**Conclusão**

Com a conclusão deste segundo projeto, podemos concluir que a linguagem Prolog, mais especificamente os seus módulos de resolução com restrições são bastante poderosos na resolução de uma ampla gama de questões de decisão/otimização. Consideramos que este projeto nos ajudou bastante a interiorizar o conceito de Programação Lógica com Restrições (PLR), assim como todos os conceitos que lhe estão associados: varáveis de decisão, labelling, restrições, entre outros.

Os resultados obtidos são bastante satisfatórios, facilmente se observa que com o aumento da dimensão do puzzle temos uma estratégia cada vez mais complexa de resolução. Estes resultados também mostram que o nosso código está bastante eficiente, uma vez que, para os exemplos apresentados não ultrapassa 1 segundo em tempo de computação (apesar de nenhum dos puzzles apresentar uma dimensão demasiado elevada).

Num próximo trabalho poderíamos melhorar a resolução de puzzles gerados aleatoriamente, uma vez que agora ainda contém algumas falhas. Mas apesar disso estamos muito confiantes no nosso projeto, pois conseguimos apresentar boas soluções para qualquer dimensão de puzzle.

(resultados)

**Descrição do problema**

O desafio proposto para resolução é um problema de decisão, Bosnian Road. Este puzzle carateriza-se por o seu tabuleiro ser quadrado, ou seja, uma matriz com o mesmo número de linhas e de colunas. Antes de ser resolvido, a maior parte das casas encontram-se vazias, porém algumas possuem um número que pode variar entre 1 e 8. Este número está diretamente ligado à resolução do puzzle. Cada casa pode ter 8 casas adjacentes (cima, baixo, direita, esquerda, cima-esquerda, cima-direita, baixo-esquerda, baixo-direita), e esse número traduz o número dessas mesmas casas adjacentes que irão ser ocupadas pelo circuito. O objetivo deste problema é desenhar esse circuito de forma a respeitar todos os dados iniciais do problema e de forma a que cada célula do circuito esteja ligada, somente e apenas, a duas outras células do mesmo. Desta forma, a solução do puzzle deve ser um único circuito fechado, uma espécie de snake, onde as únicas partes que se tocam é a cabeça e a cauda.

**Abordagem**

O primeiro passo na abordagem foi modelar o puzzle como um problema de restrições. O grupo empenhou-se em perceber quais as variáveis de decisão a utilizar no predicado de labelling, a forma mais correta de restringir essas variáveis, como expandir o problema para um caso menos específico, métodos para testar os resultados obtidos e a forma mais intuitiva de interagir com o utilizador.

**Variáveis de decisão**

A solução do nosso puzzle passa por atribuir às restantes células do tabuleiro um estado: ora se encontram vazias, ora fazem parte do circuito. Deste modo, a variável S é a nossa única variável de decisão, caso não se trate de uma pista inicial esta variável apenas pode assumir o valor 0 ou 11, zero no caso de se encontrar vazia, 11 no caso de fazer parte o circuito, sendo este o domínio das variáveis de decisão.

**Restrições**

Para resolver o problema proposto desenvolveu-se um predicado chamado solve que aceita um tabuleiro inicial, o tamanho do mesmo, e retorna a solução. As restrições encontram-se todas implementadas no predicado restrict. Este predicado percorre todas as casas do tabuleiro alterando-as consoantes as restrições implementadas.

(código restrict)

De um modo geral são definidas duas grandes restrições: uma pista deve-se encontrar rodeada das respetivas casas do circuito; o circuito deve ser único e fechado. Estas restrições são respetivamente implementadas pelos predicados check\_surrounded e check\_closed.

O predicado check\_surrounded analisa cada célula individualmente. Caso essa célula se trate de uma pista utiliza o predicado get\_adjacent para ir buscar todos os valores de todas as células que se encontram à volta da célula em questão (cima, baixo, direita, esquerda, cima-esquerda, cima-direita, baixo-esquerda, baixo-direita). Depois basta verificar se o número de células com o valor 11 é exatamente igual à pista.

(surrounded)

Relativamente à restrição implementada no predicado check\_closed é necessário verificar que o circuito é fechado. Deste modo é verificada a valor das casas adjacentes, mas neste caso sem as diagonais, pois estas iriam levar à aceitação de circuitos inválidos. Cada casa deve, apenas e somente, possuir uma ligação a duas casas adjacentes (cima-baixo, cima-direita, cima-esquerda, etc.).

(closed)

**Visualização da Solução**

Para representar o tabuleiro foi utilizada uma lista inteiros. O número zero corresponde a uma casa vazia, enquanto que o número 11 corresponde a uma casa pertencente ao circuito. Além disso, para representar as pistas são utilizados os números entre 1 e 8, que representam o número de casas adjacentes á casa com a pista, que fazem parte do caminho.

(exemplo de um tabuleiro)

Na interface com o utilizador os valores das pistas permanecem os mesmos, mas os restantes são alterados para permitir uma melhor visualização da solução. O número 11 é transformado no caracter X enquanto que é zero é transformado num espaço. Esta transformação é feita através do predicado arrange\\_board.

(arrange)

Para imprimir o tabuleiro no ecrã é utlizado o predicado print\\_board, sendo que se recorre aos predicados print\\_row\\_by\_\row, print\\_line , print\\_space\\_line, print\\_black\\_line, para permitir um melhor alinhamento e visualização dos conteúdos do tabuleiro.

(print board)

Quando juntamos todos estes predicados com o predicado que resolve o puzzle em si, obtemos a seguinte solução para o puzzle da figura X.

(imagem)

**Resultados**

Quando um puzzle é resolvido, além da solução final, o nosso código também imprime no ecrã diferentes valores de estatísticas, nomeadamente: o tempo que o programa gastou a encontrar a solução, resumptions, entailments, prunings, backtracks e constraints.

Para recolher dados foram utilizados diferentes puzzles, previamente declarados no código fonte, com as dimensões 4x4, 6x6, 8x8 e 10x10. Como seria esperado, qualquer um destes valores conforme aumenta a dimensão do puzzle.

**Função de Avaliação**

Como o nosso puzzle não é um problema de otimização, não fazia sentido criar uma função de avaliação das soluções obtidas. Não sendo necessário comparar várias soluções para obter a melhor, não foi desenvolvido nada nesse sentido. Na nossa implementação a partir do momento que se cumprem todas as restrições, a solução encontrada é válida. Assim sendo, a nossa forma de avaliar o desempenho do programa é através de algumas estatísticas, por exemplo o tempo decorrido, o número de backtracks, o número de restrições, entre outras. Estas estatísticas são mostradas ao utilizador sempre que é resolvido um puzzle.

Estratégia de Pesquisa

A nossa estratégia de labeling etiquetagem tenta ser o mais otimizada possível. Após a obtenção de valores que conseguem ultrapassar todas as restrições implementadas, a nossa função de etiquetagem apenas computa o melhor valor. Isto é possível através da utilização de uma combinação de opções que em conjunto resulta muito bem. Esta combinação passa for ff, bissect e down. A opção ff , ou seja, first fail, faz label à variável mais à esquerda e com o menor domínio, para que seja possível detetar a falha o mais cedo possível. A opção bissect faz com que para cada variável, a escolha seja feita através do ponto médio do domínio, daí o nome bisseção. Por último, a opção down, tenta primeiro os elementos do domínio com o valor mais elevado.

Referências

Vários autores, The Prolog Library – SICStus Prolog, https://sisctis.sics.se/sisctus/docs/4.3.0/html/sicstus/The-Prolog-Lybrary.html , documentação dos predicados disponíveis nas várias bibliotecas do SICStus Prolog

Slides da disciplina sobre PLR, Henrique Lopes Cardoso (regente), <https://moodle.up.pt/course/view.php?id=638> consultado em dezembro de 2017

Ehud Sterling Leon, Shapiro (autor), The Art of Prolog, Second Edition: Advanced Programming Techniques (Logic Programming), 1994

Pierre Deransart, Abdel Ed-Dbali, Laurent Cervoni (autores), Prolog: The Standard:Reference Manual, Springer, 2007

Pagina com as Regra do Jogo, vários autores, http://logicmastersindia.com/lmitests/dl.asp?attachmentid=645&view=1