STAR BATTLE

Relatório Final

Eduardo Fernandes ei12130@fe.up.pt[1] and
José Ricardo Coutinho ei12161@fe.up.pt[1]

Faculdade de Engenharia da Universidade do Porto

Resumo *Star Battle*. Resolução de um puzzle recorrendo a utilização de restrições na programação em lógica.Recorrendo à biblioteca *clpfd* do *SICStus Prolog* foi possível implementar um *solver* capaz de resolver puzzles que estejam de acordo com as restrições do mesmo. O programa resultante permite obter todas as soluções válidas de qualquer puzzle válido fornecido. Também é possível de gerar puzzles.

1 Introdução

O objectivo principal deste trabalho foi a implementação de um programa capaz de resolver um puzzle na linguagem *Prolog* recorrendo à programação em lógica com restrições sobre domínios finitos.

2 Descrição do Problema

O problema consiste num puzzle que é composto por um tabuleiro quadrangular de dimensão N por N.

A dimensão N define o número de áreas no tabuleiro, assim sendo, se tivermos um puzzle de 5 por 5 temos necessariamente de ter 5 áreas no tabuleiro, como o representado na figura 1.

Com o tabuleiro também é fornecido um número de estrelas, o mesmo representa o número de estrelas a serem colocadas em cada uma das áreas referidas anteriormente, assim como o número de estrelas a ser colocadas em cada linha e coluna. Assim sendo o número total de estrelas irá ser o produto do número de estrelas fornecido com o número de áreas do tabuleiro (equação 1).

$$N_{totalestrelas} = N_{estrelas} * N_{areas} \tag{1}$$

No caso da figura 1 temos no total 5 estrelas (equação 2).

$$N_{totalestrelas} = 1 * 5 = 5 \tag{2}$$

Não é possível colocar estrelas em posições adjacentes onde já exista outra peça, em qualquer das direções, seja vertical, horizontal ou diagonal.

Um exemplo de um tabuleiro e sua solução encontra se na figura 1.

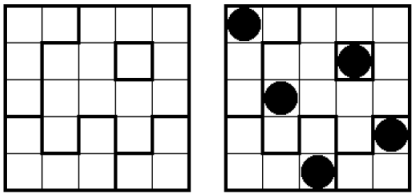

Figura 1: Tabuleiro e sua solução

Em suma o objetivo do jogo é colocar todas as estrelas de maneira a satisfazer todas as regras.

3 Abordagem

3.1 Variáveis de Decisão

As variáveis de decisão do solver são as coordenadas das estrelas. Estas estão diretamente relacionadas com as posições fornecidas pelas áreas e com o tamanho tabuleiro.

3.2 Restrições

As restrições para a resolução do tabuleiro com sucesso encontram-se abaixo. As mesmas encontram-se separadas da mesma maneira no programa.

Peças adjacentes - Segundo as regras do jogo não é possível ter estrelas adjacentes, seja na horizontal, na vertical ou mesmo na diagonal. Assim sendo foi necessário criar uma restrição que verifique se existe alguma peça ou adjacente.

A restrição consiste em garantir que nenhum destes 3 sistemas de equações sejam verdadeiros dando qualquer conjunto de duas estrelas, sendo x e y as coordenadas de uma determinada estrela.

Caso diagonal:

$$\begin{cases} |\mathtt{x}_1 - \mathtt{x}_2| & = 1 \\ |\mathtt{y}_1 - \mathtt{y}_2| & = 1 \end{cases} \tag{3}$$

Caso horizontal:

$$\begin{cases} |\mathtt{x}_1 - \mathtt{x}_2| & = 1 \\ |\mathtt{y}_1 - \mathtt{y}_2| & = 0 \end{cases} \tag{4}$$

Caso vertical:

$$\begin{cases} |\mathtt{x}_1 - \mathtt{x}_2| & = 0 \\ |\mathtt{y}_1 - \mathtt{y}_2| & = 1 \end{cases} \tag{5}$$

Horizontal - O número de estrelas em cada linha tem de ser exactamente igual ao número de estrelas fornecido com o tabuleiro, independentemente da area em que se encontrarem.

Vertical - O número de estrelas em cada coluna tem de ser exactamente igual ao número de estrelas fornecido com o tabuleiro, independentemente da area em que se encontrarem.

Área - Dentro de cada área o número de estrelas fornecido tem de ser exatamente igual ao fornecido com o tabuleiro.

3.3 Estratégia de Pesquisa

Aplica primeiro a restrição de domínio entre 1..N sobre as coordenadas de cada estrela. Depois restringe as coordenadas de X e Y por um número de ocorrências, utilizando o predicado global_cardinality, onde cada X=1,2,..,N vai ocorrer N vezes. Segue-se a restrição de proximidade que impede que qualquer estrela não possa estar adjacente a outra, isto é que não possa estar a 1 unidade de distância em qualquer direção. Por fim define-se as estrelas como elementos de uma área.

4 Visualização da Solução

O resultado é visualizado através da consola do *SICStus*. O programa retorna uma solução na forma de lista com as coordenadas:
[X1,Y1,X2,Y2,X3,Y3,..]
e
mostra uma matrix preenchida com áreas numeradas e estrelas.

5 Resultados

Foi possivel cumprir todas as restrições definidas na seccção 3.2.

```
| ?- run1(L).
[2950,0]
[3440,490]

Resumptions: 475287
Entailments: 139595
Prunings: 242717
Backtracks: 1120
Constraints created: 78382

Map of Matrix:
[1,1,2,2,2]
[1,2,2,3,2]
[1,2,2,2,2]
[4,2,4,2,5]
```

```
[4,4,4,5,5]

Map of Solutions:
[*,1,2,2,2]
[1,2,2,*,2]
[1,*,2,2,2]
[4,2,4,2,*]
[4,4,*,5,5]

L = [1,1,2,3,4,2,3,5,5,4] ?



| ?- run2(L).
[2940,10]
[2950,10]

Resumptions: 16372
Entailments: 4155
Prunings: 7701
Backtracks: 73
Constraints created: 1612

Map of Matrix:
[4,1,4,4]
[4,4,4,2]
[3,4,4,4]
[4,4,4,4]

Map of Solutions:
[4,*,4,4]
[4,4,4,*]
[*,4,4,4]
[4,4,*,4]

L = [2,1,4,2,1,3,3,4] ?
```

6 Conclusão

Foi possivel implementar com sucesso o *solver* para o puzzle.