Grupo Nº \_42\_



**Inteligência Artificial**

1.º Semestre 2014/2015

**Fill-a-Pix**

Relatório de Projecto

Índice

[1 Implementação Tipos e Representação Problema PSR 2](#_Toc404722800)

[1.1 Tipos Abstractos de Informação 2](#_Toc404722801)

[1.2 Representação do problema Fill-a-Pix como PSR 3](#_Toc404722802)

[2 Implementação Procuras e Funções Obrigatórias 4](#_Toc404722803)

[2.1 Fill-a-pix🡪psr 4](#_Toc404722804)

[2.2 Psr->Fill-a-pix 4](#_Toc404722805)

[2.3 Heurística de Grau 4](#_Toc404722806)

[2.4 Heurística MRV 4](#_Toc404722807)

[2.5 Procura-Retrocesso e Inferência 4](#_Toc404722808)

[3 Optimizações, Heurísticas e Técnicas adicionais utilizadas 5](#_Toc404722809)

[3.1 Optimizações especificas para o problema Fill-a-Pix 5](#_Toc404722810)

[3.2 Criação/Combinação de Heurísticas 5](#_Toc404722811)

[3.3 Utilização de técnicas adicionais 5](#_Toc404722812)

[4 Estudo Comparativo 6](#_Toc404722813)

[4.1 Critérios a analisar 6](#_Toc404722814)

[4.2 Testes Efectuados 6](#_Toc404722815)

[4.3 Resultados Obtidos 6](#_Toc404722816)

[4.4 Comparação dos Resultados Obtidos 6](#_Toc404722817)

[4.5 Escolha do resolve-best 6](#_Toc404722818)

# Implementação Tipos e Representação Problema PSR

## Tipos Abstractos de Informação

Neste ponto devem ser abordados, pelo menos, os seguintes tópicos:

•Descrição detalhada das estruturas de dados utilizadas no programa.

•Justificação das escolhas efectuadas em relação a possíveis alternativas e às operações implementadas.

Se implementarem mais do que um tipo psr para o projecto, deverão descrever os vários tipos psr implementados e justificar a sua implementação.

-- começa aqui --

Para implementarmos o tipo Restrição usámos apenas listas porque era um tipo bastante simples só com dois elementos

Para implementarmos o PSR, utilizámos uma lista para guardar os domínios de cada variável e outra lista para as várias restrições do problema. Utilizámos uma hash-table em que a chave é uma variável para guardar um par com a posição do domínio dessa variável na lista de domínios e o segundo elemento do par era o valor atribuído à variável. Utilizámos ainda outra hash-table em que a chave é novamente a variável e o valor guardado é uma lista com as restrições em que essa variável está inserida.

Relativamente ao uso de listas, utilizámos esta estrutura em detrimento, por exemplo de vectores por serem de fácil utilização e manipulação e aplicação de funções como mapcar e reverse.

As hash-tables foram escolhidas para podermos ter um acesso rápido utilizando as variáveis em vez de termos um vector ou lista e depois termos que percorrer a estrutura até a encontrarmos.

## Representação do problema Fill-a-Pix como PSR

Nesta secção deverão descrever como é que decidiram representar o puzzle Fill-a-Pix como um problema de Satisfação de Restrições. Ou seja, deverão indicar as variáveis, os domínios, e as restrições escolhidas para representar um puzzle Fill-a-Pix. Deverão também justificar a escolha efectuada, comparando com possíveis alternativas de representação.

-- começa aqui –

Para as variáveis, o nome escolhido é uma string com um inteiro entre 0 e N\*M-1 em que o N e o M são, o número de colunas e linhas da matriz do jogo. O domínio das variáveis é o intervalo entre 0 e 1 em que 0 representa uma posição da área de jogo que não está pintada, e a quando a variável tem o valor 1 é porque está pintada.

Um predicado fill-a-pix verifica se o número de casas preenchidas é superior ao valor pretendido (mesmo que algumas variáveis não estejam atribuídas) e o mesmo com o número de zeros, sendo o resultado então falso. Se não, devolve verdade se alguma variável não está atribuída e soma todas as variáveis, testando se o resultado é igual ao valor pretendido.

# Implementação Procuras e Funções Obrigatórias

Nesta secção deverão descrever a implementação das várias funções/heurísticas e mecanismos de inferência pedidos explicitamente no enunciado. No entanto, deverão apenas focar-se na descrição de particularidades que não estejam especificadas no documento de esclarecimento. Ou seja, **não queremos ler o pseudocódigo e respectivos comentários** criados por nós. Em particular devem descrever as seguintes funções/algoritmos:

## Fill-a-pix🡪psr

Deverão descrever brevemente o algoritmo de conversão de um tabuleiro Fill-a-Pix para o tipo PSR. Devido à sua complexidade, é importante descrever o processo de criação e teste de restrições.

## Psr->Fill-a-pix

Aqui deverão descrever o algoritmo oposto, i.e. a conversão de um PSR resolvido para um tabuleiro Fill-a-Pix.

## Heurística de Grau

Deverão descrever brevemente como é que implementaram a heurística de grau.

## Heurística MRV

Deverão descrever brevemente como é que implementaram a heurística MRV.

## Procura-Retrocesso e Inferência

Nesta secção deverão descrever algum detalhe que considerem relevante na implementação das procuras por retrocesso com os vários tipos de inferência. Mais uma vez, aqui não é pretendido que façam copy-paste do pseudocódigo, mas sim que descrevam algum pormenor que achem importante. Por exemplo, o que usaram para representar uma inferência, e como é que essa é aplicada/removida ao psr.

2.1

fill-a-pix->psr

Começa-se por criar todas as variáveis do psr. Um fill-a-pix tem n\*m variáveis (linha \* coluna). Para dar um nome único a cada variável, escolheu-se numerar as posições de 0 a (n\*m -1).

Para preencher o domínio, fez-se uma lista com n\*m elementos com domínio 0. Há medida que se encontra números no fill-a-pix, atualiza-se o domínio correto (0 1). Para isto foi necessário a implementação de um algoritmo que descobre todas as posições adjacentes validas. Este algoritmo cria uma lista com as 9 posições adjacentes e elimina as posições inválidas a seguir (posição negativa/posição que excede o numero máximo de linhas ou colunas).

Finalmente, para criar as restrições usou-se as mesmas posições adjacentes calculadas no domínio para as variáveis afetadas. Por fim, chegou-se à conclusão que um predicado fill-a-pix teria de verificar se o número de casas preenchidas é superior ao valor pretendido (mesmo que algumas variáveis não fossem atribuídas) e o mesmo com o número de 0s, sendo o resultado então falso. Se não, devolve verdade se alguma variável não está atribuída e soma todas as variáveis, testando se o resultado é igual ao valor pretendido.

2.2

Psr->fill-a-pix

Este algoritmo percorre as variáveis todas do fill-a-pix e preenche um array de dimensão linhas\*colunas. A primeira variável corresponde à posição (0 0) e a segunda à posiçao (0 1). Percorre-se então as variáveis todas do psr, e após ter preenchido m (numero de colunas) variáveis, incrementa-se o valor da linha.

Isto repete-se até o array estar preenchido.

2.3

Heurística-grau deve devolver a variável com grau maior, ou seja, a variável com mais restrições comuns com as outras variáveis não atribuídas.

Para tal, percorre uma lista de variáveis não atribuídas e procura por restrições comuns com todas as outras variáveis não atribuídas.

2.4

Heurística MRV

Herustica-mrv deve devolver a variável não atribuída com domínio menor. Este algoritmo percorre então a lista de variáveis não atribuídas e devolve a variável com menor elementos.

2.5

Procura-retrocesso e inferência

Criou-se um novo tipo, o tipo inferência que consiste numa lista com dois elementos - variável e domínio.

# Optimizações, Heurísticas e Técnicas adicionais utilizadas

## Optimizações especificas para o problema Fill-a-Pix

Pensou-se em implementar uma optimização para o resolve-best em que caso a soma dos valores das variáveis necessárias for igual ao número de variáveis desse predicado, sabe-se que todas essas casas vão ter o valor 1. Como tal, o domínio dessas variáveis não é um intervalo entre 0 e 1 mas sim 1. Assim, não é necessário criar um predicado para essa restrição. No entanto não chegámos a implementar esta funcionalidade.

## Criação/Combinação de Heurísticas

Não se implementou nenhuma herurística para além das listadas no enunciado.

## Utilização de técnicas adicionais

Na procura-retrocesso-best se o domínio da variável escolhida tiver apenas um elemento, ao invés de fazer um teste de consistência, atribui-se logo esse valor à variável.

# Estudo Comparativo

Pretende-se estudar a eficácia dos vários algoritmos e variantes implementados para a resolução de um problema Fill-a-Pix.

## Critérios a analisar

Descrever quais os critérios a ser usados para comparar as várias variantes e justificar a sua escolha.

Os critérios utilizados são o número de testes de consistência realizados para resolver o problema, o tempo que leva a terminar e a memória utilizada pelo algoritmo. Considera-se que estes três critérios são necessários para avaliar a eficiência do algoritmo, sendo que quanto menores forem os seus valores, melhor será a sua eficiência.

## Testes Efectuados

Descrever os problemas utilizados para efectuar o estudo. Justificar a sua *relevância* para os critérios a comparar.

Para comparar a procura-retrocesso-simples com a procura retrocesso-grau utilizámos os testes “e1\_1” e “e1\_2” presentes no ficheiro “exemplo.lisp”.

## Resultados Obtidos

Enumerar os resultados obtidos, sob a forma mais adequada (tabela e gráficos, se necessário).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algoritmo | Problema | Testes | Tempo (s) | Espaço (KB) |
| PRS | E1\_1 | 69 | 0,002 | 107 |
| PRG | E1\_1 | 77 | 0,006 | 524 |
| PRS | E1\_2 | 382 | 0,003 | 182 |
| PRG | E1\_2 | 59 | 0,007 | 318 |
|  | 7 |  |  |  |
|  | 7 |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algoritmo | Problema | Testes | Tempo (s) | Espaço (KB) |
| FC | E4 | 4301 | 0,08 | 2434 |
| MAC | E4 | 8577 | 0,15 | 4134 |

## Comparação dos Resultados Obtidos

Analisar, caso a caso, os resultados, focando a diferença de desempenho das várias variantes entre si. A análise não deve ser somente uma descrição dos resultados obtidos, mas sim uma análise dos factores que possam ter conduzido a esses resultados.

## Escolha do resolve-best

Com base nos resultados obtidos na comparação anterior, devem descrever e justificar quais as variantes escolhidas para implementar a vossa função resolve-best.