Programação em Lógica - Trabalho Prático Nº 2

Resolução de Problema de Decisão usando Programação em Lógica Com Restrições

João Fernando de Sousa Almeida - ei10099@fe.up.pt
e
Lázaro Gabriel Barros da Costa - up201405342@fe.up.pt

Faculdade de Engenharia da Universidade do Porto
Mestrado Integrado em Engenharia Informática e Computação
FEUP-PLOG, Turma 3MIEIC06, Grupo Tents_3

Resumo O objetivo principal deste trabalho é o desenvolvimento de um programa em Programação em Lógica com Restrições para a resolução do problema de decisão correspondente ao puzzle *Tents*.

U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Conteúdo

1 Introdução

O objetivo principal deste trabalho é o desenvolvimento de um programa em Programação em Lógica com Restrições para a resolução do problema de decisão correspondente ao puzzle *Tents*[PR1]. No puzzle mencionado, é dado um tabuleiro em grelha, com palmeiras colocadas em posições específicas, e cabe ao jogador colocar uma tenta junto de cada palmeira, mediante restrições específicas.

O puzzle apresenta ainda um detalhe adicional: a chave de resposta. Trata-se de um conjunto de dígitos que corresponde à área contígua máxima sem tendas, para para linha, e ordenado de cima para baixo. No caso de números de dois dígitos, introduz-se apenas o dígito das unidades.

No presente artigo apresenta-se o problema em estudo; detalha-se a abordagem adoptada ao desenvolvimento do programa; expõe-se a estratégia para a representação em texto da solução; e finalmente são colocadas as conclusões atingidas, assim como possibilidades de trabalho futuro.

2 Descrição do Problema - O Puzzle *Tents*

No puzzle *Tents*, é dado inicialmente um tabuleiro em grelha: este é quadrado e pode ter qualquer dimensão, tendo palmeiras colocadas em posições específicas. O objetivo do jogo é colocar uma tenta junto de cada palmeira, mediante restrições específicas. Na Figura 1 apresenta-se um exemplo de um tabuleiro inicial.

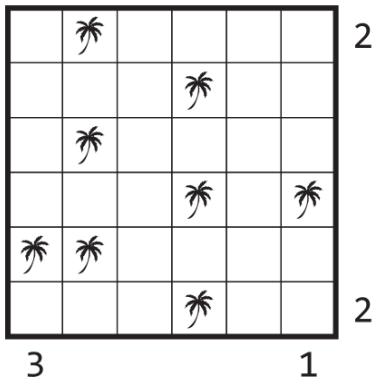

Figura 1. Exemplo de *Tents* inicial

Deve ser colocada uma tenda para cada árvore, numa posição adjacente vertical ou horizontalmente; as tendas não se podem posicionar de modo adjacente a outras tendas, em qualquer direção (nem diagonalmente). Certas linhas e colunas podem ter um número colocado no exterior do tabuleiro, este indica o número de

tendas requerido para a solução da correspondente linha ou coluna. Na Figura 2 apresenta-se o tabuleiro do puzzle correspondente à solução do apresentado na Figura 1.

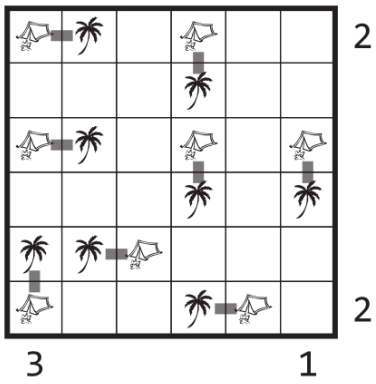

Figura 2. Exemplo de *Tents* resolvido

3 Abordagem

O problema em questão foi abordado como um Problema de Satisfação de Restrições (PSR). Nesta secção serão detalhados os correspondentes aspetos do problema.

3.1 Variáveis de Decisão

De modo a otimizar a resolução do problema, ao invés de considerar todas as posições do tabuleiro, optou-se por colocar numa lista (VarsEnd) as posições que têm palmeiras adjacentes, sendo que são essas em estudo. Após a inicialização (predicado init), obtém-se - na lista VarsEnd - o conjunto de variáveis onde serão aplicadas as restrições. O valor correspondente a cada posição do tabuleiro, na lista indicada, corresponde a zero caso a posição não contenha uma palmeira ou uma tenda, ou a um no caso alternativo.

3.2 Restrições

As restrições aplicadas sobre o problema correspondem às regras do puzzle, havendo assim uma correspondência entre cada regra e o(s) predicado(s) utilizado(s), tal como resumido na Tabela 1.

No que toca à primeira regra, utiliza-se o predicado *palmeiraAdj*. Sabendo que, no conjunto de posições à volta de uma palmeira (na vertical e horizontal)

Tabela 1. Regras - Predicados

Regra	Predicados
1. Tendas junto a cada palmeira (vertical ou horizontal)	*palmeiraAdj*
2. As tendas não se tocam em qualquer direcção	*tents2x2*
3. Os números fora do tabuleiro indicam o número de tendas na linha ou coluna	*Restrit*; *tentsLine*

deve haver uma tenda, com este predicado soma-se o valor correspondente às quatro células em questão. O resultado (sabendo que o valor de cada célula será zero caso não contenham tendas, ou uma unidade caso contenham) permite, verificando se é igual ou superior a uma unidade, saber se existe pelo menos uma tenda na vizinhança.

Quanto à segunda regra, faz-se recurso ao predicado *tents2x2*. A estratégia adoptada para verificar essa regra eficientemente foi a de efetuar um varrimento a todo o tabuleiro com quadrados 2*2, que apenas podem conter uma tenda.

Finalmente, de modo a verificar que o número de tendas na linha ou coluna que contém um número declarado fora do tabuleiro é consistente com esse valor, foram implementados os predicados *Restrit* e *tentsLine*. *Restrit* restringe o número de tendas por coluna à quantidade indicada exteriormente, e *tentsLine* trata da restrição equivalente, embora para cada linha.

3.3 Estratégia de Pesquisa

Utiliza-se a estratégia de pesquisa por defeito do Prolog, nomeadamente por backtracking. esta termina ao encontrar a primeira solução que satisfaça as restrições aplicadas. Para determinados casos, poderá existir mais do que uma solução - sendo o exemplo mais simples o de um tabuleiro com apenas uma palmeira, no qual existirão quatro soluções válidas: uma em cada direção.

4 Visualização da Solução

De modo a visualizar a solução obtida, são utilizados dois conjuntos de predicados: o primeiro obtém e apresenta a chave de resposta, e o segundo apresenta o tabuleiro completo. Na Figura 3 apresenta-se o *output* resultante da execução do programa implementado para o tabuleiro 6*6 providenciado no enunciado. Foi considerada a hipótese de se introduzir na interface de texto um indicador de relação tenda-palmeira tal como no enunciado, no entanto a opção foi posta de parte na medida em que acrescentaria uma camada adicional de dificuldade e seria também redundante na medida em que apenas observando o tabuleiro é trivial entender quais os objetos relacionados - tendo em conta as limitações das adjacências para as tendas e as palmeiras.

Apresenta-se de seguida os predicados correspondentes:

Chave de resposta:

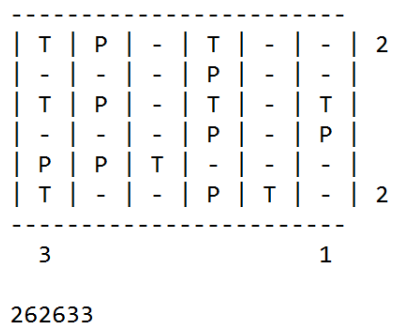

Figura 3. Exemplo de *Tents* resolvido

– *showResult*: utilizado para percorrer todas as linhas do tabuleiro, fazendo recurso do predicado auxiliar *showResultAux* para encontrar o maior valor contíguo de área sem tendas.

Apresentação do tabuleiro completo (predicados principais) :

– *showBoard*: invoca os predicados correspondentes às escritas em ecrã dos diversos componentes do tabuleiro;
– *showBoardLine* e showBoardCol: são utilizados para exibir o interior do tabuleiro, mostrando as separações entre as células e recorrendo ao predicado *showChar*;
– *showChar*: coloca no ecrã o caracter correspondente aos objetos no tabuleiro - 'P' para palmeira; 'T' para tenda e '-' para células vazias.

5 Conclusões e Trabalho Futuro

Durante o desenvolvimento do projeto, os seguintes obstáculos destacaram-se pelo tempo que foi necessário para os ultrapassar:

– Devido ao *backtracking* não ter sido considerado inicialmente, no caso de haver uma falha de restrição - numa situação em que existia uma chamada *write(")*, que se repetia quando uma condição de soma falhava, em vez de falhar o predicado
– Sendo que os estudantes não possuíam experiência na utilização de PLR, foi difícil a adaptação ao paradigma, que embora sendo Prolog, envolve uma mudança bastante significativa face, por exemplo, ao primeiro Trabalho Prático.
– A elaboração das restrições de modo a obter a solução pretendida apresentou também diversas dificuldades, e numa fase inicial houve alguma dificuldade nesse sentido.

Por outro lado, durante toda a elaboração do programa, foi possível ultrapassar as dificuldades e falta de experiência iniciais. Para além disto, os estudantes também entenderam melhor as vantagens de PLR para resolver problemas

de decisão deste tipo; sendo bastante interessante em termos computacionais a redução iterativa do número de variáveis em estudo, em comparação com a abordagem tradicional (não baseada em restrições). Desta forma, torna-se mais fácil a pesquisa de uma solução para o problema, sendo que se reduz consideravelmente as opções quanto aos valores das variáveis. Combinando o que foi referido com a eficiência do Prolog no processamento, resulta numa abordagem deveras apelativa para tratar problemas de decisão ou otimização.

6 Bibliografia

Referências

[PR1] K. Rajesh: Object Placement (Episode 8) (2016)