Four Winds - Programação em Lógica
Resolução de Problema de decisão/optimização usando Restrições

Duarte Brandão e Pedro Tavares

Faculdade de Engenharia da Universidade do Porto,
Rua Roberto Frias, sn, 4200-465 Porto, Portugal
FEUP-PLOG, Turma 3MIEIC06, Grupo FourWinds_4
ei10060@fe.up.pt,up201406991@fe.up.pt
http://www.fe.up.pt

Resumo Four Winds é um puzzle baseado numa grelha com algumas células preenchidas com números, desenhando-se linhas a partir dessas células de forma a que o tamanho das linhas provenientes de uma determinada célula sejam iguais ao número representado. Estas linhas horizontais ou verticais não se podem cruzar ou sobrepor umas as outras ou com os números. O objectivo deste trabalho é criar um algoritmo que resolva grelhas Four Winds recorrendo à linguagem CLP utilizando o ambiente SICSTUS.

Keywords: Four Winds, Puzzle, Restrições, Domínios, Prolog, CLP

1 Introdução

Com este trabalho pretende-se obter conhecimentos de boas práticas de programação em lógica recorrendo a restrições e domínios para resolver problemas de decisão. Foi também uma oportunidade para experimentarmos funções nativas do SICSTUS das quais não tínhamos conhecimento até agora. O objetivo deste trabalho é resolver um puzzle Four Winds. O programa por nós desenvolvido analisa e resolve problemas Four Winds de NxN células.

2 Descrição do Problema

Um tabuleiro de Four Winds consiste numa lista de listas com posições vazias(0) e posições preenchidas que indicam o tamanho das linhas que partem desse ponto. O objetivo do puzzle é completar o tabuleiro com linhas que partem destes pontos sem que elas de intersetem entre si.

3 Abordagem

O tabuleiro do puzzle é analisado e a localização e valor de cada célula inicialmente preenchida no tabuleiro guardada em NUMLIST. Os números presentes na solução final correspondem ao index nesta lista.

3.1 Variáveis de Decisão

Cada célula vazia do Tabuleiro é uma variável de decisão e, visto que usamos uma lista de listas para representarmos o Tabuleiro, para fazermos o *labeling* usamos o predicado *makeLabeling* que aplica o *labeling* a cada uma das listas do Tabuleiro. O domínio de cada uma destas variáveis vai desde 1 até ao número de células inicialmente preenchidas com números.

3.2 Restrições

Para a resolução do nosso puzzle usamos as seguintes restrições:

Com esta restrição conseguimos controlar que o número de ocorrências de cada Index no VARLIST é igual ao valor em NUMLIST+1.

```
1 % Control the size of Index spread
2 lineControl(VARLIST,NUMLIST,1),
3 colControl(VARLIST,NUMLIST,1),
```

Com esta restrição garantimos que o spread do Index em VARLIST não se sobrepõem sobre outros Index.

```
1 % Control the Index not crossing over
2 overPass(VARLIST,1,NUMLIST),
```

4 Visualização da Solução

De forma a representar a solução do puzzle, apresentamos o tabuleiro preenchido com números indicando as várias linhas.

```
1 Sol:
2 | 2| 1| 1| 1|
3 | 2| 3| 3| 3|
4 | 2| 4| 3| 5|
5 | 2| 4| 3| 5|
```

5 Visualização das Estatísticas

Para que pudéssemos comparar a execução com as diferentes otimizações recorremos à biblioteca *system* para determinar o tempo de execução e a *fd_ statistics* para obter informações sobre o processamento das restrições necessárias para a resolução do problema.

```
An answer has been found!
Elapsed time: 0.001 seconds
Resumptions: 3203768216
Entailments: 1500249915
Prunings: 1468800870
Backtracks: 7627304
Constraints created: 3299
```

6 Resultados

O tempo de execução varia dependendo da complexidade do problema e do tamanho do mesmo, como também poderá variar de computador para computador. Mesmo assim deixamos aqui alguns dos resultados obtidos como referência.

6.1 Problema Tab 4x4

```
tab1([
  [0,0,2,0],
  [3,0,4,0],
  [0,0,0,0],
  [0,1,0,1]
  ]).

Sol:
| 2| 1| 1| 1|
| 2| 3| 3| 3|
| 2| 4| 3| 5|
| 2| 4| 3| 5|

An answer has been found!
Elapsed time: 0.000 seconds
Resumptions: 156
Entailments: 98
Prunings: 244
Backtracks: 0
Constraints created: 98
```

6.2 Problema Tab 5x5

```
tab2([
  [0,0,3,0,0],
  [3,0,4,0,0],
  [0,0,0,0,1],
  [1,1,0,1,0],
  [0,3,0,0,0]
  ]).

Sol:
| 2| 1| 1| 1| 1|
| 2| 2| 3| 3| 3|
| 2| 6| 3| 4| 4|
| 5| 6| 3| 7| 7|
| 5| 8| 8| 8| 8|

An answer has been found!
Elapsed time: 0.000 seconds
Resumptions: 1229
Entailments: 470
Prunings: 887
Backtracks: 3
Constraints created: 371
```

6.3 Problema Tab 7x7

```
tab4([
  [0,0,0,0,10,0,0],
  [3,0,0,0,0,0,2],
  [5,0,0,0,0,2,0],
  [0,4,0,0,0,0,0],
  [0,0,3,0,0,0,0],
  [0,0,0,0,0,0,5],
  [2,0,0,3,0,0,0]
  ]).

Sol:
| 1| 1| 1| 1| 1| 1| 1|
| 2| 2| 2| 2| 1| 3| 3|
| 4| 4| 4| 4| 1| 5| 3|
| 4| 6| 6| 6| 1| 5| 8|
| 4| 6| 7| 7| 1| 5| 8|
| 9| 6| 7| 8| 8| 8| 8|
| 9| 9| 7|10|10|10|10|

An answer has been found!
Elapsed time: 0.095 seconds
```

```
Resumptions: 250100
Entailments: 106969
Prunings: 111176
Backtracks: 2903
Constraints created: 853
```

7 Conclusões

Embora o nosso programa resolva os puzzles, conforme o tamanho dos tabuleiros aumenta, também o tempo até encontrar uma solução aumenta pelo que no exemplo 12x12 não conseguimos obter uma solução em menos de 3 horas. Uma forma de reduzir este tempo seria melhorar a atribuição do domínio a cada uma das variáveis de decisão, por exemplo, o domínio só poderia ser o conjunto de números que se encontram na mesma linha e coluna que a variável.

Com este trabalho fomos capazes de perceber o quão úteis poderão ser as restrições em Prolog e como estas poderão simplificar a resolução de alguns problemas, que noutra linguagem seria muito mais complexa.

Referências

1. Four Winds - World of Puzzles, http://www.worldpuzzle.org/championships/types-of-puzzles/wpc/