**Four Winds:**

**Resolução de Problemas de Decisão com recurso a Programação Lógica com Restrições**

Ana Rita Torres e Rui Pedro Soares

**Turma:**3MIEIC06**, Grupo:** Four\_Winds\_2

Faculdade de Engenharia da Universidade do Porto,

Rua Dr. Roberto Frias, s/n 4200-465 Porto, Portugal

{up201406093, up201404965}@fe.up.pt

http://www.fe.up.pt

**Resumo** Este artigo completa a realização do segundo projeto da Unidade Curricular de Programação em Lógica do Mestrado Integrado de Engenharia Informática e Computação. O projeto tem como objetivo principal o desenvolvimento da solução para o jogo *Four Winds* escrita em Prolog, apresentado este um problema de decisão. A abordagem usada permite lidar com tabuleiros de diversas dimensões, assim como com um número de peças variável. É também implementada uma solução do problema sob a forma de texto de modo a permitir a visualização desta.

**Palavras-Chave:** prolog, four winds, solução, decisão

# Introdução

O objetivo deste projeto era implementar a resolução de um problema de decisão ou otimização em Prolog com restrições.

O grupo optou por um problema de decisão, em específico, o puzzle Four Winds. O puzzle apresenta uma estrutura quadrangular e o seu preenchimento é realizado através de traços e setas.

Este artigo visa descrever de forma detalhada o problema; a abordagem utilizada pelo grupo para a resolução deste; a explicação da representação utilizada para as soluções; as estatísticas de resolução de puzzles de diferentes dimensões; conclusões relativas ao projeto, no geral.

# Descrição do Problema

O puzzle Four Winds apresenta uma estrutura quadrangular, isto é, tem o mesmo número de linhas e colunas. Um tabuleiro inicial tem o seguinte aspeto.

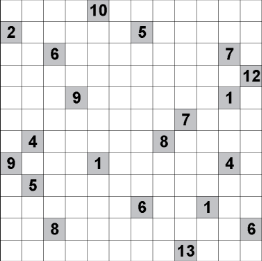


Figura - Puzzle Inicial(12x12)

Desenha-se uma ou mais linhas a partir de cada célula numerada de forma a que cada número indique o comprimento total de linhas que são desenhadas a partir dessa célula, excluindo a própria célula. As linhas podem ser, apenas, verticais ou horizontais. Estas podem ligar centros de células adjacentes, mas sempre sem se sobrepor ou cruzar entre elas e, relativamente às células também. Não há qualquer tipo de limitação relativo à repetição dos valores das células numeradas.

Para facilitar a familiarização com as regras do puzzle, segue-se o tabuleiro anterior resolvido.

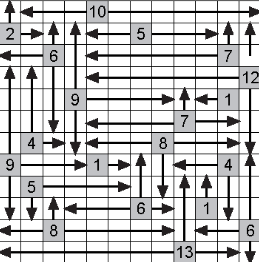


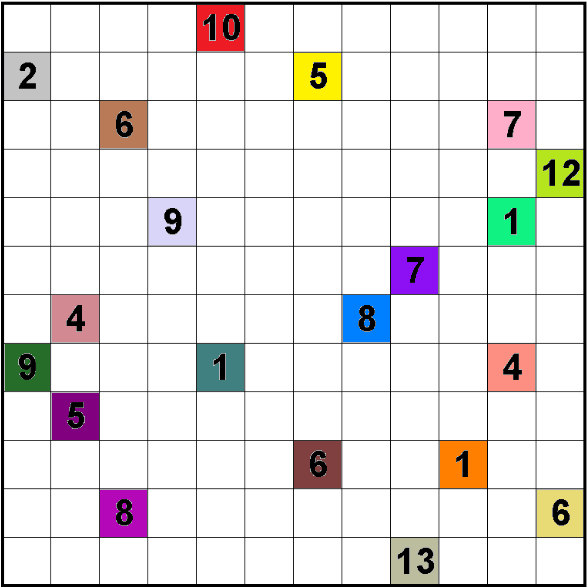
Figura - Puzzle Resolvido(12x12)

# Abordagem

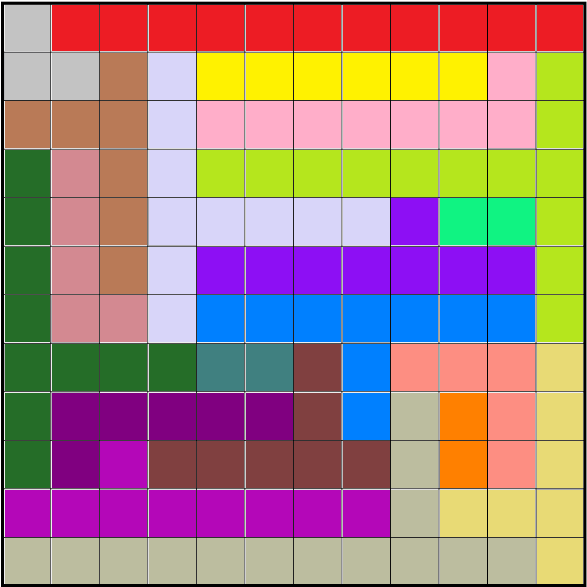
Tal como muitos outros dos outros projetos à escolha, este trata-se de um problema de obter áreas, no tabuleiro, que se adequem à solução. Como tal, abordámos o problema com a seguinte resolução em mente:

1. Cada número no tabuleiro inicial corresponde à área de uma região, excluindo a célula onde se encontra. O primeiro passo consiste em identificar essa região com um índice.
2. Obter todos os tuplos (Índice da Região, Área da Região, Coordenada X da capital, Coordenada Y da capital).
3. Inicializar um tabuleiro com variáveis, e colocar os índices das capitais no sítio correto.
4. Aplicar restrições às restantes células que não sejam capitais.
5. Obter a solução possível, através de *labeling*.

Como tal, um tabuleiro como o seguinte



, deverá resultar na solução possível.



A solução obtida para cada célula consiste no índice da região que a ocupa. No exemplo, a célula (1,0) teria índice 0, correspondente à região da capital em (4,0), com valor de área igual a 10.

O predicado que trata de resolver o puzzle é ***solver(Puzzle, Solution)***, sendo que Puzzle é o tabuleiro (em formato Prolog) inicial, e Solution a matriz-solução. Solution pode dar mais que um resultado, consoante o puzzle escolhido.

É de notar que a interface criada (predicado fourWinds) mostra-nos apenas uma das potenciais múltiplas soluções.

Para além disto, achamos relevante referir que enquanto a nossa solução é uma matriz, o predicado *labeling* necessita de uma lista simples.

Para resolver isto, recorremos ao "flattening" da matriz-solução, ou seja, convertê-la numa lista simples (preservando todas as restrições, domínios, etc. aplicados) e só depois aplicando *labeling*.

Depois disto, voltamos a convertê-la numa matriz, para ser mais intuitiva de imprimir no ecrã.

## Variáveis de Decisão

A solução tem NxN variáveis de decisão totais (uma para cada célula).

Cada variável deve ter como seu domínio a região possível que pode ocupar a respetiva casa.

Inicialmente, é-lhes dado o domínio de 0 (índice mínimo) a Z, sendo Z o número de regiões a serem criadas (obtido facilmente, porque cada capital representa sempre uma região, logo Z é o número de capitais).

Depois, é-lhes restrito o domínio de acordo com as regiões que podem ocupar a célula. Cada casa pode no máximo ter 4 regiões (cima, baixo, esquerda, direita) no seu domínio.

Na geração da solução (em labeling), testa-se todos os valores possíveis de variáveis de decisão.

## Restrições

Organizámos as principais restrições do puzzle em três predicados:

* ***constrainCells***: cada célula pode no máximo ter no domínio da sua variável de decisão correspondente 4 regiões possíveis.

Esta função vai, de célula a célula, restringir o domínio de cada variável para as possíveis casas.

Isto é feito verificando no tabuleiro inicial, para as coordenadas da célula atual, as regiões imediatamente (excluindo espaços vazios) acima, abaixo, à esquerda e à direita da célula. Estas serão as únicas regiões passíveis de pertencer ao domínio da variável de decisão da célula, e como tal, restringe-se para tal.

* ***constrainAreas***: este é essencialmente o grande problema do puzzle: fazer com que as regiões tenham áreas correspondentes às indicadas pelas capitais.

Isto é feito obtendo a área da região, e restringindo-a para o indicado na capital. Isto exclui todas os resultados em que uma área tivesse mais células ocupadas do que o indicado.

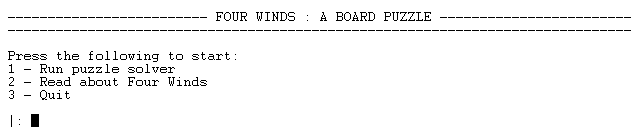
* ***constrainCohesions***: embora as outras restrições sejam as mais fundamentais, não chegam, pois podiam levar a casos onde a área de cada região é efetivamente respeitada, mas as regiões ficariam divididas pelas outras regiões. Este predicado obriga a que o total de células a partir de cada capital seja o da área indicada, e portanto, que a região esteja coesa.

## Estratégia de Pesquisa

Usámos a estratégia de etiquetagem *bisect*. No entanto, as outras opções de etiquetagem não mudam significativamente a performance do programa, devido à natureza da solução pela qual optámos, e pela quantidade de restrições aplicadas.

# Visualização da Solução

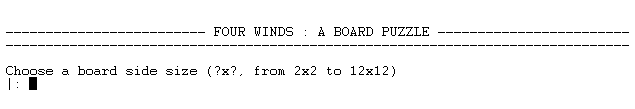
Para iniciar o programa é necessário inserir: “fourWinds. “na consola e o utilizador irá deparar-se com o seguinte menu.



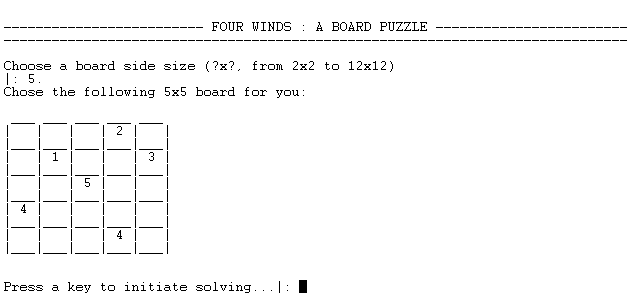
Como se pode verificar este dá-nos três opções:

1. Ir para o solucionador de puzzles;
2. Ler sobre o puzzle;
3. Sair do jogo.

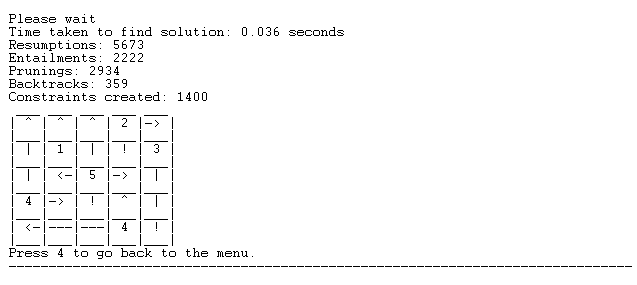
Ao selecionar o “1.” o utilizador é redirecionado para o ecrã abaixo apresentado.



Aqui o utilizador pode escolher o tamanho do tabuleiro, entre os valores apresentados, que quer ver solucionado. Para efeitos de exemplificação, suponha-se que o utilizador optou por um tabuleiro de tamanho 5. A impressão deste é realizada pelo predicado ***printUnsolvedPuzzle*** este recebe o puzzle sob a forma de matriz, uma matriz que é composta por números, no caso da célula se encontrar vazia -1, se se encontrar numerada estará lá representado esse valor.



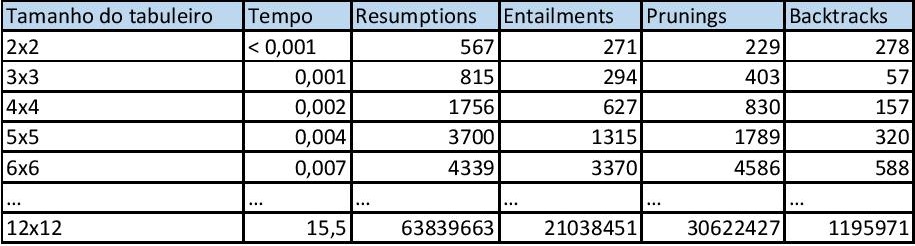
Esta imagem representa um tabuleiro do tamanho inserido pelo utilizador no seu estado inicial e clicando em qualquer tecla é-lhe apresentada a solução, assim como as estatísticas relativas ao puzzle.



Como se pode ver no ponto 2, o jogo usa setas para indicar a direção em que as linhas são desenhadas as representações horizontais pareceram lógicas ao grupo por outro lado, as representações verticais implicaram alguma criatividade e como tal o acento circunflexo (^) é usado para representar uma seta para cima e o ponto de exclamação (!) a seta para baixo.

Os predicados desenvolvidos para a visualização do tabuleiro encontram-se no ficheiro printer.pl.

# Resultados



A performance do código desenvolvido revelou-se pouco eficiente em puzzles que apresentam dimensões maiores. Os resultados apresentados na tabela não são 100% fiáveis, uma vez que usamos uma amostra pequena (cerca de 4 a 5 puzzles por dimensão), consequência de não ter sido concebido um gerador de puzzles.

# Conclusões e Trabalho Futuro

# Bibliografia

1. Regras do Four Winds, <http://www.worldpuzzle.org/championships/types-of-puzzles/wpc/>
2. SWI-Prolog, <http://www.swi-prolog.org>
3. SICStus Prolog, <http://sicstus.sics.se/>

# Anexo