Organização de Jantar

Autores: Tiago José Grosso Pacheco (up201402722) e Vasco Ferreira Ribeiro (up201402723)

Afiliações: FEUP-PLOG, Turma 3MIEIC04, Grupo Organização de Jantares_5

Resumo

Neste trabalho teríamos, assim, de resolver o problema de Organizar um Jantar, utilizando a linguagem PROLOG e as diversas técnicas de restrição e otimização que nela existe. O problema de Organizar um Jantar consiste em conseguir dispor os vários convidados do jantar em mesas, de forma que, as pessoas se encontrem dispostas não aleatoriamente, mas sim pelo grau de afinidade entre as mesmas (quer grau de parentesco quer hobbies e interesses). Para resolver o problema, usamos a biblioteca clpfd do SICStus, pois contém várias funções uteis para a resolução.

1 Introdução:

O objetivo deste programa era implementar a solução de otimização em Prolog com restrições.

O problema Organização de Jantar tem como objetivo organizar convidados em mesas, de forma que haja compatibilidade entre os convidados.

Segue, no decorrer deste documento, a abordagem usada para a resolução do mesmo e a sua análise.

A solução implementada permite obter uma solução dando uma lista de listas já com as afinidades entre os convidados.

2 Descrição do Problema:

O problema principal deste trabalho é então Organizar um Jantar, dispondo os convidados em mesas. A organização tem como base um número máximo de mesas, um número máximo de convidados por mesa e um número de convidados. A principal restrição é as pessoas se sentir confortáveis, ou seja, terem afinidade com as pessoas que partilham a mesa. Pretende-se, assim, saber quantas mesas utilizar, e como ficarão distribuídas as pessoas.

3 Abordagem:

3.1 Variáveis de decisão:

A variável de decisão utilizada, foi a disposição dos convidados na mesa.

O domínio das mesas, Tables, é então um número de 1 a NumberofGuests (número de convidados), onde cada número representa um convidado.

```
domain(Tables,1,NumberofGuests)
```

3.2 Restrições:

As restrições, tal como explicado acima no ponto 2, servem para posicionar as pessoas num dos lugares da mesa.

Assim, através de restrições fizemos com que em primeiro lugar, as pessoas fossem diferentes das que se encontram ao seu lado.

Para além das restrições, relacionadas com a normal diferença de pessoa que está ao seu lado, tentamos com que primeiro verificar se a pessoa se encontra num grupo, caso acontecesse ficarem próximas. De seguida, verificamos o estado civil das pessoas e os seus interesses (livros ou filmes), de forma a que pessoas com gostos parecidos ficassem juntas.

```
restrictions(_,Index,NumberofTables,_,_):-
  Index >= NumberofTables.
restrictions(Tables,Index,NumberofTables,Numbero-
fGuests,Guests):-
  Index < NumberofTables,
  element(Index,Tables,P),
  NextInd is Index +1,
  element(NextInd,Tables,Pi),
  NextIndex is NextInd +1,
  P #\= Pi,
  nth1(P,Guests,[_,_,GId,Int,H]),
  nth1(Pi,Guests,[_,_,GIdi,Inti,Hi]),
  (GId \= 0 -> GId #= GIdi;Int #= Inti #\/ H #= Hi),
  restrictions(Tables,NextIndex,NumberofTables,Numbero-
fGuests,Guests).
```

3.3 Função de Avaliação:

Não achamos necessária a implementação de uma função de avaliação visto que as relações são dadas num ficheiro input, logo o erro é menor. Para além disso, seria relativo considerar se a pessoa A tem mais afinidades com a C ou a B em alguns aspetos, caso os gostos fossem todos idênticos.

3.4 Estratégia de Pesquisa:

A estratégia de pesquisa utilizada é percorrer a lista de convidados presentes, e ir verificando os seus gostos pondo restrições em cada convidado, de forma a que a compatibilidade seja maior.

4 Visualização da Solução:

Para visualização da solução, e das mesas preenchidas com os convidados, usamos um ciclo for que percorre o número de mesas e vai dando print dos convidados presentes em cada, que são guardados previamente numa lista de listas, onde o primeiro elemento é uma lista com os convidados da primeira mesa.

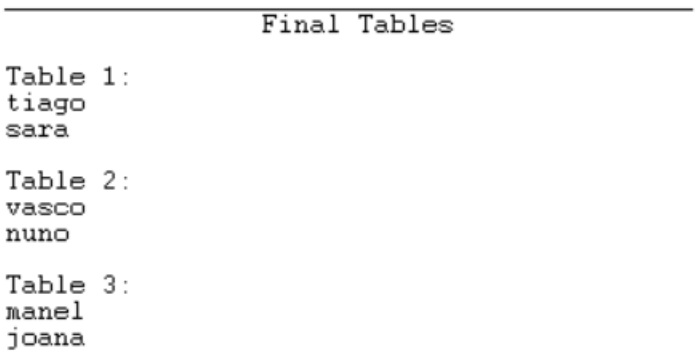

Fig. 1. Solução Apresentada.

5 Resultados:

No que toca a resultados, fica difícil observar a otimização neste problema, uma vez que, não conseguimos solucionar o problema para um diferente tamanho nas mesas. Assim, não conseguimos obter a diferença de tempos no caso de as mesas serem maiores ou menores.

Por outro lado, quando o número de convidados aumenta, o programa desenvolve-se de uma forma rápida tornando difícil cronometrar o efeito.

6 Conclusões:

Para concluir este projeto, podemos afirmar que a utilização de Prolog para certos problemas complexos, facilita o trabalho do programador, apesar de este não ser um problema de fácil resolução, o que levou a não o conseguir resolver na sua totalidade.. No entanto, no que toca a outras tarefas mais simples, como mostrar a representação das mesas de uma forma mais interativa, se tornar bastante mais complicado comparado com outras linguagens.