Sum Star and Logic Programming *

Duarte Carvalho[1 [201503661]]

Faculdade de Engenharia da Universiade do Porto, Porto, Portugal feup@fe.up.pt
https://sigarra.up.pt/feup/

Abstract. Neste artigo, está descrito o problema de decisão "Sum Star", assim como vários aspetos da aplicação desenvolvida em PROLOG com restrições, para resolver este tipo de puzzles, como, por exemplo, a abordagem ao problema, a visualização da solução, e, também, resultados e conclusões deste projeto.

Keywords: Logic Programming · Restrictions · NP-Problems · PROLOG · Puzzles

1 Introdução

O objetivo deste trabalho é a construção de um programa em Programação em Lógica com Restrições, desenvolvido em PROLOG, capaz de resolver puzzles do jogo "Sum Star", que é um problema de decisão combinatória, estabelecendo variáveis de decisão, domínios, restrições e estratégias de pesquisa, para que se possa encontrar uma solução correta no menor tempo possível.

Este artigo começa por descrever o problema escolhido, depois é explicada a abordagem que foi escolhida para resolver o problema, de seguida abordamos a visualização da solução que o programa apresenta no fim da sua execução e finalmente apresentamos resultados e refletimos sobre estes na conclusão.

* Supported by organization FEUP.

2 Descrição do problema

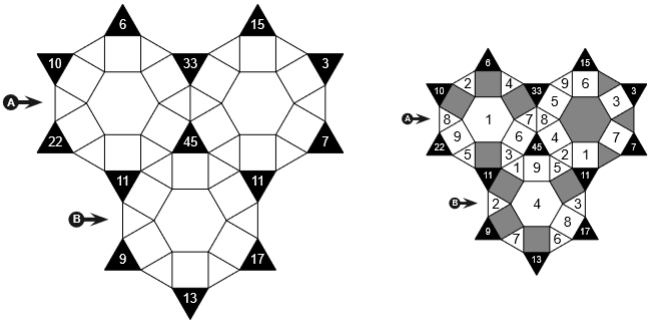

Fig. 1. Estado inicial do puzzle (lado esquerdo) e possível solução(lado direito).

Na figura 1, do lado esquerdo, temos o estado inicial do puzzle. O puzzle é constituído por três dodecágono, cada um destes rodeado por 6 triângulos pretos, podendo alguns dos triângulos ser partilhados entre vários dodecágonos. Cada dodecágono tem 6 quadrados, 6 triângulos e um hexágono.

O objetivo deste puzzle é preencher com números de 1 a 9 ou um X (equivale a um 0) os polígonos que estão dentro de cada dodecágono, podendo apenas usar uma vez cada número dentro de cada dodecágono, no entanto, o X tem de ser usado 4 vezes, de forma a que a soma dos números que estão nos nos polígonos que partilham um vértice com um triângulo pretos seja igual ao número que está nesse triângulo preto, sem que nenhum desses elementos da soma se repita, a não ser que seja um 'X'. Outra condição é que, polígonos com um 'X' não podem partilhar arestas com outros elementos que tenham um 'X'.

Olhando para a solução do puzzle, percebemos facilmente estas regras.

2.1 Resumo

- Cada dodecágono contém números de 1-9 e quatro 'X'.
- A soma dos números em polígonos que partilham uma aresta com um triângulo preto tem de ser igual ao número neste, sem repetições.
- Polígonos marcados com um 'X' não podem partilhar arestas com outro que esteja marcado com 'X'

3 Abordagem

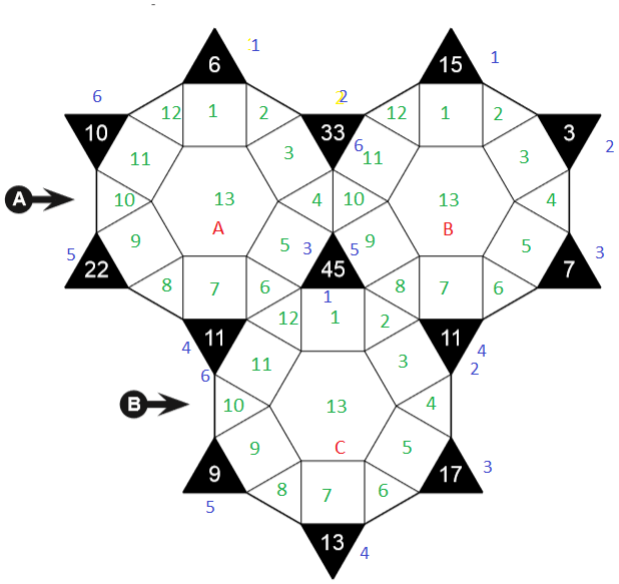

Fig. 2. Índices das listas: vermelho - índice correspondente a um dodecágono. verde - índice dum polígono dum dodecágono. azul - índice dos triângulos para restringir soluções

Para resolver este problema decidi criar o seguinte predicado:

```
solveSimpleCase(+ListOfLists, -R).
```

Recebe em 'ListOfLists' uma lista de listas com os valores dos triângulos à volta de cada dodecágono, nos índices marcados a azul, podendo elementos serem repetidos, visto que há triângulos partilhados entre dodecágonos. Neste caso, 'ListOfLists' seria:

```
firstProblem(ListOfLists) :-
  ListOfLists = [
              [6, 33, 45, 11, 22, 10],
              [15, 3, 7, 11, 45, 33],
              [45, 11, 17, 13, 9, 11]
              ].
```

Este predicado retorna outra lista de listas, correspondendo cada lista a um dodecágono (A, B ou C), em que cada elemento duma lista é o valor achado pelo programa para cada índice, ou seja, retorna uma lista do género:

```
R = [
        [A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11, A12, A13],
        [B1, B2, B3, B4, B5, B6, B7, B8, B9, B10, B11, B12, B13],
        [C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C11, C12, C13]
    ]
```

As variáveis nestas listas, são as variáveis de decisão.

3.1 Variáveis de decisão

Tal como dito na secção anterior, as variáveis de decisão são todas as que pertencem à lista retornada por 'solveSimpleCase'.Todas essas variáveis pertencem ao domínio finito de números inteiros de 0 a 9.

3.2 Restrições

De forma a obter resultados corretos foram aplicadas várias restrições:

- Cada lista em R, só pode conter uma vez cada número de 1 a 9 e pode ter quatro 0's.
- Os valores em índices que partilhem arestas com triângulos pretos, quando somados, têm de no total ser iguais ao valor no triângulo e não podem haver elementos repetidos.
- Dois índices seguidos, excluindo o índice 13, não podem ter o valor 0 ('X'), isto aplica-se para triângulos partilhados, sendo necessário comparar elementos das várias listas.

global$_{c}$*ardinality* Não necessariamente uma restrição, mas talvez um estabelecimento de domínio, implicando que os valores sejam todos diferentes, a não ser que sejam 0's.

simpleRestrictions Esta restrição recebe a lista de triângulos que rodeiam um dodecágono e restringe os valores da lista que partilham uma aresta com um triângulo preto não partilhado, obrigando que a soma desses valores seja o valor nesse triângulo. Por exemplo: no caso do dodecágono A na figura 2, esta restrição obrigaria a que a soma dos índices 1, 2 e 12 fosse igual a 6. O mesmo é feito para os índices que partilham aresta com os triângulos pretos de índice 6 e 5. Neste caso, não nos precisamos de preocupar com elementos repetidos na soma, porque como apenas elementos de um dodecágono são utilizados, estes já estão restringidos pela restrição descrita anteriormente.

```
A1 + A2 + A12 #= Restriction1
```

doubleRestrictions Esta restrição, tal como a anterior, recebe uma lista que contém o valor do triângulo preto que é partilhado pelas duas listas sobre as quais esta restrição vai atuar, garantindo que a soma dos valores que partilham uma aresta com este triângulo é igual ao valor do triângulo e que são todos diferentes, podendo haver 0's repetidos. Por exemplo: na figura2, para os dodecágonos A e B, esta restrição obrigaria que os índices 2, 3, 4 do dodecágono A e os índices 10, 11, 12 do dodecágono B sejam todos diferentes e que a soma destes valores seja igual a 33.

```
notRepeatedInSum2([A2, A3, B11, B12]),
A2 + A3 + A4 + B10 + B11 + B12 #= Restriction2.
```

O predicado 'notRepeatedInSum2', que será abordado mais à frente, garante que estes elementos são todos diferentes.

tripleRestriction Esta restrição, recebe uma lista com o valor do triângulo central, partilhado pelos 3 dodecágonos, e recebe as 3 listas com os elementos dos dodecágonos e restringe os valores que partilham uma aresta com o triângulo preto central, garatindo que a soma destes elementos é igual ao valor do triângulo central e que os valores são todos diferentes, podendo haver 0's repetidos. Na figura 2, por exemplo, a restrição obrigaria a que a soma dos valores dos índices 4, 5 e 6 do dodecágono A, 10, 9 e 8 do dodecágono B e os valores dos índices 12, 1, e 2 do dodecágono C, fosse igual a 45 e que todos esses valores fossem diferentes, podendo haver 0's repetidos.

```
notRepeatedInSum([A4, A5, A6, B10, B9, B8, C12, C1, C2]),
A4+A5+A6+B10+B9+B8+C12+C1+C2 #= Restriction.
```

O predicado 'notRepeatedInSum', que será abordado mais à frente, garante que estes elementos são todos diferentes.

notRepeatedInSum e notRepeatedInSum2 Estas restrições são aplicadas quando se restringe os valores dos índices que partilham arestas com o triângulo central e com os trîângulos partilhados entre dois dodecágonos, garantindo que esses valores são todos diferentes, podendo haver 0's repetidos.

crossListShadedEdges Esta restrição garante que os valores dos polígonos onde dois dodecágonos se tocam não são 0's, ou seja, não são 'X'. Por exemplo, obriga que a soma do valor do índice 4 do dodecágono A e do valor do índice 10 do dódecagono B, seja diferente de 0.

```
A4 + B10 #\= 0.
```

shadedDontShareEdge Esta restrição garante que se o valor do hexágono central é 0, o valor dos quadrados não pode ser 0 e que num par constituído por um triângulo e um quadrado, só um deles é que tem o valor 0.

3.3 Função de avaliação

Como a solução deste problema é verificável através do output final das listas, não é necessário uma função de avaliação para verificar se a solução está correta.

3.4 Estratégia de pesquisa

A estratégia de pesquisa escolhida foi o 'min' que atribui primeiro valor à variável mais à esquerda que pode ter o valor mais baixo do domínio atribuído a essa variável. Esta estratégia não foi escolhida a priori, mas sim a posteriori, porque não consegui arranjar nenhuma razão teórica que me levasse a escolher outra opção de pesquisa a não ser a default. No entanto, testei várias vezes o programa com as várias estratégias de pesquisa possíveis e concluí que a opção 'min' era sempre a mais rápida, como se pode ver no próximo gráfico onde estão representados por barras tempos de execução, em milisegundos, das várias opções de labeling (quanto mais curta a barra, menor o tempo de execução).

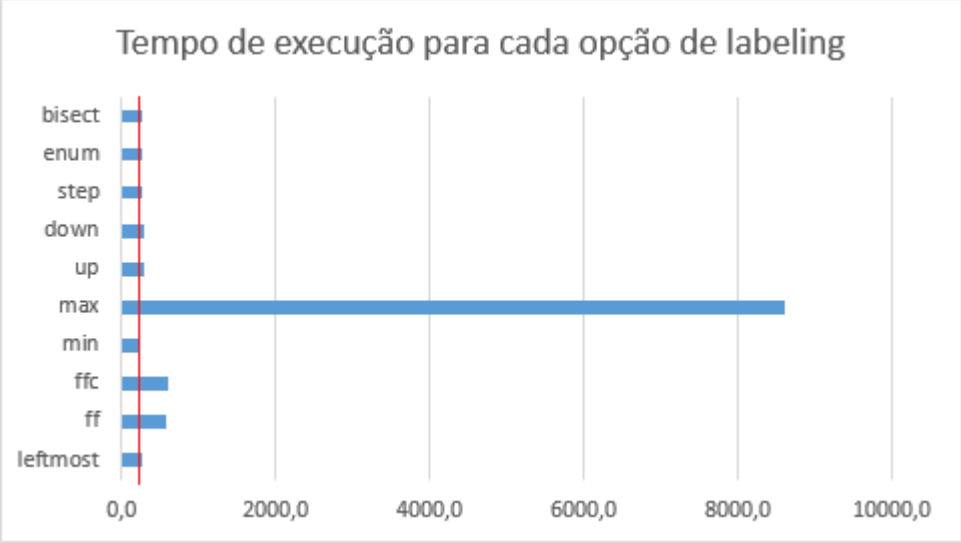

Fig. 3. Gráfico de tempos de execução das várias opções de labeling.

4 Visualização da solução

O predicado de visualização de solução, 'printSolutions', está no ficheiro 'facts.pl'. Com este predicado o utilizador consegue ver em forma de lista os valores achados para cada índice do dodecágono. É apresentado a letra do dodecágono (A, B ou C) e os valores à frente. Os 0's são representados com X, tal como é dito no enunciado do problema. Para além disso são apresentadas estatísticas da resolução do problema.

```
Time: 0.53s

Resumptions: 827492
Entailments: 144830
Prunings: 485639
Backtracks: 10256
Constraints created: 8950

A:  X 4 X 7 6 3 X 5 9 8 X 2 1

B:  6 X 3 X 7 X 1 2 4 8 5 9 X

C:  9 5 X 3 8 6 X 7 X 2 X 1 4
```

Fig. 4. Exemplo de visualização de um problema.

5 Resultados

Como não consegui fazer o meu programa de forma a que aceitasse diferentes números de dodecágonos, sendo impossível fazer comparações e apresentar resultados.

6 Conclusões e trabalho futuro

Este projecto denotou as diversas vantagens da linguagem para resolver problemas de decisão combinatória, e penso que é impressionante como é possível resolver um problema, aparentemente complexo, apenas estabelecendo variáveis de decisão, os seus domínios e aplicar restrições aos seus domínios.

Infelizmente, não consegui fazer o meu trabalho de forma a que pudesse resolver problemas com mais de três dodecágonos, de forma a que fosse possível obter e analisar resultados. No entanto, penso que não podia ter feito melhor, visto que tínhamos de entregar 4 projectos na mesma semana e eu não tinha colega de grupo, penso que se pudesse dividir tarefas com um colega podia ter satisfeito todos os requisitos deste projecto.

References

1. SICStus Prolog, https://sicstus.sics.se/. Last accessed 21 Dec 2018

7 Anexos

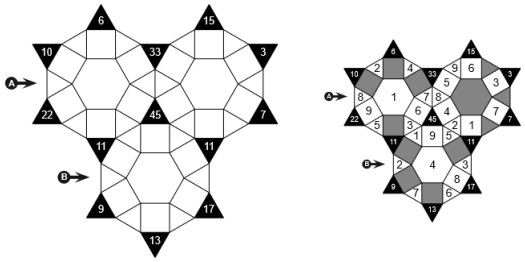

Fig. 5. Estado inicial do puzzle (lado esquerdo) e possível solução(lado direito).

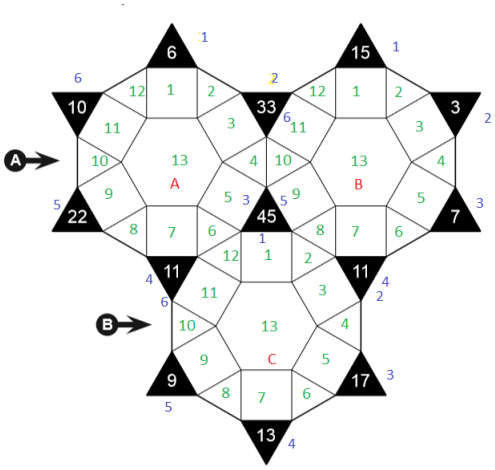

Fig. 6. Índices das listas: vermelho - índice correspondente a um dodecágono. verde - índice dum polígono dum dodecágono. azul - índice dos triângulos para restringir soluções

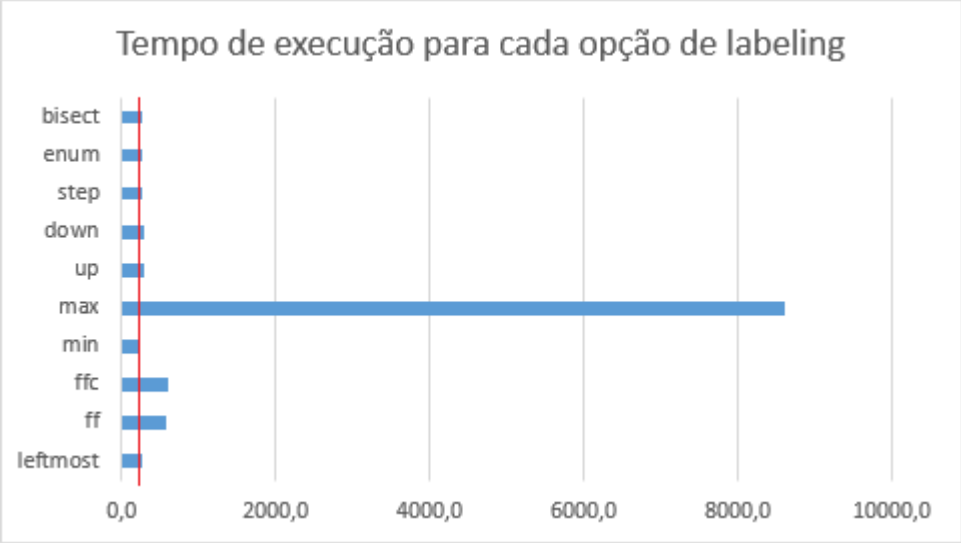

Fig. 7. Gráfico de tempos de execução das várias opções de labeling.

	Opção de Labeling									
Tempo (ms)	leftmost	ff	ffc	min	max	up	down	step	enum	bisect
Tentativa 1	260	570	600	210	8570	280	280	280	280	280
Tentativa 2	280	570	600	200	8620	280	290	260	260	260
Tentativa 3	280	570	610	200	8600	280	290	280	260	260
Média	273,3	570,0	603,3	203,3	8596,7	280,0	286,7	273,3	266,7	266,7

Fig. 8. Tabela de tempos de execução das várias opções de labeling.

```
Time: 0.53s

Resumptions: 827492
Entailments: 144830
Prunings: 485639
Backtracks: 10256
Constraints created: 8950

A:  X 4 X 7 6 3 X 5 9 8 X 2 1


B:  6 X 3 X 7 X 1 2 4 8 5 9 X


C:  9 5 X 3 8 6 X 7 X 2 X 1 4
```

Fig. 9. Exemplo de visualização de um problema.