**3.ABORDAGEM**

Na resolução deste problema na linguagem Prolog foi utilizada uma lista para representar a distribuição de público inicial *(****InputGroups****),* pois para as pessoas do mesmo grupo ficarem em lugares contíguos não é necessário ter em consideração as restantes filas.

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |
| 10 | 11 | 12 |

Na posição correspondente ao número do lugar na lista *InputGroups*, está o número do grupo da pessoa inicialmente nesse lugar. Assim, se a tabela X representar o público, os números em cada célula o número de cada lugar e as cores os diferentes grupos, a lista tida em consideração para resolver o problema será: *InputGroups=[1,1,2,2,3,3,3,2,2,1,3,3].*

**3.1 VARIÁVEIS DE DECISÂO**

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 10 |
| 4 | 3 | 8 |
| 9 | 5 | 6 |
| 7 | 11 | 12 |

A solução do problema vem na forma de 2 listas que representam a *distribuição de público final*: uma para representar a distribuição dos grupos (***OutputGroups***) e outra que nos diz qual o lugar inicial da pessoa atualmente naquele lugar (***OutputIndexs***) Assim a solução deste problema tendo em conta a distribuição inicial a cima seria: *OutputGroups=[1,1,1,2,2,2,2,3,3,3,3,3]* e *OutputIndexs=[1,2,10,4,3,8,9,5,6,7,11,12].*

Tanto o tamanho da lista *OutputGroups*, como o da lista *OutputIndexs* é igual ao tamanho da lista *InputGroups*. No que toca ao domínio, o da lista *OutputGroups* é de 1 até ao número de grupos e da lista *OutputIndexs* é de 1 até ao número de pessoas.

**3.2 RESTRIÇÕES**

***1. Os elementos da lista OutputIndexs têm que ser todos distintos.***

Como cada posição da lista *OutputIndexs* é constituída pelo número do lugar inicial da pessoa atualmente nessa posição e os números de lugares são únicos foi chamado o seguinte predicado de restrição *all\_distinct(OutputIndexs).*

***2. Na lista OutputGroups o elemento na posição i tem que ser o elemento que está na posição j da InputGroups, sendo a posição j o elemento na posição i da lista OutputIndexs* (*OutputGroups[i] = InputGroups[j] AND j = OutputIndexs[i])***

É necessário garantir que os grupos estão corretamente atribuídos a cada lugar na solução do problema. Para isso foi utilizado o predicado *get\_groups(InputGroups, OutputIndexs, OutputGroups).*

get\_groups(\_,[],[]).

get\_groups(InputGroups, [OutputIndexsH|OutputIndexsT], [OutputGroupsH|OutputGroupsT]):-

element(OutputIndexsH, InputGroups, OutputGroupsH),

get\_groups(InputGroups, OutputIndexsT, OutputGroupsT).

***3. A distância de um elemento até ao próximo do mesmo grupo (caso exista) tem que ser 0***

Como o objetivo do problema é juntar as pessoas do mesmo grupo, impõe-se que a distância de um elemento de um grupo até ao próximo elemento do mesmo grupo caso este exista, para isso é usado o predicado *approximate(OutputGroups)*

approximate([]).

approximate([OutputGroupsH|OutputGroupsT]):-

get\_distance(Distance, NotUnique, OutputGroupsT, OutputGroupsH),

NotUnique #=> Distance #=0,

approximate(OutputGroupsT).

O predicado *get\_distance,* com a ajuda de um autómato, retorna em *Distance* a distância até ao primeiro elemento com o valor *OutputGroupsH* e caso este exista o predicado retorna em *NotUnique* o valor 1, caso contrário o valor 0.

**3.3 FUNÇÃO DE AVALIAÇÃO**

Este problema é um problema de otimização, ou seja, o objetivo não é somente encontrar uma solução, mas sim encontrar a melhor solução.

Neste caso, a melhor solução é a que implica:

1) menor distância percorrida por cada pessoa na mudança de lugar

2) menor número de trocas

Assim, para cada lugar na distribuição final é calculada a diferença desse mesmo lugar e do lugar inicial da pessoa, isto é possível pois a lista *OutputGroups* guarda o lugar inicial na posição final da pessoa. Para isso é usado o seguinte predicado *fill\_differences(OutputIndexs,OutputIndexs, Differences).*

fill\_differences(\_,[],[]).

fill\_differences(OutputIndexs, [OutputIndexsH|OutputIndexsT], [DifferencesH|DifferencesT]):-

element(OutputPos, OutputIndexs, OutputIndexsH),

DifferencesH #= abs(OutputPos-OutputIndexsH),

fill\_differences(OutputIndexs, OutputIndexsT, DifferencesT).

Depois da lista *Differences* ter sido obtida, somam-se todos os elementos desta lista obtendo-se a variável *TotalDifference*

Seguidamente conta-se o número de elementos não zero da lista *Differences*, ou seja, contam-se os elementos que se moveram com a ajuda do predicado *get\_changes(NumOfChanges, Differences),* que usa um autómato para retornar o contador desejado em *NumOfChanges.*

Assim, para minimizar com igual peso o deslocamento das pessoas e o número de deslocamentos significa otimizar uma função (linear) objetivo:

***Minimizar F = 1 \* TotalDifference + 1 \* NumOfChanges***

**3.4 ESTRATÉGIA DE PESQUISA**

Foram testadas várias opções de pesquisa para a resolução deste problema. Para se poder chegar a alguma conclusão com os testes teve que ser usada a mesma distribuição inicial. Esta distribuição é uma plateia de 10 pessoas de 5 grupos diferentes, tendo cada grupo 2 pessoas: *InputGroups = [1,2,3,4,5,1,2,3,4,5].* A tabela X apresenta os dados desses mesmos testes. Podemos então concluir que a melhor estratégia de pesquisa é a utilização das opções *step min* e a pior estratégia é o uso das opções *bisect max*.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **leftmost** | **min** | **max** | **first\_fail** | **anti\_first\_fail** | **occurrence** | **most\_constrained** | **max\_regret** |
| **step** | 3,347 | **1,718** | 18,072 | 18,708 | 49,706 | 2,877 | 16,060 | 21,570 |
| **enum** | 3,533 | 5,125 | 18,717 | 18,383 | 21,699 | 3,121 | 19,539 | 20,850 |
| **bisect** | 3,223 | 2,997 | **73,378** | 17,607 | 56,772 | 2,932 | 16,652 | 27,014 |
| **middle** | 3,409 | 4,655 | 28,249 | 30,042 | 71,761 | 2,957 | 28,405 | 25,004 |
| **median** | 3,534 | 5,056 | 28,257 | 23,818 | 69,325 | 3,250 | 25,992 | 25,295 |

A tabela X apresenta as vários durações (em segundos) da resolução do problema registadas para várias combinações de opções de labelling. As opções em cada linha são as que controlam em que modo é que as escolhas são feitas para cada variável selecionada, enquanto que as opções em cada coluna são as opções que controlam a ordem em que a próxima variável é escolhida para atribuição.

**5 RESULTADOS**

Para se poderem tirar conclusões dos resultados obtidos foram medidos o tempo de resolução, o número de retrocessos e o número de restrições criadas. Seguem-se as condições de teste e as respetivas conclusões:

1. Fez-se variar o número de pessoas na audiência, mantendo-se o número de grupos (tabela X, gráfico X, gráfico X e gráfico X).

O tempo de resolução do problema e o número de retrocessos variam exponencialmente com o aumento do número de pessoas da audiência, enquanto que o número de restrições criadas varia linearmente com o aumento de pessoas da audiência. Pode-se então concluir que o tempo de resolução depende do número de retrocessos e não do número de restrições criadas.

1. Fez-se variar o número de grupos, mantendo-se o número de pessoas na audiência (tabela X, gráfico X, gráfico X e gráfico X).

Tal como nas condições anteriores, o tempo e o número de retrocessos variam da mesma forma, confirmando a conclusão que o tempo depende do número de retrocessos e não do número de restrições criadas.

O número de restrições criadas mantém-se para o mesmo número de pessoas da audiência, com a exceção de quando os elementos são todos do mesmo grupo. Provavelmente isto deve-se ao facto de que quando os elementos são do mesmo grupo o domínio da variável *OutputGroups* será de 1 a 1, sendo logo atribuídos os valores à lista não precisando das restantes restrições.

O tempo varia exponencialmente quando o nº de grupos varia de 1 a 5. Quando varia entre 6 e 9, o tempo varia de forma não conclusiva. Provavelmente será pelo facto de que, quando o nº de grupos é maior que 5 e porque o nº de elementos é 10, haverá grupos só de 1 elemento enquanto outros grupos têm 2 elementos, isto vai tornar as condições inconstantes. Quando o número de grupos é 10, o tempo é muito pequeno pois só há um elemento de cada grupo, não havendo necessidade para trocas