Bosnian Snake

Luís Correia[1] and Vicente Espinha[2]

[1] Turma 04, Grupo BosnianSnake_5
up201503342@fe.up.pt
[2] Turma 04, Grupo BosnianSnake_5
up201503764@fe.up.pt

Resumo Este artigo completa a realização do segundo projeto da Unidade Curricular de Programação em Lógica do Mestrado Integrado em Engenharia Informática e Computação da Faculdade de Engenharia da Universidade do Porto. O projeto tem como principal ojectivo a realização da abordagem necessária para a resolução de problemas de satisfação de restrições, por forma a aplicá-la em casos concretos, como os problemas de otimização ou decisão combinatória. A abordagem foi aplicada ao puzzle 2D Bosnian Snake, construindo um programa que implementa um solver para a sua resolução. Através da manipulação de predicados disponibilizados pelo SICTUS Prolog, mostramos neste artigo a resolução deste problema e respectiva analise.

Keywords: feup,plog,prolog,bosnian snake,sisctus

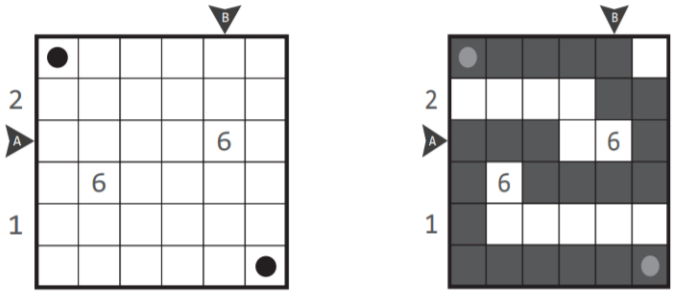
1 Introdução

Este projeto, desenvolvido no âmbito da Unidade Curricular de Programação em Lógica do Mestrado Integrado em Engenharia Informática e Computação da Faculdade de Engenharia da Universidade do Porto, tem como principal objetivo avaliar a capacidade para resolver problemas de otimização ou decisã, utilizando conceitos de programação em lógica com restrições.

De entre as várias opções disponibilizada, o grupo escolheu um problema de decisão: o puzzle Bosnian Snake.O puzzle consiste em criar uma cobra(linha continua), em que a posição final e inicial são fornecidas. A cobra não se toca, nem na diagonal. Números fora da grelha indicam o número de células da cobra na respetiva coluna ou linha.Números dentro da grelha indicam quantas, das 8 células à volta deste, são células da cobra.

2 Descrição do Problema

O Bosnian Snake é um puzzle de tamanho variável. O tabuleiro é inicializado com a posição inicial e final da cobra, e, talvez, alguns números dentro do tabuleiro e outros fora, tal como podemos observar na imagem da esquerda. O principal objetivo consiste em preencher o tabuleiro com uma linha contínua(cobra),desde a posição inicial até à final, tendo exatamente o número de células pintadas quanto os números indicarem. Os números exteriores ao tabuleiro indicam o número exato de células pintadas na respetiva coluna ou linha. Por outro lado, os números no interior referem-se ao número de células pintadas que aquela célula tem de ter nas 8 células adjacentes. A figura do lado direito ilustra o exemplo anterior resolvido.



3 Abordagem

Primeiramente,para a conceção da abordagem,tentou-se perceber como modelar o puzzle como um problema de restrições. De seguida, escolher as variavéis de decisão a usar no predicado labeling, entender as restrições necessárias para a resolução do problema e aplicá-las, restrigindo as variáveis.

Foi tido em consideração a melhor forma de o puzzle ser vizualizado. Sendo a consola do SICSTUS Prolog muito básica, a representação da cobra é feita com o valor 1, e com valor 0 as casas vazias.

3.1 Variáveis de Decisão

Neste puzzle, as variáveis de decisão correspondem às células do tabuleiro , que inicialmente se encontram vazias. O domínio delas vaia entre 0 e 1, correspondendo 1 a célula da cobra e 0 casa vazia.

3.2 Restrições

As restrições utilizadas na resolução deste puzzle são as seguintes:

- Verificar que a cobra não se toca, nem na diagonal, ou seja, verificar que à volta de uma célula cobra("1") exitem ,exatamente, duas céluas cobra(continues/3).a.
- Números fora da grelha indicam o número de células da cobra na respetiva coluna ou linha(Vertical_restrictions/4 e horizontal_restrictions/4).
- Números dentro da grelha indicam quantas, das 8 células à volta deste, são células da cobra(around/3).

3.3 Estratégia de Pesquisa

De modo a tornar a pesquisa mais eficiente, foi utilizada a opção ffc(first fail cosntraint), no predicado laveling para ordenação das variáveis. Esta estratégia consiste em ordenar as variáveis, das que têm mais restrições para as que têm menos.

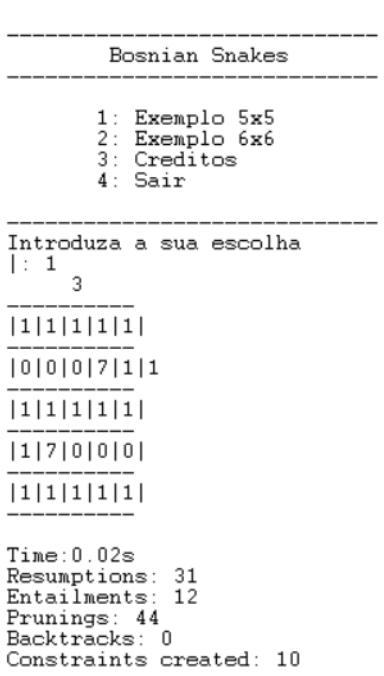
Para a ordenação de valores foi utilizada a estratégia por omissão do labeling(ordem ascendente).

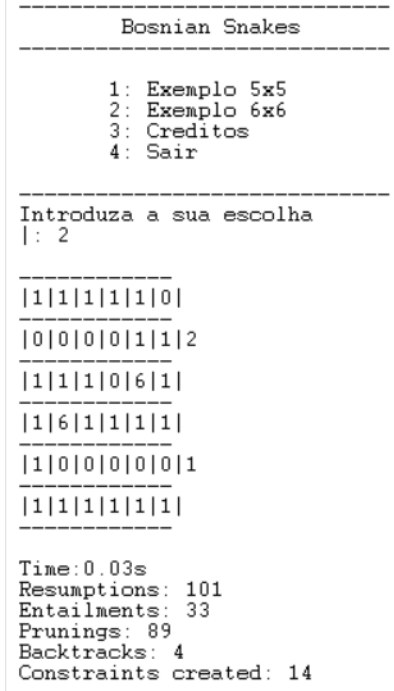
4 Visualização da Solução

Em relação à visualização da solução, é utilizado o predicado , que percorre todas as linhas do tabuleiro, desenhando-as de forma mais agradável para o utilizador.Este predicado, também, utiliza o predicado , que ajuda a delimitar o tabuleiro.

Para além do tabuleiro, são ainda mostradas algumas estatísticas como o tempo que demora a resolver o puzzle, se foi feio algum backtracking, número de restrições feitas, entre outras.

As imagens abaixo ilustram dois exemplos(tabuleiro 5x5 e 6x6,respetivamente).





5 Resultados

Após os testes com os exemplos apresentados acima, verificou-se que para um tamanho menor de taboleiro, o programa resolve-o mais rapidamente, em comparação com outros superiores, como podemos verificar na tabela abaixo.

	ffc		
	Tempo(s)	Backtracks	Constraints
5x5	0.02	0	10
6x6	0.03	4	14

6 Conclusões e Trabalho Futuro

Este projeto foi essencial para uma melhor compreensão do mecanismo subjacente à programação em lógica com restrições.

Conclui-se,também, que a linguagem Prolog é uma linguagem muito poderosa e eficiente para a resolução de problemas de satisfação de restrições, como os problemas de otimização ou decisão combinatória.

Por fim,considerou-se que os resultados não são totalmente satisfatórios, pois não foi feita a geração aleatória de um novo tabuleiro,visto que demorou-se muito tempo a resolver alguns bugs, e, no final,não houve tempo para o fazer.

Referências

1. Bosnian Snake,
   `http://logicmastersindia.com/lmitests/dl.asp?attachmentid=645&view=1`
2. SICStus Prolog,
   `https://sicstus.sics.se/`