvent,Index2,I,Sum):-
    atividades(A),length(A,Num),Num =
    Event,atividade(CurrentEvent,_,Largura,_),mod(N,Largura) =:=
    0,Index3 is Index2 + Largura
    ,sortbyrepetition(L,0,N,CurrentEvent,CurrentEvent,Index3,I,Sum).
%sortbyrepetition(L,Index,N,Event,CurrentEvent,Index2):-
    atividades(A),length(A,Num),Num =
    Event,write('hhhhhh'),atividade(CurrentEvent,_,Largura),mod(N,Largura)
    \= 0,Index3 is Index2 +
    Largura,sortbyrepetition(L,0,N,CurrentEvent,CurrentEvent,Index3).
%sortbyrepetition(L,Index,N,Event,CurrentEvent,Index2):-
    atividades(A),length(A,Num),Num =
    Event,write('hhhhhh'),atividade(CurrentEvent,_,Largura),mod(N,Largura)
    \= 0,mod(N,Index2) \= 0,Index3 is Index2 + Largura, (( Index3 >
    N*CurrentEvent -> Index4 is N*CurrentEvent);(Index4 is Index2 +
    Largura)) ,sortbyrepetition(L,0,N,CurrentEvent,CurrentEvent,Index4).
sortbyrepetition(L,Index,N,Event,CurrentEvent,Index2,I,Sum):-
    atividades(A),length(A,Num),Num =
    Event,atividade(CurrentEvent,_,Largura,_),mod(N,Largura) \=
    0,mod(N,Index2) \= 0,Index3 is Index2 + Largura, (( Index2 <
    N*(CurrentEvent-1) -> Index4 is N*(CurrentEvent-1));(Index4 is
    Index2 + Largura))
    ,sortbyrepetition(L,0,N,CurrentEvent,CurrentEvent,Index4,I,Sum),!.
```

```
%sortbyrepetition(L,Index,N,Event,CurrentEvent,Index2):- Index2 \=
    0,mod(Index2,N) =:= 0,CurrentEvent2 is CurrentEvent +
    1,atividade(CurrentEvent2,_,Largura),sublist(L,S,Index2,Largura),Event1
    is CurrentEvent2 + 1,Iterador is N*CurrentEvent2
    ,iterador(L,S,Iterador,Event1,N),write(Iterador),
%sortbyrepetition(L,Iterador,N,Event1,CurrentEvent2,Index2).

sortbyrepetition(L,Index,N,Event,CurrentEvent,Index2,I,Sum):- Index2 \=
    0, Index2 <
    CurrentEvent*N,atividade(CurrentEvent,_,Largura,_),Index2 + Largura
    >= N * CurrentEvent,NovaLargura is N*CurrentEvent -
    Index2,sublist(L,S,Index2,NovaLargura),atividade(CurrentEvent,_,_,Habilidade),
participante(Id,_,Habilidade),member(Id,S),CurrentEvent2 is CurrentEvent
    + 1,Event1 is CurrentEvent2 + 1,Iterador is Index + N*CurrentEvent2
    ,sortbyrepetition(L,Iterador,N,Event1,CurrentEvent2,Index2,I,Sum),!.
sortbyrepetition(L,Index,N,Event,CurrentEvent,Index2,I,Sum):- Index2 \=
    0, Index2 <
    CurrentEvent*N,atividade(CurrentEvent,_,Largura,_),Index2 + Largura
    >= N * CurrentEvent,NovaLargura is N*CurrentEvent -
    Index2,sublist(L,S,Index2,NovaLargura),CurrentEvent2 is CurrentEvent
    + 1,Event1 is CurrentEvent2 + 1,Iterador is Index + N*CurrentEvent2
    ,sortbyrepetition(L,Iterador,N,Event1,CurrentEvent2,Index2,I,Sum),!.


sortbyrepetition(L,Index,N,Event,CurrentEvent,Index2,I,Sum):-
    ativities(A),length(A,Num),CurrentEvent2 is CurrentEvent + 1,Num =
    CurrentEvent2,atividade(CurrentEvent,_,Largura,_), \+
    sublist(L,S,Index2,Largura).

sortbyrepetition(L,Index,N,Event,CurrentEvent,Index2,I,Sum):-
    atividade(CurrentEvent,_,Largura,_),Index2 + Largura =< N *
    CurrentEvent,sublist(L,S,Index2,Largura),Event1 is Event +
    1,Iterador is Index + N*CurrentEvent
    ,iterador(L,S,Iterador,Event1,N,I,T,CurrentEvent),
sortbyrepetition(L,Iterador,N,Event1,CurrentEvent,Index2,T,Sum),!.
sortbyrepetition(L,Index,N,Event,CurrentEvent,Index2,I,Sum):-
    atividade(CurrentEvent,_,Largura,_),Index2 + Largura > N *
    CurrentEvent,Iterador is Index + N*CurrentEvent,Event1 is Event +
    1,sortbyrepetition(L,Iterador,N,Event1,CurrentEvent,Index2,I,Sum).

sortbyrepetition(L,Index,N,Event,CurrentEvent,Index2,I,Sum):- Index2 \=
    0,CurrentEvent2 is CurrentEvent +
    1,atividade(CurrentEvent2,_,Largura,_),sublist(L,S,Index2,Largura),Event1
    is CurrentEvent2 + 1,Iterador is Index + N*CurrentEvent2
    ,iterador(L,S,Iterador,Event1,N,I,Sume,CurrentEvent),
sortbyrepetition(L,Iterador,N,Event1,CurrentEvent2,Index2,Sume,Sum).

%,Temp is Iterador + N ,Event2 is Event + 2,iterador(L,S,Temp,Event2,N).%
```

```
%sortbyrepetition(L,Index,N,Event,CurrentEvent,Index2,I,Sum):- Index2 \=
    0,ativities(A),length(A,Num),Temp is Num*N ,Ajust is mod(N,Index2),N
    \= Ajust, Temp - Index2 =:= N + Ajust.

%Imprime informacao no monitor
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
printList(L,TamanhoGrupo,NumeroElem,N2,N,NumGrupo,NumeroElem):- N =<
    NumeroElem.

printList(L,TamanhoGrupo,NumeroElem,-1,N,NumGrupo,NumeroFinal):- N >=
    NumeroElem,write('----------------------------------------------------------------------------------
write('Group '),write(NumGrupo),nl,NumGrupo1 is NumGrupo +
    1,printList(L,TamanhoGrupo,NumeroElem,0,N,NumGrupo1,NumeroFinal),!.

printList(L,TamanhoGrupo,NumeroElem,TamanhoGrupo,N,NumGrupo,NumeroFinal):-N
    >=
    NumeroElem,printList(L,TamanhoGrupo,NumeroElem,-1,N,NumGrupo,NumeroFinal),!.

printList([L|Resto],TamanhoGrupo,NumeroElem,N2,N,NumGrupo,NumeroFinal):-
    NumeroElem1 is NumeroElem + 1,N3 is N2 + 1,N >= NumeroElem1,Elem is
    L,participante(Elem,Nome,Habilidade),write(Nome-Habilidade),nl,
printList(Resto,TamanhoGrupo,NumeroElem1,N3,N,NumGrupo,NumeroFinal),!.

printGroup(_,_,NumeroEvento,_,NumeroEvento).

printGroup(L,N,NumeroEvento,NumeroElem,N2):- NumeroEvento1 is
    NumeroEvento +
    1,write('------------------------------------------------------------------------------------
    Groups for Event:
    '),atividade(NumeroEvento1,Nome,Tamanho,_),write(Nome),write('\n'),
printList(L,Tamanho,NumeroElem,-1,N,1,NumeroFinal),length(L3,N),append(L3,L2,L)
,printGroup(L2,N,NumeroEvento1,0,N2),!.

%Predicados de estatistica
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
reset_timer :- statistics(walltime,_).

print_time :-

       statistics(walltime,[_,T]),

       TS is ((T//10)*10)/1000,

       nl, write('Time: '), write(TS), write('s'), nl, nl.

%Parte inicial do programa
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
solver(L):- participant(P),
ativities(A),
length(P,N),
```

```
length(A,N2),
N3 #= N2 * N,
length(L,N3),
domain(L,1,N),
ndifferent(L,N,0,N3),
sortbyrepetition(L,0,N,1,1,0,0,Sum),
reset_timer,
labeling([maximize(Sum),all],L),nl,
%write(Sum),nl,
printGroup(L,N,0,0,N2),nl,
write('### STATISTICS ###'),nl,
    fd_statistics,print_time,nl.
```