Constraint Satisfaction and Optimization Problem: Choosing Speakers for a Diversified Topic Conference

Carolina Centeio Jorge (up201403090) and Tiago Almeida (up20130565)

FEUP-PLOG, 3MIEIC03,
Oradores Convidados 1

Abstract. *This project aims to find the better set of speakers for a certain conference. The goal is to maximize the difference(difference or distance?) between topics (each speaker has a topic associated) considering some money, gender and country constraints. Since it is a Constraint Satisfaction and Optimization Problem, we used SICStus Prolog and the clp(FD) module.*

Keywords: CSOP, Prolog, clpfd, clp, constraint, logic programming, optimization

1 Introduction

In the scope of the course PLOG, we were asked to implement a solution of a Constraint Satisfaction and Optimization Problem. Therefore, we chose Invited Speakers as subject to our project. This problem is similar to the Production Management problem, since we have to fulfill several constraints on the speakers diversity, but we need to maximize the difference*(difference or distance?)* amongst topics in the conference.
In this paper we will describe the problem in detail and its approach: decision variables (their domains and constraints), evaluation function and search strategy. In the end, we will present our solution, results and analyze them.

2 Problem Description

For a certain conference (that will take D days and T topics), there is the need to choose topics and speakers to talk about these topics. Each speaker has a name, a gender, a topic, a country, an travel expense and accommodation type associated.
There is a budget that cannot be exceeded. This budget will be used to pay speakers travel and accommodation expenses. The travel expense depends on the speaker and so does the accommodation, since one can choose to stay just for one night or for all conference (D nights).

Also, the speakers have to be equally distributed in gender and there cannot be two speakers coming from the same country.
All pairs of topics have a defined distance. The goal is to choose the speakers that maximize this distance amongst all topics in the conference.

3 Approach

To approach*(Doesn't sound good, I think a simple solve would sound better)* this problem, we create several lists with the speakers information. The lists Speakers, Topics, Countries, Gen, Desloc and Aloj have the information about names, topics, countries, gender, travel expense and type of accommodation of all possible speakers. We also create the list Invited, which contains the decision variables and the table of distances between all pairs of topics (Dist). The index determines the speaker in every list.

3.1 Decision Variables

In this project, the decision variables are the Boolean values for each speaker, that determines whether this is going to be chosen or not. Therefore, there are as many variables as possible speakers and their domain is Boolean (0 or 1). Ex: Invited = [I1,I2,I3], I1, I2 and I3 are the decision variables. I1 can be 1 or 0 meaning the speaker 1 is or is not going.

3.2 Constraints

- Only T speakers can be chosen (the sum of the decision variables has to match T).
- There has to be the same number of male and female speakers (the scalar product of Gen and Invited has to be 0)
- All invited speakers must come from different countries (If I[x] and I[y] are both 1, Countries[x] and Countries[y] must be different)
- The travel and accommodation expenses (the sum of the scalar product of Desloc and Invited + (the sum of the scalar product of Desloc and Invited *(The scalar product of Desloc and Invited gives you one value, so how can it be a sum?)*) * Night Price) must be covered by the budget (<= budget).

3.3 Evaluation Function

Since we aim to find the maximum distance amongst topics, we calculate the sum of distances between the topics of all invited speakers (Invited is boolean, Distance is >= 0).

$$\sum_{i=1}^{T} Invited_i \sum_{j=i}^{T} Invited_j * Distance_{ij}$$

3.4 Search Strategy

We used maximize(Total) in labeling, in order to find the set of invited speakers that maximizes the function described above (Total is the result of this function).

4 Solution Presentation

The predicate we use to display the result is shown below as well as the output.

```
printResult([], _, _, _, _, _, _, _).

printResult([0|Is], [_|Ss], [_|Ts], [_|Cs], [_|Gs], [_|Ds], [_|As], DPrice) :-
        printResult(Is, Ss, Ts, Cs, Gs, Ds, As, DPrice).

printResult([1|Is], [S1|Ss], [T1|Ts], [C1|Cs], [G1|Gs], [D1|Ds], [A1|As], DPrice) :-
        nome(S1, Nome),
        pais(C1, Pais),
        genero(G1, Genero),
        topic(T1, Topic),
        Aloj is A1 * DPrice,
        write(Nome), write(' - '), write(Genero), write(' - '), write(Pais),
        write(' - '), write(Topic), write(' - '), write(D1), write(' - '),
        write(Aloj), nl,
        printResult(Is, Ss, Ts, Cs, Gs, Ds, As, DPrice).
```

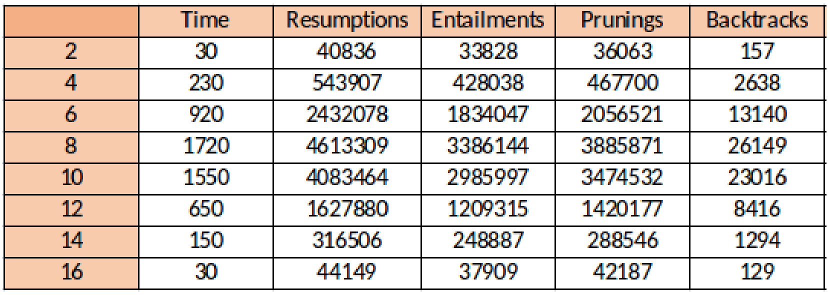
```
Result:
Time: 40
Resumptions: 40829
Entailments: 33828
Prunings: 36070
Backtracks: 157
Constraints created: 642
Topics distance: 10
Despesa alojamento: 3*100
Despesa deslocamento: 1700
Akemi - Male - Japao - Artificial Intelligence - 900 - 200
Seo-yun - Female - Coreio do Sul - Programming Sensors - 800 - 100
```

The **printResult** receives 8 arguments, Invited, Speakers, Topics, Countries, Gen, Desloc and Aloj arrays and the price per accommodation day. The predicate will then go through the Invited array and every time it's set to 1, it searches the database for the name, topic, country, gender, trip and accommodation values of that respective speaker according to the index received in the respective arrays.

We also display the statistics by using the **fd_statistics** predicate.

5 Results

We ran the application eight times to study the relation between the execution time, resumptions, entailments, prunings, backtracks and the number of lectures. Throughout the eight executions, the number of possible speakers was 20.



	Time	Resumptions	Entailments	Prunings	Backtracks
2	30	40836	33828	36063	157
4	230	543907	428038	467700	2638
6	920	2432078	1834047	2056521	13140
8	1720	4613309	3386144	3885871	26149
10	1550	4083464	2985997	3474532	23016
12	650	1627880	1209315	1420177	8416
14	150	316506	248887	288546	1294
16	30	44149	37909	42187	129

Fig. 1. Time, Resumptions, Entailments, Prunings and Backtracks depending on the number of lectures.

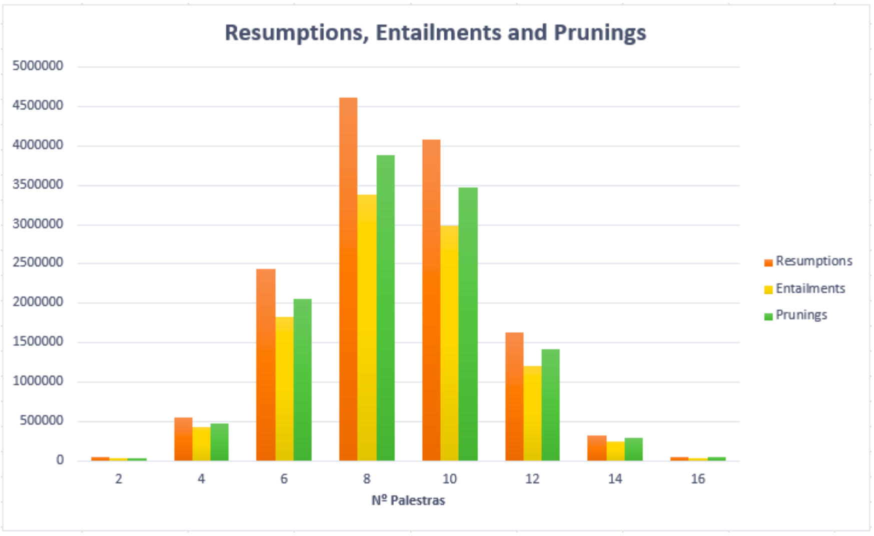


Fig. 2. Resumptions, Entailments and Pruning Graph.

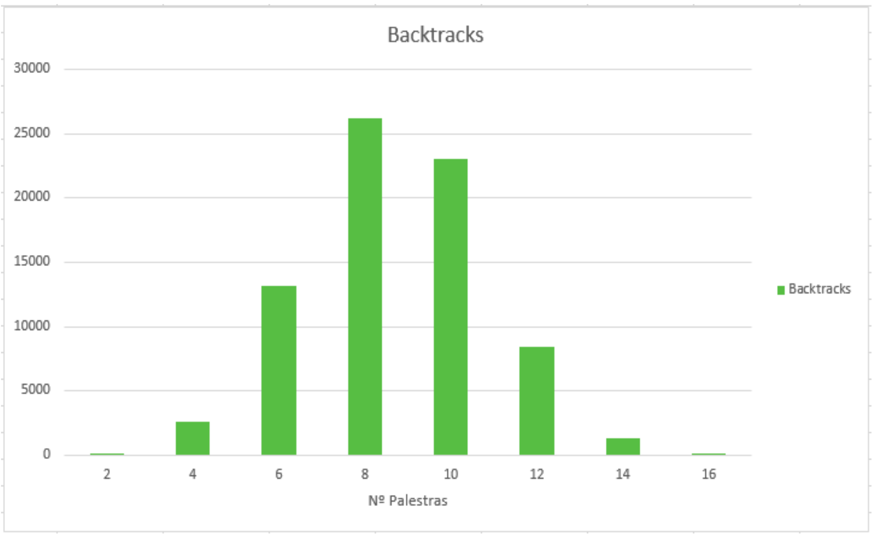


Fig. 3. Backtracks Graph.

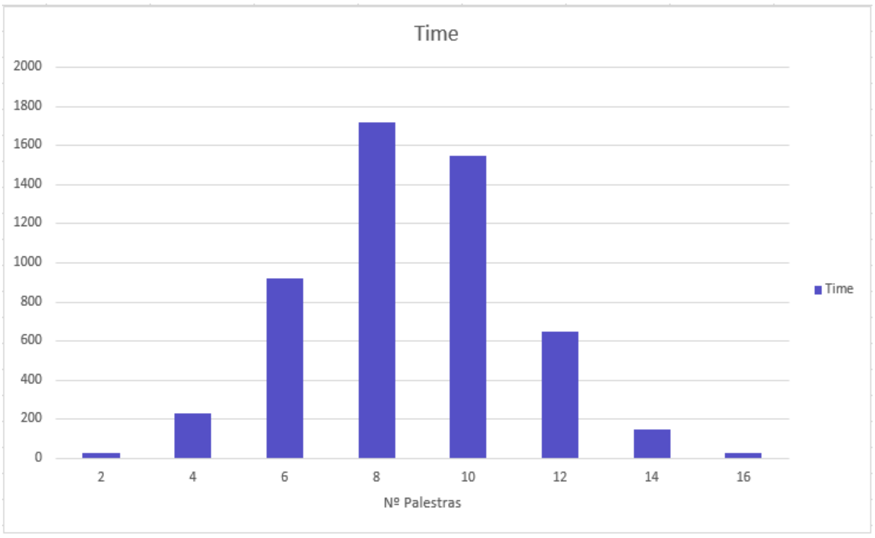


Fig. 4. Time Graph.

As we can see, all graphs have a similar distribution. The program finishes execution in less than a second when the number of lectures are 2, 4, 6, 12, 14 or 16. However, when the number of lectures are 8 or 10, the programs takes more than one second to execute. This behavior was already predicted and it can be easily explained with the Pascal's Triangle. The number of possible solutions increases the closest the number of lectures is to the average, therefore, to maximize the solution, it is necessary to process more possibilities, which takes more time.

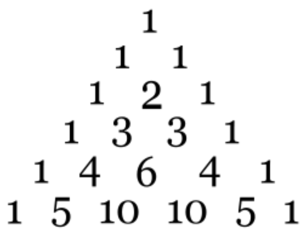


Fig. 5. Pascal Triangle example from 1 to 5 number of lectures.

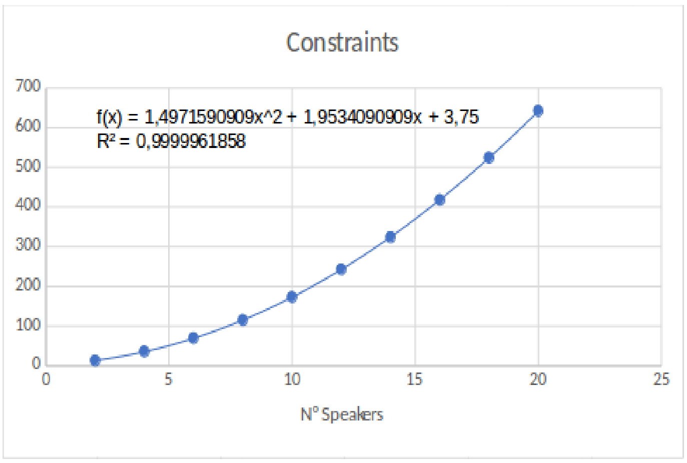


Fig. 6. Number of constraints depending on the number of available speakers.

As seen above, the number of constraints follows a quadratic distribution depending on the number of available speakers. Although most restrictions follow a linear evolution curve, the country restriction follows a quadratic one. For every possible pair of possible speakers, we need to force them to have different countries, therefore adding one speaker to the database implies adding N more country restrictions, N being the number of available speakers.

6 Conclusions and Future Work

This project was essential to better understand the mechanism behind logic programming with restrictions taught in the PLOG subject, since we were able to apply the concepts to the project. We learned that it is possible to solve complex problems with few lines of code using PLR and with great abstraction.

The results were very satisfactory since the program was able to solve a problem with some complexity in a short amount of time.

The project is well built and we don't think there's a lot of significant improvements. However, the country restrictions are significantly increasing the executing time since they follow a quadratic distribution. We weren't able to implement an algorithm to solve these restrictions with a linear complexity, but it's definitely one feature to improve, if possible.

References

1. SICStus Prolog, `https://sicstus.sics.se/`

7 Anexo

7.1 source.pl

```
:-include('Database.pl').
:-use_module(library(clpfd)).
:-use_module(library(lists)).
:-set_prolog_flag(toplevel_print_options,
                  [quoted(true), portrayed(true), max_depth(0)]).


invited(Invited,NPalestras, NDias):-
        createSpeakerArrays(Speakers, Topics, Countries, Gen, Desloc, Aloj),
        createProximityMatrix(Topics, Dists),
        length(Speakers, Len),
        length(Invited, Len),
        domain(Invited,0,1),
        night(DPrice),
        calcDiasAloj(Aloj, NDias, DiasAloj),
        scalar_product(Desloc,Invited,#=,AllDesloc),
        scalar_product(DiasAloj,Invited,#=,AllAloj),
        budget(Budget),
        (AllDesloc + (AllAloj * DPrice)) #=< Budget,
        SumLen is Len-1,
        length(Sums,SumLen),
        calcDist(Invited,Dists,Sums),
        sum(Sums,#=,Total),
        scalar_product(Gen,Invited,#=,0),
        sum(Invited,#=,NPalestras),
        alldif1(Invited,Countries),
        statistics(walltime,_),
        labeling([maximize(Total)], Invited),
        statistics(walltime,[_,T]),
        write('Time: '), write(T), nl,
        fd_statistics,
        write('Topics distance: '), write(Total),nl,
        write('Despesa alojamento: '), write(AllAloj * DPrice),nl,
        write('Despesa deslocamento: '), write(AllDesloc), nl,
        printResult(Invited, Speakers, Topics, Countries, Gen, Desloc, DiasAloj, DPrice)

createProximityMatrix([], []).
createProximityMatrix([T1|Ts], [D1|Ds]) :-
        getDistLine(T1, Ts, D1),
        createProximityMatrix(Ts,Ds).

getDistLine(_, [], []).
getDistLine(T2, [T1|Ts], [D1|Ds]) :-
       distance(T2, T1, D1),
       getDistLine(T2, Ts, Ds).

printResult([], _, _, _, _, _, _, _).

printResult([0|Is], [_|Ss], [_|Ts], [_|Cs], [_|Gs], [_|Ds], [_|As], DPrice) :-
        printResult(Is, Ss, Ts, Cs, Gs, Ds, As, DPrice).

printResult([1|Is], [S1|Ss], [T1|Ts], [C1|Cs], [G1|Gs], [D1|Ds], [A1|As], DPrice) :-
        nome(S1, Nome),
        pais(C1, Pais),
        genero(G1, Genero),
        topic(T1, Topic),
        Aloj is A1 * DPrice,
        write(Nome), write(' - '), write(Genero), write(' - '), write(Pais),
        write(' - '), write(Topic), write(' - '), write(D1), write(' - '),
        write(Aloj), nl,
        printResult(Is, Ss, Ts, Cs, Gs, Ds, As, DPrice).

createSpeakerArrays(Speakers, Topics, Countries, Gen, Desloc, Aloj) :-
        findall(Speaker, speaker(Speaker,_,_,_,_,_), Speakers),
        findall(Topic, speaker(_,_,Topic,_,_,_), Topics),
        findall(Country, speaker(_,_,_,Country,_,_), Countries),
        findall(Gender, speaker(_,Gender,_,_,_,_), Gen),
        findall(D, speaker(_,_,_,_,D,_), Desloc),
        findall(A, speaker(_,_,_,_,_,A), Aloj).

calcDiasAloj([],_,[]).
calcDiasAloj([A1|As], NDias, [D1|Ds]) :-
        (A1 #= 1) #=> D1 #= NDias,
        (A1 #= 2) #=> D1 #= 1,
        calcDiasAloj(As, NDias, Ds).

calcDist(_,[[]|_],_).

calcDist([I1|Is],[D1|Ds],[S1|Ss]):-
        scalar_product(D1,Is,#=,Sum),
        S1 #= Sum * I1,
        calcDist(Is,Ds,Ss).


alldif1([],[]).
alldif1([Inv|RInv],[Country|RCountries]):-
        alldif1_aux(Inv,RInv,Country,RCountries),
        alldif1(RInv,RCountries).

alldif1_aux(_,[],_,[]).
alldif1_aux(Inv,[Inv2|RInv],Country,[Country2|RCountries]):-
        (Inv#=1 #/\ Inv2#=1) #=> (Country #\= Country2),
         alldif1_aux(Inv,RInv,Country,RCountries).
```

7.2 Database.pl

```
distance(X, X, 0).

distance(1, 2, 1).
distance(1, 3, 10).
distance(1, 4, 10).
distance(1, 5, 10).
distance(1, 6, 10).
distance(1, 7, 10).
distance(1, 8, 10).
distance(1, 9, 10).
distance(1, 10, 10).
distance(1, 11, 1).
distance(1, 12, 10).
distance(1, 13, 10).
distance(1, 14, 10).
distance(1, 15, 10).
distance(1, 16, 10).

distance(2, 1, 1).
distance(2, 3, 10).
distance(2, 4, 1).
distance(2, 5, 10).
distance(2, 6, 10).
distance(2, 7, 10).
distance(2, 8, 10).
distance(2, 9, 10).
distance(2, 10, 10).
distance(2, 11, 1).
distance(2, 12, 10).
distance(2, 13, 10).
distance(2, 14, 10).
distance(2, 15, 10).
distance(2, 16, 10).

distance(3, 1, 10).
distance(3, 2, 10).
distance(3, 4, 1).
distance(3, 5, 10).
distance(3, 6, 10).
distance(3, 7, 10).
distance(3, 8, 10).
distance(3, 9, 10).
distance(3, 10, 10).
distance(3, 11, 10).
distance(3, 12, 1).
distance(3, 13, 10).
distance(3, 14, 10).
distance(3, 15, 10).
distance(3, 16, 10).

distance(4, 1, 10).
distance(4, 2, 1).
distance(4, 3, 1).
distance(4, 5, 10).
distance(4, 6, 1).
distance(4, 7, 10).
distance(4, 8, 1).
distance(4, 9, 10).
distance(4, 10, 10).
distance(4, 11, 10).
distance(4, 12, 10).
distance(4, 13, 10).
distance(4, 14, 10).
distance(4, 15, 10).
distance(4, 16, 10).

distance(5, 1, 10).
distance(5, 2, 10).
distance(5, 3, 10).
distance(5, 4, 10).
distance(5, 6, 10).
distance(5, 7, 10).
distance(5, 8, 10).
distance(5, 9, 10).
distance(5, 10, 10).
distance(5, 11, 10).
distance(5, 12, 1).
distance(5, 13, 10).
distance(5, 14, 10).
distance(5, 15, 10).
distance(5, 16, 10).
distance(6, 1, 10).
distance(6, 2, 10).
distance(6, 3, 10).
distance(6, 4, 1).
distance(6, 5, 10).
distance(6, 7, 10).
distance(6, 8, 10).
distance(6, 9, 10).
distance(6, 10, 1).
distance(6, 11, 10).
distance(6, 12, 10).
distance(6, 13, 10).
distance(6, 14, 10).
distance(6, 15, 10).
distance(6, 16, 10).

distance(7, 1, 10).
distance(7, 2, 10).
distance(7, 3, 10).
distance(7, 4, 10).
distance(7, 5, 10).
distance(7, 6, 10).
distance(7, 8, 10).
distance(7, 9, 10).
distance(7, 10, 10).
distance(7, 11, 10).
distance(7, 12, 10).
distance(7, 13, 10).
distance(7, 14, 1).
distance(7, 15, 10).
distance(7, 16, 10).

distance(8, 1, 10).
distance(8, 2, 10).
distance(8, 3, 10).
distance(8, 4, 1).
distance(8, 5, 10).
distance(8, 6, 10).
distance(8, 7, 10).
distance(8, 9, 10).
distance(8, 10, 10).
distance(8, 11, 10).
distance(8, 12, 10).
distance(8, 13, 10).
distance(8, 14, 10).
distance(8, 15, 10).
distance(8, 16, 10).

distance(9, 1, 10).
distance(9, 2, 10).
distance(9, 3, 10).
distance(9, 4, 10).
distance(9, 5, 10).
distance(9, 6, 10).
distance(9, 7, 10).
distance(9, 8, 10).
distance(9, 10, 10).
distance(9, 11, 10).
distance(9, 12, 10).
distance(9, 13, 10).
distance(9, 14, 10).
distance(9, 15, 1).
distance(9, 16, 10).

distance(10, 1, 10).
distance(10, 2, 10).
distance(10, 3, 10).
distance(10, 4, 10).
distance(10, 5, 10).
distance(10, 6, 1).
distance(10, 7, 10).
distance(10, 8, 10).
distance(10, 9, 10).
distance(10, 11, 10).
distance(10, 12, 10).
distance(10, 13, 10).
distance(10, 14, 10).
distance(10, 15, 10).
distance(10, 16, 10).

distance(11, 1, 1).
distance(11, 2, 1).
distance(11, 3, 10).
distance(11, 4, 10).
distance(11, 5, 10).
distance(11, 6, 10).
distance(11, 7, 10).
distance(11, 8, 10).
distance(11, 9, 10).
distance(11, 10, 10).
distance(11, 12, 10).
distance(11, 13, 10).
distance(11, 14, 10).
distance(11, 15, 10).
distance(11, 16, 10).

distance(12, 1, 10).
distance(12, 2, 10).
distance(12, 3, 1).
distance(12, 4, 10).
distance(12, 5, 1).
distance(12, 6, 10).
distance(12, 7, 10).
distance(12, 8, 10).
distance(12, 9, 10).
distance(12, 10, 10).
distance(12, 11, 10).
distance(12, 13, 10).
distance(12, 14, 10).
distance(12, 15, 10).
distance(12, 16, 10).

distance(13, 1, 10).
distance(13, 2, 10).
distance(13, 3, 10).
distance(13, 4, 10).
distance(13, 5, 10).
distance(13, 6, 10).
distance(13, 7, 10).
distance(13, 8, 10).
distance(13, 9, 10).
distance(13, 10, 10).
distance(13, 11, 10).
distance(13, 12, 10).
distance(13, 14, 10).
distance(13, 15, 10).
distance(13, 16, 1).

distance(14, 1, 10).
distance(14, 2, 10).
distance(14, 3, 10).
distance(14, 4, 10).
distance(14, 5, 10).
distance(14, 6, 10).
distance(14, 7, 1).
distance(14, 8, 10).
distance(14, 9, 10).
distance(14, 10, 10).
distance(14, 11, 10).
distance(14, 12, 10).
distance(14, 13, 10).
distance(14, 15, 10).
distance(14, 16, 10).

distance(15, 1, 10).
distance(15, 2, 10).
distance(15, 3, 10).
distance(15, 4, 10).
distance(15, 5, 10).
distance(15, 6, 10).
distance(15, 7, 10).
distance(15, 8, 10).
distance(15, 9, 1).
distance(15, 10, 10).
distance(15, 11, 10).
distance(15, 12, 10).
distance(15, 13, 10).
distance(15, 14, 10).
distance(15, 16, 10).

distance(16, 1, 10).
distance(16, 2, 10).
distance(16, 3, 10).
distance(16, 4, 10).
distance(16, 5, 10).
distance(16, 6, 10).
distance(16, 7, 1).
distance(16, 8, 10).
distance(16, 9, 10).
distance(16, 10, 10).
distance(16, 11, 10).
distance(16, 12, 10).
distance(16, 13, 1).
distance(16, 14, 10).
distance(16, 15, 10).


night(100).

budget(10000).
```

```
genero(-1, 'Female').
genero(1, 'Male').

%speaker(nome, genero, topico, pais, precoviagem, preco noite/pack)

speaker(1, 1, 1, 1, 200, 1).
speaker(2, -1, 2, 1, 200, 2).
speaker(3, 1, 3, 2, 700, 1).
speaker(4, -1, 4, 3, 500, 1).
speaker(5, 1, 5, 4, 100, 1).
speaker(6, -1, 6, 5, 800, 2).
speaker(7, 1, 7, 6, 800, 1).
speaker(8, -1, 2, 2, 400, 1).
speaker(9, 1, 8, 7, 400, 2).
speaker(10, -1, 9, 8, 500, 1).
speaker(11, 1, 10, 9, 400, 1).
speaker(12, -1, 8, 10, 900, 2).
speaker(13, 1, 11, 11, 600, 1).
speaker(14, -1, 12, 12, 400, 1).
speaker(15, 1, 13, 13, 200, 1).
speaker(16, -1, 14, 14, 700, 2).
speaker(17, 1, 15, 8, 1000, 2).
speaker(18, -1, 10, 15, 500, 1).
speaker(19, 1, 1, 10, 900, 1).
speaker(20, -1, 16, 16, 800, 2).

nome(1, 'Rui').
nome(2, 'Sofia').
nome(3, 'Will').
nome(4, 'Sarra').
nome(5, 'Pierre').
nome(6, 'Li').
nome(7, 'Elliot').
nome(8, 'Emily').
nome(9, 'Aleksander').
nome(10, 'Aada').
nome(11, 'Abraham').
nome(12, 'Aiko').
nome(13, 'Aarav').
nome(14, 'Emma').
nome(15, 'Omar').
nome(16, 'Mary').
nome(17, 'Aadolf').
nome(18, 'Marta').
nome(19, 'Akemi').
nome(20, 'Seo-yun').

pais(1, 'Portugal').
pais(2, 'USA').
pais(3, 'Tunisia').
pais(4, 'France').
pais(5, 'China').
pais(6, 'Australia').
pais(7, 'Polonia').
pais(8, 'Finland').
pais(9, 'Denmark').
pais(10, 'Japao').
pais(11, 'India').
pais(12, 'Holanda').
pais(13, 'Marrocos').
pais(14, 'Africa do Sul').
pais(15, 'Brasil').
pais(16, 'Coreio do Sul').

topic(1, 'Artificial Intelligence').
topic(2, 'Machine Learning').
topic(3, 'Web Security').
topic(4, 'Virtual Reality').
topic(5, 'Remote Access').
topic(6, 'Cloud Hosting').
topic(7, 'People VS Machine').
topic(8, 'Game Desing').
topic(9, 'Browser Compatibility').
topic(10, 'Big Data').
topic(11, 'Genetic Algorithm').
topic(12, 'Biometric Tecnology').
topic(13, 'Nanotechnology').
topic(14, 'Intelligent Assistance').
topic(15, 'Web: Responsive Design').
topic(16, 'Programming Sensors').
```