Resolução de Problema de Decisão usando Programação em Lógica com Restrições: C-Note

Carlos Lousada e José David Rocha

FEUP-PLOG, Turma 3MIEIC04, Grupo C-Note_5

Faculdade de Engenharia da Universidade do Porto,
Rua Dr. Roberto Frias, 4200-465, Porto, Portugal

Resumo. Este projeto foi desenvolvido no âmbito da unidade curricular de Programação Lógica [1], no Sistema de Desenvolvimento SICStus Prolog [2]. O objetivo é resolver um problema de decisão com implementação de restrições. O problema selecionado foi o C-Note [4], que consiste na adição de dígitos antes ou depois dos dígitos presentes em cada célula da grelha de modo a que a soma em cada coluna e em cada linha seja igual (originalmente, a 100). Desta forma, através da linguagem Prolog, foram criados um gerador e um solucionador destes puzzles, que serão abordados no decorrer deste artigo.

Keywords: C-Note, Problema de decisão/otimização, Prolog, SICStus, FEUP.

1 Introduction

O projeto foi desenvolvido no âmbito da unidade curricular de Programação em Lógica do 3º ano do curso Mestrado Integrado em Engenharia Informática e de Computação. Foi necessário implementar uma possível resolução para um problema de decisão ou otimização em Prolog, com restrições. O grupo escolheu o problema de decisão C-Note.

O problema escolhido consiste na adição de dígitos antes ou depois dos números presentes em cada célula da grelha do problema dado (ou gerado) de forma a obter soma igual à soma dada em todas as colunas e linhas.

Este artigo tem a seguinte estrutura:

- **Descrição do Problema:** descrição com detalhe o problema de otimização ou decisão em análise, incluindo todas as restrições envolvidas.
- **Abordagem:** descrição da modelação do problema como um PSR / POR:
    - **Variáveis de Decisão:** descrição das variáveis de decisão e respectivos 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 Dimensional:** inclusão de exemplos de execução em instâncias do problema com diferentes dimensões e analisar os resultados obtidos.
  - **Estratégias de Pesquisa:** inclusão de testes de diferentes estratégias de pesquisa (heurísticas de escolha de variável e de valor), comparando os resultados obtidos.
- **Conclusões e Trabalho Futuro:** conclusões retiradas deste projeto, resultados obtidos, vantagens e limitações da solução proposta e aspetos a melhorar.
- **Bibliografia:** fontes bibliográficas usadas, incluindo livros, artigos, páginas Web, entre outros, utilizados para desenvolver o trabalho.
- **Anexo:** ficheiros de dados e resultados, entre outros.

2 Descrição do Problema

C-Note é um problema de decisão. Este problema consiste numa grelha com números, na qual deverão ser adicionados dígitos à esquerda ou à direita do dígito presente, de forma a que todas as colunas e linhas da grelha tenham soma igual ao valor desejado (normalmente 100).

3 Abordagem

Para a resolução e processamento deste tipo de problemas, são utilizadas listas de listas, representativas da grelha do puzzle C-Note que foi introduzido pelo utilizador ou gerado, em que cada sublista representa uma linha. Exemplificando, a lista [[2,3,1],[3,4,1],[2,3,7]] representa o puzzle:

```
|------------|
|  2|  3|  1|
|------------|
|  3|  4|  1|
|------------|
|  2|  3|  7|
|------------|
```

3.1 Variáveis de Decisão

Ao inicializar o solver e ser recebida a grelha introduzida utilizador, é inicializada uma matriz (lista de listas) de variáveis com dimensões iguais às fornecidas. Em

seguida, serão aplicadas as restrições mencionadas no ponto seguinte e será feito o cálculo dos possíveis valores que, substituindo nas variáveis, constituiriam solução para o problema. Por fim, essa substituição será feita através do *labeling* numa lista com todas as variáveis de decisão que constituam a matriz criada inicialmente.

3.2 Restrições

Em problemas C-Note, a soma de todas as linhas e todas as colunas terá de ser igual (a 100, por definição). No programa desenvolvido, este valor é definido pelo utilizador. Além disso, os valores considerados para a possível solução, terão de conter o dígito fornecido em cada célula da grelha-puzzle inicializada anteriormente.
Relativamente à geração de problemas, apenas há a necessidade de aplicar a primeira restrição mencionada anteriormente. No entanto, após esta ser aplicada, haverá a necessidade de selecionar aleatoriamente o dígito que se encontre à esquerda ou à direita do valor de cada célula.

4 Visualização da Solução

O programa permite resolver os puzzles C-Note, e de forma a melhorar a visualização em modo de texto, tanto do problema proposto como da sua solução, foram implementados vários predicados.

Para iniciar a execução do programa, deverá ser inserido o predicado *solver(+Puzzle, +Sum)*, para resolver um problema dado por *Puzzle* (na forma de lista de listas) com a soma *Sum*, ou o predicado *generate(+Size, +Sum)*, para gerar aleatoriamente (através de random labeling [3]) um problema de tamanho *Size* com a soma *Sum*.

```
printElemTimes(0, _).
printElemTimes(Num, Char):-
    write(Char),
    NNum is Num-1,
    printElemTimes(NNum, Char).

printRow([],_):- write('|'),nl.
printRow([Elem|Tail], Digits):-
    write('|'),
    countDigits(Elem, EDigs),
    Times is Digits - EDigs,
    printElemTimes(Times, ' '),
    write(Elem),
    printRow(Tail, Digits).
```

```
printRows([], _,Num):- write('|'),printElemTimes(Num,'-'),write('|'),nl.
printRows([Row|Tail], Digits, Num):-
    write('|'), printElemTimes(Num,'-'),write('|'),nl,
    printRow(Row, Digits),
    printRows(Tail, Digits, Num).

printPuzzle(Puzzle, Sum):-
    countDigits(Sum, Digits),
    nth1(1,Puzzle,Row),
    length(Row, Int),
    Num is Int*(Digits+1)-1,
    printRows(Puzzle, Digits, Num).



generate(Size, Sum):-
    generatePuzzle(Size, Sum, Puzzle, Solution),
    write('Generated Puzzle: '),nl,
    printPuzzle(Puzzle, Sum),
    write('Solution: '), nl,!,
    once(printPuzzle(Solution, Sum)).

solver(Puzzle, Sum):-
    write('Puzzle: '), nl,
    printPuzzle(Puzzle, Sum),
    write('Solution: '), nl,!,
    solvePuzzle(Puzzle, Sum, Solution),
    once(printPuzzle(Solution, Sum)).
```

5 Resultados

Para se avaliar os resultados obtidos, foram feitas medições ao tempo de resolução, ao número de retrocessos e ao número de restrições criadas. Foram aplicadas as seguintes condições de teste e as respetivas conclusões:

- **Fez-se variar o tamanho da grelha, mantendo-se o valor da soma (Tabela 1, Fig. 1, Fig. 2 e Fig. 3 em Anexos).** O tempo de resolução do problema e o número de retrocessos variam exponencialmente com o aumento do tamanho da grelha, enquanto que o número de restrições criadas varia linearmente. Conclui-se, desta forma, que o tempo de resolução do problema depende do número de retrocessos.
- **Fez-se variar o valor da soma, mantendo-se o tamanho da grelha (Tabela 2, Fig. 4, Fig. 5 e Fig. 6 em Anexos).** Da mesma forma que nas condições

anteriores, conclui-se com estes dados que o número de retrocessos e o tempo de resolução do problema variam exponencialmente com o aumento da soma, enquanto que o número de restrições criadas varia linearmente. Confirma-se, mais uma vez, que o tempo de resolução do problema depende do número de retrocessos e não do número de restrições.

- **Fez-se variar as estratégias de labeling** [6]**, mantendo-se a soma e o tamanho da grelha (Tabela 3 em Anexos).** Destes dados conclui-se que a diferença entre as várias estratégias é pouco relevante, destacando-se, no entanto, que, em geral, a estratégia *[middle, anti_first_fail] e [median, anti_first_fail]* têm um peso negativo significativo na procura de solução.

6 Conclusões e Trabalho Futuro

Este projeto teve como objetivo aplicar o conhecimento da linguagem Prolog adquirido nas aulas teóricas e práticas da unidade curricular de Programação Lógica, mais concretamente, o módulo de restrições, que se provou útil para a resolução de problemas de decisão e otimização.

Durante o desenvolvimento deste projeto, foram encontradas certas dificuldades, principalmente na escolha das restrições e na forma de as implementar. Foi então necessária uma detalhada análise da biblioteca *clpfd* [5]e da visualização de vários programas-exemplo para o terminar o desenvolvimento deste programa.

Certos aspetos poderiam ser melhorados, principalmente a procura de um método mais eficiente para a resolução de problemas de maior complexidade, visto que para encontrar soluções para problemas com grande tamanho e/ou grande soma o tempo de execução supera alguns minutos ou até algumas horas.

Em suma, o projeto foi desenvolvido e terminado com sucesso, dado que soluciona corretamente o problema proposto. Para além disso, destaca-se também a sua contribuição positiva para melhorar a compreensão do *labeling* e de variáveis de decisão, tal como na aplicação de restrições em Prolog.

7 Anexos

Tabela 1. Variação do tamanho da grelha para uma soma de 100.

Soma	100			
Tamanho da Grelha	**Tempo(s)**	**Retrocessos**	**Restrições Criadas**	**Puzzle Input**
1	0,03	4	4	[[1]]
2	0,03	2	12	[[1,4],[4,1]]
3	0,05	265	268	[[2,9,2],[5,6,2],[9,3,6]]
4	0,66	102209	21459	[[7,1,7,2],[2,1,2,3],[1,2,2,5],[2,5,3,7]]
5	98,25	15785670	342946	[[9,2,1,2,1],[1,3,6,3,3],[1,9,4,4,2],[1,3,6,8,1],[3,9,3,2,3]]
6	8138,51	1090595724	26331667	[[8,1,1,2,1,1],[3,7,1,3,3,3],[7,4,7,6,2,1],[2,1,3,7,3,7],[2,2,9,3,4,1],[1,1,1,2,1,7]]

Tabela 2. Variação da soma para um tamanho de grelha 3.

Tamanho da Grelha	3			
Soma	**Tempo(s)**	**Retrocessos**	**Restrições Criadas**	**Puzzle Input**
10	0,03	0	15	[[8,1,1],[1,8,1],[1,1,8]]
25	0,04	23	51	[[8,1,1],[1,8,1],[1,1,8]]
50	0,05	94	81	[[7,6,7],[6,2,2],[7,3,2]]
100	0,07	249	256	[[4,4,7],[3,7,7],[4,4,1]]
250	0,07	621	517	[[1,5,5],[8,7,1],[2,8,6]]
500	0,4	12278	939	[[1,2,3],[8,8,1],[9,1,2]]
1000	2,14	60396	2122	[[4,5,5],[9,3,6],[4,1,4]]
2500	20,12	200464	4071	[[1,6,6],[3,3,4],[4,1,7]]
5000	178,05	1180097	5299	[[7,5,3],[6,9,5],[2,2,1]]

Table 3. Variação da estratégia de labeling para um tamanho de grelha 4 e soma 100.

Soma	100	Tamanho	4	[[2,9,2,7],[3,2,9,3],[4,7,9,4],[8,2,4,3]]				
	leftmost	**min**	**max**	**first_fail**	**anti_first_fail**	**occurrence**	**most_constrained**	**max_regret**
step	1,03	1,09	0,96	1,18	1,09	0,96	1,23	0,94
enum	1,12	1,09	1,23	2,12	1	1,19	1,64	1,06
bisect	1	1,19	1,19	1,11	1,56	0,88	1,12	0,95
middle	1,25	1,58	1,31	1,99	6,52	1,28	1,73	1,21
median	1,2	1,28	1,26	2,44	6,88	1,12	1,92	1,33

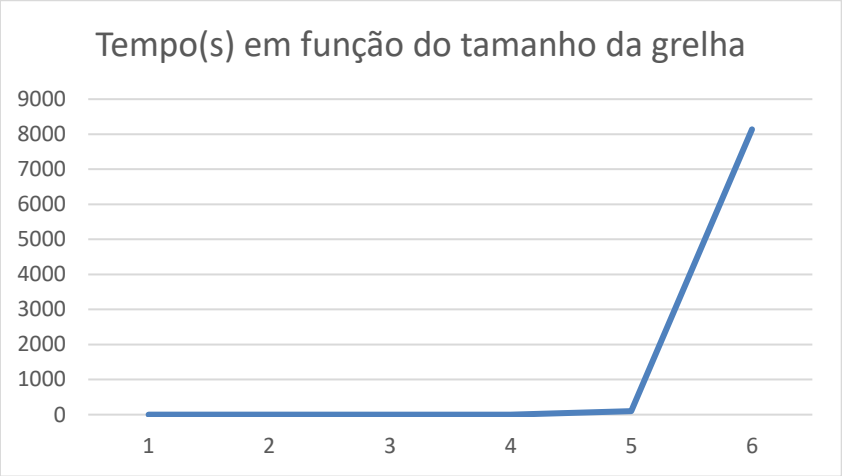

Fig. 1. Gráfico da variação do tempo em função do tamanho da grelha.

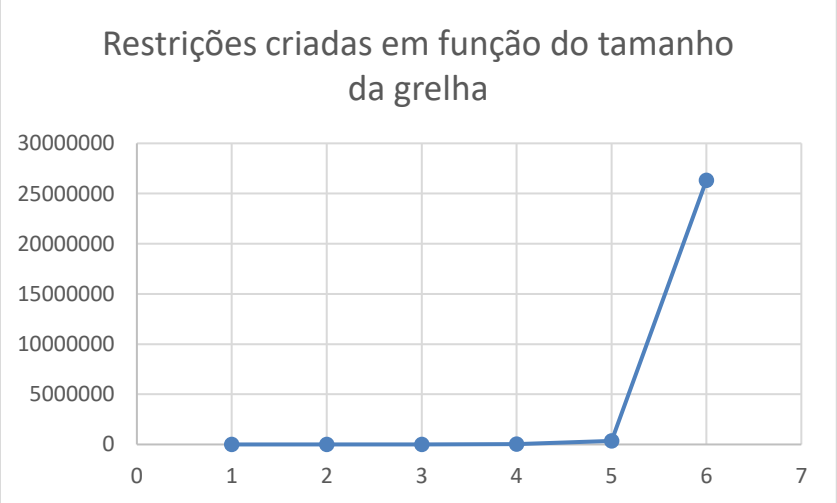

Fig. 2. Gráfico da variação das restrições em função do tamanho da grelha.

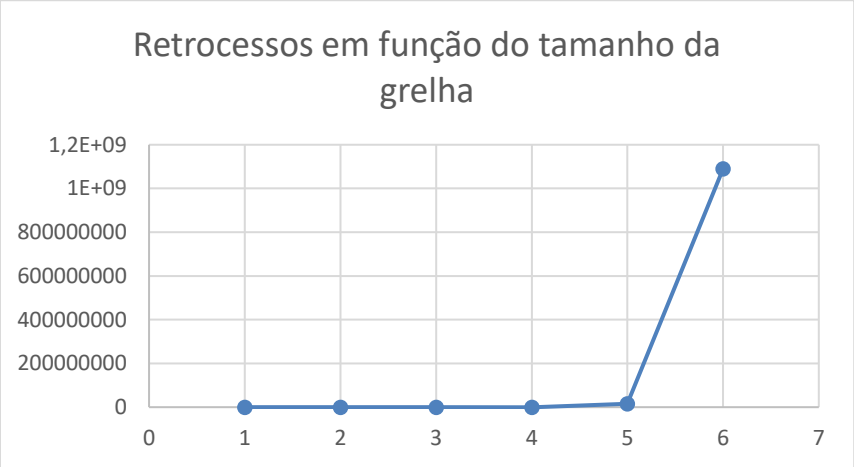

Fig. 3. Gráfico da variação dos retrocessos em função do tamanho da grelha.

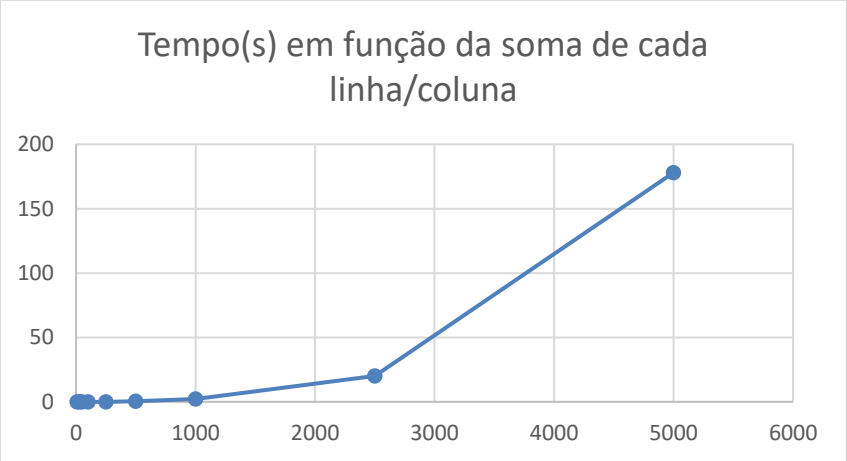

Fig. 4. Gráfico da variação do tempo em função da soma.

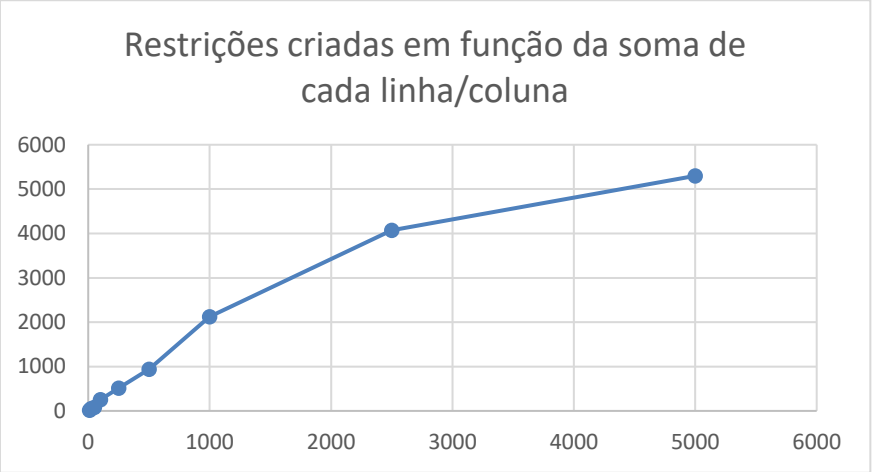

Fig. 5. Gráfico da variação das restrições criadas em função da soma.

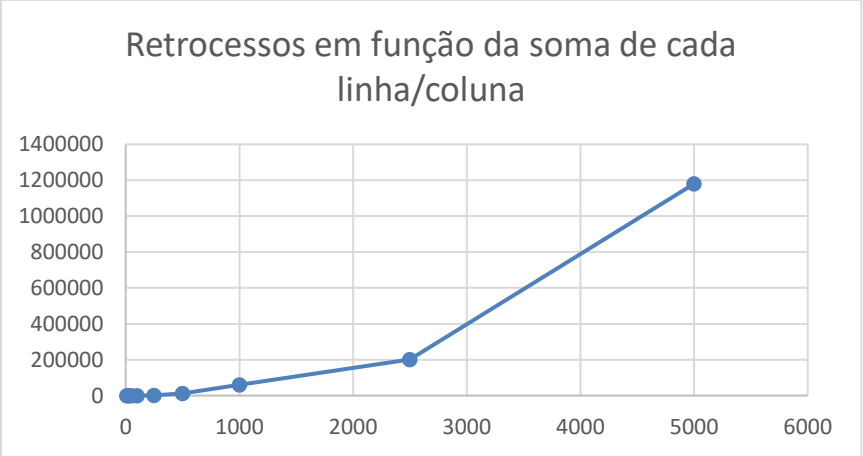

Fig. 6. Gráfico da variação dos retrocessos em função da soma.

References

1. sigarra.up.pt (2020), FEUP - Programação em Lógica, https://sigarra.up.pt/feup/pt/ucurr_geral.ficha_uc_view?pv_ocorrencia_id=459482
2. sicstus.sics.se (2020), SICStus Prolog Homepage, https://sicstus.sics.se/
3. Antunes, H. (2016), stackoverflow answer, https://stackoverflow.com/a/41211645
4. Friedman, E. (2011), C-Note Puzzles, https://erich-friedman.github.io/puzzle/100/
5. sicstus.sics.se (2020), SICStus Prolog: lib-clpfd, https://sicstus.sics.se/sicstus/docs/4.1.0/html/sicstus/lib_002dclpfd.html
6. sicstus.sics.se (2020), SICStus Prolog: Enumeration Predicates, https://sicstus.sics.se/sicstus/docs/4.1.3/html/sicstus/Enumeration-Predicates.html
7. moodle.up.pt (2020), Moodle UC: Programação em Lógica, https://moodle.up.pt/course/view.php?id=1476