Guião Relatório IA 2022/2023 LEIC-A

Bom dia, vimos aqui apresentar um solucionador do jogo *Bimaru*. A nossa solução formaliza o jogo *Bimaru* como um **Problema de Satisfação de Restrições**.

Cada **estado** do tabuleiro é representado por uma grelha quadrada de dimensões 10 x 10 em que o estado inicial é lido do *standard input*. Mantemos ainda contadores do número de peças de barco e número de peças de água por linha e coluna e do número de barcos colocados no tabuleiro. Assim, as variáveis que tivemos em consideração foi cada posição do tabuleiro, sendo os domínios de todas, igual às letras que representam cada peça, tal como foi sugerido na representação externa do tabuleiro apresentado no enunciado. Para as variáveis por atribuir decidimos usar um ponto de interrogação (`?`) e para as variáveis que apenas têm peças de barco no domínio decidimos usar a letra `x`.

O objetivo do jogo é preencher totalmente o tabuleiro com peças, de tal forma que o número de peças de barco em cada linha ou coluna tem de ser precisamente igual ao número fornecido como input para essa linha ou coluna, têm também de existir quatro submarinos, três contratorpedeiros, dois cruzadores e um couraçado dispostos horizontal ou verticalmente, em que nenhum deles pode se encontrar adjacente a nenhum outro, nem mesmo na diagonal. Assim, sempre que é feita uma atribuição a uma das variáveis, a nossa solução apenas a aceita se não quebrar nenhuma das restrições mencionadas. Tendo em consideração que para estados intermédios apenas não se pode exceder o número de barcos e os números de peças por linha e coluna, mas é possível ter valores inferiores para esses contadores.

Uma **ação** é representada por um 4-tuplo, em que as primeiras duas entradas representam a posição de um extremo do barco, a terceira entrada representa o tamanho do barco e a última entrada representa a orientação do barco através da peça da ponta. O **resultado** de uma ação é atribuir as variáveis do tabuleiro com peças do barco numa cópia do estado, de modo a colocar esse barco no tabuleiro. Considerou-se a Heurística do Maior Grau, ou seja, quando se geram os sucessores de um estado damos sempre prioridade ao barco do maior tamanho possível que obedece ao número de barcos no tabuleiro, pois conseguimos atribuir valores concretos a mais variáveis que, por sua vez estão envolvidas num maior número de restrições do que no caso de se considerar um barco menor. Estas otimizações permitem reduzir o fator de ramificação da árvore de procura, diminuindo o custo de memória e tempo de computação desnecessário. Escolheu-se utilizar *forward checking* para unir a inferência com os algoritmos de procura. Após uma ação, é verificada a consistência das restrições no grafo de restrições e no caso de alguma variável ficar com domínio vazio é efetuado *backtracking*.

Como nenhuma das restrições é quebrada, então o **teste objetivo** pode ser feito em tempo constante, pois apenas temos de ver se todos os barcos foram utilizados, estando todas as outras posições implicitamente com água. A solução tenta sempre reduzir o tabuleiro o máximo possível em cada estado, ou seja, nas linhas e colunas em que o número de peças por colocar é igual ao número de variáveis por atribuir ou em que o número de peças por colocar é zero, atribui-se às variáveis restantes dessas linhas e colunas o único valor dos seus domínios que obedece a essa restrição, de modo a manter a consistência. Sempre que possível a nossa solução restringe o domínio das variáveis a partir das atribuições que vão sendo efetuadas, ou seja, propagamos as restrições quando, por exemplo, restringimos o domínio das variáveis que rodeiam outra à qual foi atribuída uma peça do topo de um barco. Isto possibilita a descoberta de novos barcos antes de ter de procurar posições para ele, o que reduz de forma significativa o fator de ramificação.

Gonçalo Bárias (103124) e Raquel Braunschweig (102624) Grupo **al010**