

# **Puzzle 2D - Wrong Products**

Segundo Projeto

**Hugo Miguel Monteiro Guimarães**  
**Beatriz Costa Silva Mendes**

Trabalho realizado no âmbito da  
Unidade Curricular de Programação Lógica



Mestrado Integrado em Engenharia Informática e Computação  
Faculdade de Engenharia da Universidade do Porto

Porto

28 de dezembro de 2020

## Contents

<b>1</b>	<b>Resumo</b>	<b>2</b>
<b>2</b>	<b>Introdução</b>	<b>2</b>
<b>3</b>	<b>Descrição do Problema</b>	<b>3</b>
<b>4</b>	<b>Abordagem</b>	<b>3</b>
4.1	Variáveis de Decisão . . . . .	3
4.2	Restrições . . . . .	4
<b>5</b>	<b>Visualização da Solução</b>	<b>5</b>
<b>6</b>	<b>Experiências e Resultados</b>	<b>5</b>
6.1	Análise Dimensional . . . . .	5
6.2	Estratégias de Pesquisa . . . . .	5
<b>7</b>	<b>Conclusões e Trabalho Futuro</b>	<b>6</b>
<b>8</b>	<b>Referências</b>	<b>6</b>
<b>9</b>	<b>Anexo</b>	<b>6</b>

## 1 Resumo

Este trabalho foi desenvolvido no âmbito da Unidade Curricular de Programação em Lógica, através do sistema SICStus Prolog, e o seu objetivo foi resolver um problema de decisão utilizando Programação de lógica com restrição sobre domínios finitos. O problema escolhido foi ***Wrong Products*** e ao longo do relatório iremos explicar no que este consiste.

## 2 Introdução

O Problema de Otimização escolhido, *Wrong Products*, tem em vista colocar números numa grelha de modo a que cada linha e coluna contenha apenas 2 dígitos e que o produto entre os mesmos corresponda a um determinado valor no exterior da grelha, com variação de uma unidade.

Este problema tem como objetivo a implementação de uma grelha matricial com restrições sobre linhas, colunas e toda a malha utilizando programação de lógica com restrições sobre domínios finitos.

Este artigo possui a seguinte estrutura:

**Descrição do Problema:** Descrição com detalhe do problema de otimização ou decisão em análise, incluindo todas as restrições envolvidas.

**Abordagem:** Descrição da modelação do problema como um Problema de Satisfação de Restrições(PSR)

**Variáveis de Decisão** Descrição das variáveis de decisão e respetivos domínios, assim como o seu significado no contexto do problema em análise.

**Restrições** Descrição das restrições rígidas e flexíveis do problema e a sua implementação utilizando o SICStus Prolog.

**Visualização da Solução:** Explicação dos predicados que permitem visualizar a solução em modo de texto.

**Experiências e Resultados** Análise de problema e resultados obtidos

**Análise Dimensional** Exemplificação da execução de instâncias do problema com diferentes dimensões e análise dos resultados obtidos.

**Estratégias de Pesquisa** Descrição de diferentes estratégias de pesquisa, comparando os resultados.

**Conclusões e Trabalho Futuro:** Conclusões obtidas pela realização do trabalho

**Referências** Referências bibliográficas utilizadas

**Anexo** Anexos de resultados úteis para a resolução do problema.

### 3 Descrição do Problema

Wrong Products é um problema de decisão. Este problema pretende descobrir se é possível colocar números numa grelha de modo a que cada que cada número apareça uma única vez, e que cada linha e coluna contenha unicamente dois números, e que o seu produto corresponda a uma unidade acima ou abaixo de um determinado valor no exterior da grelha associado à respetiva linha ou coluna.

### 4 Abordagem

Wrong Products é um Problema de Satisfação de Restrições(PSR) e é modelado por:

#### 4.1 Variáveis de Decisão

Na resolução deste problema, é necessário criar as seguintes variáveis:

**Length** - Tamanho da grelha, indicando o número de linhas e colunas. Estes valores são iguais dado que a grelha corresponde a uma matriz quadrada. O valor de *length* é passado como argumento do predicado, pelo que o seu domínio varia conforme o pretendido pelo utilizador.

**RowValues** - Lista com os valores iniciais cuja diferença entre o produto dos 2 valores da respetiva linha seja 1.

**ColValues** - Lista com os valores iniciais cuja diferença entre o produto dos 2 valores da respetiva coluna seja 1.

RowValues e ColValues possuem o mesmo domínio, todos os valores inteiros positivos que não sejam superiores ao produto entre os 2 maiores valores permitidos, que corresponde a:

$$(Length * 2) * (Length * 2 - 1)$$

**ListOfLists** - Lista de Listas contendo a grelha do problema. Pode ser interpretada como uma representação matricial do problema. Contém valores inteiros a serem multiplicados em cada linha e coluna, e zeros que correspondem a casas vazias.

**Transpose** - Lista de Listas contendo a matriz transposta da grelha do problema. Tem como propósito facilitar a aplicação de restrições às colunas da grelha do problema.

**Table** - Forma achatada de ListOfLists. tem como objetivo permitir a aplicação de restrições sobre toda a grelha.

Dado que apenas podem existir 2 números por linha e coluna, o domínio de ListOfLists, Transpose e Table é o mesmo e corresponde a todos os valores inteiros entre 0 e  $2 * Length$ .

## 4.2 Restrições

**all\_distinct\_except\_0(List)** Permite aplicar uma restrição a todos os valores da grelha, de modo a que, tal como o nome indica, todos os valores sejam distintos exceto os zeros. Desta forma é possível garantir que não há valores semelhantes e, simultaneamente, utilizar o valor zero como representação de uma casa não ocupada.

**line\_restriction(List,Amount)** Recebe como argumentos uma lista de listas e um valor inteiro *amount*, e restringe a quantidade de zeros em cada linha de uma Lista de Listas, consequentemente garantindo que existem apenas 2 valores em cada linha. Este predicado é evocado duas vezes, a primeira para *ListOfList*, e a segunda para *Transpose*, de modo a que a restrição seja aplicada quer a linhas quer a colunas.

**multiplication\_restriction(ListOfLists,List)** Recebe uma Lista de Listas e verifica se o produto dos valores diferentes de zero de cada linha difere uma unidade em relação ao respetivo valor de uma Lista. Este predicado é evocado duas vezes, a primeira para *ListOfList* e *RowValues*, e a segunda para *Transpose* e *Colvalues*, garantindo que tanto o produto de uma linha como o de uma coluna cumprem a restrição da multiplicação enunciada anteriormente.

## 5 Visualização da Solução

Ao selecionar a opção de execução do nosso programa (se quer ver um dos problemas *default* ou resolver um problema gerado aleatoriamente), primeiro é mostrado um problema por resolver, ou seja, com os espaços dentro da matriz em branco. Este problema por resolver é obtido através do predicado **display**, que além de demonstrar a solução, faz também um *refresh* ao ecrã. Posteriormente, quando o utilizador pede a solução do problema, esta é apresentada graças ao predicado **displayWithoutClean**, que faz o mesmo que o predicado **display**, no entanto não faz um *refresh* ao ecrã.

Ambos os predicados mencionados anteriormente imprimem os valores das colunas inicialmente e imprimem recursivamente os valores da tabela juntamente com os valores por cada linha.

## 6 Experiências e Resultados

### 6.1 Análise Dimensional

Incluir exemplos de execução em instâncias do problema com diferentes dimensões e analisar os resultados obtidos.

Embora Wrong Products seja um problema apresentado sobre a forma de uma matriz quadrada de dimensão 4, resolvemos este problema de decisão dinamicamente, sendo possível fazer variar a dimensão da grelha e obter as soluções existentes, se possível.

Deste modo, executamos o problema com várias dimensões e verificamos que ...(acrescentar aqui a lgo após se terminaro projeto e fizer os gráficos)

Dimensao	Teste 1	Teste 2	Teste 3	Teste 4	Teste 5	Média
10	0.01	0.01	0	0.01	0.01	0.008
20	0.06	0.06	0.04	0.07	0.04	0.054
30	0.31	0.34	0.4	0.31	0.45	0.362
40	1.15	0.89	1.07	1.12	1.89	1.224
50	3.21	4.18	6.57	3.37	5.81	4.628
60	8.07	8.84	8.65	7.84	13.18	9.316
70	29.81	18.98	17.26	17.45	17.78	20.256
80	48.04	37	38.92	45.76	35.78	41.1

Figure 1: Tabela com Média de Tempos de Execução

### 6.2 Estratégias de Pesquisa

Devem ser testadas diferentes estratégias de pesquisa (heurísticas de escolha de variável e de valor), comparando os resultados obtidos. Devem ser



Figure 2: Grafico com Média de Tempos de Execução

usadas formas convenientes para apresentar os resultados (tabelas/gráficos).

Temos que criar um gerador e arranjar varias heurísticas que produzam uma solução

## 7 Conclusões e Trabalho Futuro

## 8 Referências

## 9 Anexo