

Obtenção de instâncias e resoluções de puzzles do Problema de Decisão Gold Star

André Gomes^[up201806224] e Gonçalo Teixeira^[up201806562]

FEUP-PLOG, Turma 3MIEIC06, Grupo Gold_Star_4
<http://web.fe.up.pt>

Resumo Começa-se por definir o objetivo deste projeto na Introdução, seguido de uma breve informação sobre cada ponto do relatório. Na descrição do problema explica-se em que é que consiste o puzzle Gold Star e como este se simplifica num sistema de equações. Na abordagem explicita-se as variáveis de decisão deste PSR juntamente com as restrições aplicadas aos operadores e às variáveis de resultado. Esta secção é seguida de uma breve demonstração de como é feita a visualização das soluções, em formato estrela e em formato de lista, antes da maior secção, as experiências realizadas. São primeiro demonstrados os resultados da análise dimensional e como o tempo de execução cresce exponencialmente com a complexidade da estrela, seguido pela análise de estratégias de pesquisa, nomeadamente os melhores argumentos usados no labeling. Terminando o relatório com uma conclusão e possível trabalho futuro.

Keywords: Configuração · Operadores · Gold Star

1 Introdução

Este relatório detalha o projeto desenvolvido para o segundo trabalho prático da unidade curricular de Programação em Lógica do MIEIC-FEUP, na qual se explora o puzzle Gold Star e como este pode ser resolvido com auxílio a programação em lógica por restrições. O objetivo inicial proposto para este trabalho foi de criar um ficheiro que contenha todas as soluções possíveis do puzzle Gold Star, partindo de uma estrela de cinco pontas. Não sabendo o grau de exigência computacional que este problema poderia ou não ter, tornou-se um desafio interessante da qual conseguimos retirar conclusões satisfatórias.

O relatório começa por descrever o problema estudado, passando para a abordagem tomada por nós para o resolver, contendo detalhes sobre as variáveis de decisão usadas e as restrições aplicadas. Após isto é brevemente mencionado como é feita a representação dos problemas e soluções, acabando com uma análise das experiências executadas, junto com as conclusões possíveis de retirar dos dados obtidos.

2 Descri o do Problema

O problema de decis o Gold Star consiste em resolver um sistema de 5 equa es da forma

$$\begin{cases} A_B = D_E \\ G_F = D_C \\ I_H = F_E \\ G_H = J_A \\ I_J = B_C \end{cases} \quad (1)$$

Sendo $_$ poss vel de ser substituído por um dos seguintes operadores:

$$+ \ - \ \times \ \div \quad (2)$$

Entre as 5 equa es, apenas s o partilhadas 10 vari veis para ser poss vel formar um padr o em forma de estrela ao dispor as equa es como na figura 1.

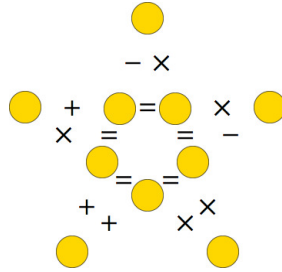


Figura 1. 5 equa es dispostas em padr o de forma estrela

Para obter uma solu o v lida, apenas podem ser usados os n meros inteiros de 0 at  9 e cada d gito s  pode ser usado uma vez. Para al m disso, os resultados de cada lado da equa o n o podem resultar em n meros n o inteiros.

3 Abordagem

Os puzzles Gold Star correspondem a um problema de decis o, ou um problema de satisfa o de restri es - **PSR**. Por isso, nos pr ximos 2 pontos s o explicadas as vari veis usadas, tal como os seus dom nios, e as restri es aplicadas que restringem os valores das vari veis dentro dos seus dom nios.

3.1 Vari veis de Decis o

Tendo como ponto de partida uma configura o do problema Gold Star como a da Fig.1,   poss vel obter uma lista de operadores que servem como argumento para o predicado `gold_star/2`.

O predicado `gold_star/2`, a partir de uma lista de operadores, consegue calcular uma solução para uma configuração, usando programação em lógica com restrições. Este predicado possui dez variáveis de decisão, uma para cada variável do sistema de equações, guardadas numa Lista `Result`, cujo domínio é $[0,9]$.

Para obter configurações do problema temos o predicado `operators/3`, que faz uso de restrições para criar configurações dos operadores do problema. Este predicado possui também 10 variáveis de decisão, cada uma correspondendo a um operador. O domínio é $[1,4]$, para que cada operador corresponda a um número inteiro, de acordo com os predicados `numb_signal/2`.

3.2 Restrições

Devido à simplicidade do problema, este não contém restrições flexíveis, apenas restrições rígidas, tanto para os operadores como para os operandos.

Operadores No predicado `operators(1,_,_)` é criada uma lista de operadores que pode ser alimentada ao predicado `gold_star/2`. Nesta criação é aplicada uma restrição explícita no predicado `bigger/2`:

Demonstração. O primeiro operador de uma configuração deve ter maior ou igual valor numérico que os restantes operadores

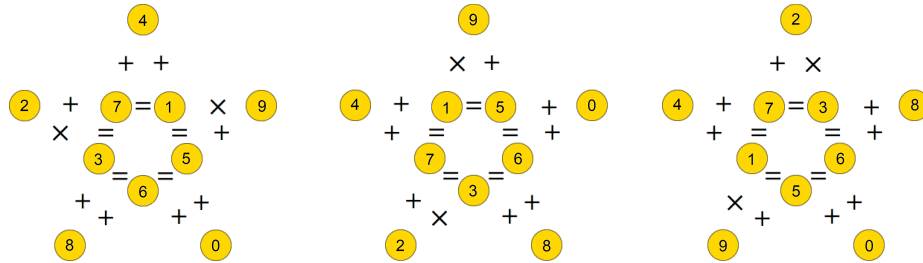


Figura 2. A estrela do centro tem a configuração $[3,1,1,1,1,3,1,1,1]$. A estrela da esquerda é obtida através de uma rotação da estrela do centro, formando a configuração $[1,1,3,1,1,1,1,1,3,1]$. A estrela da direita é obtida através de uma combinação de uma rotação e uma reflexão, resultando na configuração $[1,3,1,1,1,1,1,3,1,1]$

Na figura 2 apenas a configuração central obedece à restrição e só essa configuração é calculada. Todas as outras configurações, possíveis de obter através de deslocamentos de todos os elementos dentro da lista correspondem a aplicar rotações e reflexões na estrela original, o que faz com que seja desnecessário calcular estas configurações, já que são facilmente obtidas.

```

bigger( _, [] ).
bigger( Op1, [Op | Rest] ) :-
    Op1 #>= Op,
    bigger( Op1, Rest ).

```

Figura 3. Predicado `bigger/2` que recebe como primeiro argumento o primeiro elemento da lista a ser instanciado, e aplica a restri o `#>=` a todos os restantes elementos. A condi o termina quando n o houver mais operadores   qual se possa aplicar a restri o

Operandos Na primeira fase do projeto s o era poss vel solucionar uma estrela de cinco pontas. Isto facilitou a aplica o de restri es, j  que as  nicas restri es que eram necess rias correspondiam  s equa es da estrela (Fig. 11).

Com a adi o de funcionalidade dimensional foi necess rio refazer os predicados que atribuíam as restri es aos operadores. Verificou-se que, da forma como os operandos e os operadores s o declarados ¹ era poss vel achar uma forma de declarar restri es de forma iterativa.

Para saber as equa es que resultavam de uma estrela de n pontas era necess rio desenhar a estrela e ver as equa es uma a uma analisando cada ponta da estrela. Com o m todo achado,   poss vel criar um conjunto de «equa es direccionais» a partir de um tamanho de pontas e retirar facilmente da  o conjunto de equa es.

Para criar um conjunto de equa es direccionais a partir de um n mero de pontas, comea-se por definir a primeira e as duas  ltimas equa es, j  que estas s o semelhantes para qualquer n mero de pontas. Estas equa es s o lidas da esquerda para a direita. As restantes equa es (excepto para a estrela de 3 pontas que apenas possui 3 equa es) s o formadas usando os restantes operadores e operandos de acordo com a figura 12, lidas da direita para a esquerda. A ordem de leitura importa devido aos operadores `-` e `÷` que n o s o comutativos.

Este m todo traduz-se para prolog a partir dos predicados `first_restrictions/2` e `remaining_restrictions/2`. No primeiro predicado s o declaradas as restri es que correspondem  s 3 equa es comuns a todos e no segundo predicado s o aplicadas iterativamente o resto das restri es.

4 Visualiza o da Solu o

Para visualizar uma estrela de cinco pontas   poss vel usar o predicado `print_star/2` para imprimir a configura o e os resultados da estrela na consola do SICStus (Fig. 13).

Este desenho foi codificado   fora para ficar apelativo ao representar uma estrela de cinco pontas. Para estrelas com um n mero diferente de pontas se-

¹ Os operadores s o declarados, comeando no ponta do topo, primeiro o operador da esquerda, depois o da direita, continua-se no sentido hor rio para as outras pontas. Os operandos s o declarados, comeando no operador do topo da estrela, em sentido hor rio, passando para a base da pr xima ponta, depois para o topo da ponta e continua at  acabar.

ria necessário fazer uma função de visualização para cada. Como esse não é o objectivo deste projecto, estes predicados não foram elaborados.

Invés disso, para representar a configuração e a solução dessa configuração, as duas listas são impressas na consola, separadas por um espaço, na mesma linha, usando o predicado `print_result/2`. Esta forma é compacta e simples, componentes necessárias quando for necessário guardar várias destas listas num ficheiro.

5 Experiências e Resultados

As seguintes experiências podem resultar em tempos diferenças se forem executados em máquinas diferentes. Todos os resultados correspondem á mesma máquina.

5.1 Análise Dimensional

Na primeira experiência realizada testaram-se combinações de execução para estrelas de cinco pontas, consistindo em variações do uso de restrições e uso de `! (cut)` no final do predicado de resolução para achar apenas uma ou todas as soluções para uma configuração. Originando os resultados da tabela 1.

Tabela 1. Resultados para a estrela de 5 pontas, com erro na restrição de divisão

ID Predicado	Nº de Resultados	Tempo de execução
1. <code>Print_restricted_one_sol</code>	44535	197,672 seg
2. <code>Print_restricted_all_sol</code>	751380	283,062 seg
3. <code>Print_unrestricted_one_sol</code>	132759	563,547 seg
4. <code>Print_unrestricted_all_sol</code>	1516944	733,031 seg

Nesta primeira experiência não se reparou no erro da restrição de divisão comentada nas linhas finais da figura 11. Com esta restrição errada foram criadas soluções inválidas, mas foi possível obter uma noção do tempo de execução de cada predicado, apontando numa boa direção, já que seria possível chegar ao objetivo proposto na introdução.

Tendo corrigido esta restrição obteve-se os resultados da tabela 2 e o gráfico da figura 5.

A partir daqui obteve-se o tempo de execução de cada predicado e o número de resultados obtidos. A primeira conclusão que se retira destes dados é que o número de resultados obtidos é imensamente inferior ao esperado, vendo o caso da estrela de 5 estrelas, como contém 10 operadores, cada um tomando 1 de 4 valores possíveis, significa que é possível formar 4^{10} configurações² de operadores, mas ao verificar os resultados de `5-unrestricted-one`, apenas foram

² 1 048 576

Tabela 2. Resultados de pesquisa de solu es para estrelas de diferentes dimens es com backtracking indesejado

Tips	Restriction	Solutions	Seconds	Records
3 restricted	one		0,047	23
	all		0,047	132
unrestricted	one		0,172	76
	all		0,172	264
4 restricted	one		1,765	82
	all		1,813	239
unrestricted	one		7,453	397
	all		7,473	738
5 restricted	one		61,844	522
	all		62,203	781
unrestricted	one		336,515	3 567
	all		339,359	5 071
6 restricted	one sol		2142,828	4 996
	all sol		2199,500	15 886
unrestricted	one sol		unfinished	27913...
	all sol		not_run	—

encontrados 3567 resultados, correspondentes a 0,34% de todas as combina es poss veis.

Devido a mais um lapso de aten o, ap s o **labeling** n o encontrar solu o, estavam a ser reescritas algumas restri es no processo de **redo**. Com isto corrigido a experi ncia foi refeita, mas os resultados, na tabela 6 e gr fico da figura 4, n o foram muitos diferentes.

A partir destes resultados   poss vel verificar a melhoria que a aplica o de restri es na determina o dos operadores causou. Em m dia, com o uso de restri es, o programa tomou 25,35% do tempo que demoraria na sua contraparte sem restri es, com tend ncia a diminuir com o aumento de pontas da estrela. Para al m disso, com o uso de restri es, foram achadas em m dia 27,22% dos resultados que se obteriam sem restri es. Uma mudana significativa que tende a aumentar com o n mero de pontas da estrela.

Ao analisar o gr fico (Fig. 5) resultante da tabela 2 verifica-se que o tempo que demora a calcular todas as solu es cresce exponencialmente com o aumento do n mero de pontas da estrela, tal como o n mero de resultados, proveniente do gr fico da figura 6.

Com os resultados de 5-**unrestricted-all** alcanou-se o objetivo proposto na introdu o deste relat rio de obter um ficheiro que contenha todas as solu es poss veis deste puzzle. Na busca deste ficheiro obteve-se tamb m todas as solu es poss veis para estrelas de 3 e 4 pontas.

5.2 Estrat gias de Pesquisa

Ap s analisar o problema durante as experi ncias anteriores, tentou-se criar um predicado **variable(Sel)** para usar como argumento de **labeling**, baseado

nas suposições de que, ao começar por atribuir valores às variáveis que estão afetadas por divisões, seria possível eliminar muitos dos valores do domínio, já que a divisão tem de resultar num número inteiro e que como a operação de divisão não é comutativa, seria mais fácil eliminar possíveis valores porque da mesma forma que a anterior, têm de dar resultados inteiros.

Após testar o predicado acima, não se obteve resultados positivos (Tab. 7 e Fig. 7), já que o cálculo demorou em média mais 60% que o uso de labeling sem argumentos. Com isto em mente decidiu-se testar todas as combinações possíveis de argumentos de labeling, tanto para a formação de configurações de operadores como para o predicado que tenta achar as soluções para as configurações.

Encontra-se no Anexo A as tabelas 4 e 5 que contêm os resultados de testar todas as configurações possíveis de heurísticas para determinar qual delas realizava uma pesquisa mais rápida. Também no Anexo B se encontram as figuras 9 e 10 com os gráficos relativos às tabelas anteriores.

No caso dos operadores não ocorre muita diferença entre os argumentos usados, sendo que a configuração mais lenta (ffc, middle, down) demorou apenas mais 3,8 segundos que a mais rápida (occurrence, enum, up). Já no caso dos operandos nota-se diferenças significativas entre os diferentes argumentos. A configuração mais rápida (min, middle, up) demorou 36,406 segundos enquanto que a mais lenta (occurrence, step, down) demorou 161,672 segundos, a configuração predefinida (leftmost, step, up) que corresponde ao labeling sem argumentos, demorou 81,609 segundos e das configurações com um argumento `variable(Sel)`, a mais rápida demorou 104,469 segundos.

Conclui-se daqui que a configuração mais rápida demorou apenas 22,5% do tempo da mais lenta e 44,6% da predefinida. Melhorias significativas, tendo em conta que o tempo de execução aumenta exponencialmente com o número de pontas da estrela.

Com estas novas heurísticas descobertas fez-se uma última bateria de testes para verificar a diferença de tempos usando as melhores combinações de argumentos, resultando nos valores da tabela 3 e no gráfico da figura 8.

Estes resultados tomaram em média 66,9% do tempo que os resultados ao usar labeling sem argumentos (tabela 2). Não tendo em conta os resultados das estrelas de 3 pontas, que são obtidos quase instantaneamente, e por isso não foram muito afetados pela heurística, este valor desce para 52,3%.

Tabela 3. Tabela com resultados da execu o de diferentes predicados usando a melhor combina o de heur sticas para a gera o de operadores e para a resolu o de puzzles

Tips	Restriction	Solutions	Seconds	Records
3 restricted	one	0,047	23	
	all	0,062	132	
unrestricted	one	0,141	76	
	all	0,172	264	
4 restricted	one	1,203	82	
	all	1,281	239	
unrestricted	one	4,641	397	
	all	4,718	738	
5 restricted	one	32,000	522	
	all	32,204	781	
unrestricted	one	138,093	3 567	
	all	140,016	5 071	
6 restricted	one	796,078	4 996	
	all	803,734	15 886	

6 Conclus es e Trabalho Futuro

Este projeto serviu para demonstrar que um problema que parece simples numa primeira vista consegue conter muita informa o poss vel de analisar, que foi demonstrado pela an lise dimensional e pela an lise de estrat gias de pesquisa.

Gostar amos de real ar o m todo exemplificado na sec o 3.2, que n o foi f cil de descobrir e que possibilitou ignorar o conceito de equa es em formato de estrela para poder formar os predicados que aplicam as restri es aos operandos, apenas baseando-se nas posi es dos operadores e operandos nas suas listas respectivas.

Por fim, deixamos como trabalho futuro uma restri o que n o conseguimos criar em prolog durante o desenvolvimento deste projeto:

Com a restri o **bigger** s o calculadas configura es a mais que equivalem   mesma configura o com um deslocamento em todos os operadores, por exemplo:

$$\begin{aligned}
&[-, +, -, -, -, -][2, 3, 4, 5, 0, 1] \\
&[-, -, +, -, -, -][1, 2, 3, 4, 5, 0] \\
&[-, -, -, +, -, -][0, 1, 2, 3, 4, 5] \\
&[-, -, -, -, +, -][5, 0, 1, 2, 3, 4] \\
&[-, -, -, -, -, +][4, 5, 0, 1, 2, 3]
\end{aligned}$$

Apenas uma destas configura es seria necess ria de calcular, mas n o conseguimos achar um m todo que filtre correctamente as configura es de acordo com o pretendido.

A Tabelas de dados

Tabela 4: Resultados de testar todas as combinações de heurística possíveis para encontrar configurações de operadores

Next Var	Next Value	Value Choice	Seconds
anti_first_fail	bisect	down	24,141
		up	24,828
	enum	down	23,703
		up	25,438
	median	down	23,438
		up	24,14
	middle	down	23,484
		up	23,75
step	down	25,984	
	up	24,266	
ff	bisect	down	24,312
		up	25,235
	enum	down	24,312
		up	24,375
	median	down	23,469
		up	24,031
	middle	down	23,375
		up	23,703
step	down	24,297	
	up	23,984	
ffc	bisect	down	24,453
		up	23,797
	enum	down	23,281
		up	24,844
	median	down	23,75
		up	23,89
	middle	down	26,859
		up	25,188
step	down	24,219	
	up	24,031	
leftmost (default)	bisect	down	24,234
		up	24,063
	enum	down	23,39
		up	23,219
	median	down	23,718
		up	23,782
	middle	down	23,797
		up	23,953
Continua na próxima página			

Tabela 4 – Continua o da p gina anterior

Next Var	Next Value	Value Choice	Seconds
max	step (default)	down	23,719
		up(default)	23,531
	bisect	down	23,922
		up	24
	enum	down	23,657
		up	23,531
	median	down	24,75
		up	23,453
	middle	down	25,485
		up	25,062
	step	down	23,75
		up	24,579
max_regret	bisect	down	25,859
		up	25,672
	enum	down	25,688
		up	25,625
	median	down	25,796
		up	25,985
	middle	down	25,266
		up	25,563
	step	down	25,953
		up	26,36
min	bisect	down	25,594
		up	24,703
	enum	down	25,734
		up	25,578
	median	down	23,937
		up	24,641
	middle	down	24,406
		up	24,547
	step	down	24,656
		up	24,157
occurrence	bisect	down	24,813
		up	25,125
	enum	down	24,218
		up	23,063
	median	down	23,516
		up	24,812
	middle	down	24,328
		up	23,859
	step	down	23,469
		up	23,531

Tabela 5: Resultados de testar todas as combinações de heurística possíveis para encontrar soluções, dado uma configuração

Next Var	Next Value	Value Choice	Seconds
anti_first_fail	bisect	down	62,344
		up	58,453
	enum	down	124,766
		up	125,172
	median	down	80,688
		up	50,015
	middle	down	80,953
		up	49,969
	step	down	79,656
		up	49,125
ff	bisect	down	74,203
		up	74,031
	enum	down	80,391
		up	80,547
	median	down	86,297
		up	67,281
	middle	down	85,859
		up	67,094
	step	down	80,922
		up	61,781
ffc	bisect	down	100,718
		up	100,266
	enum	down	109,719
		up	110,093
	median	down	128,781
		up	91,641
	middle	down	128,328
		up	94,907
	step	down	123,547
		up	92,078
leftmost (default)	bisect	down	82,391
		up	81,64
	enum	down	98,282
		up	98,609
	median	down	104,969
		up	82,312
	middle	down	103,469
		up	81,969
	step (default)	down	104,219
		up(default)	81,609
Continua na próxima página			

Tabela 5 – Continua o da p gina anterior

Next Var	Next Value	Value Choice	Seconds
max	bisect	down	93,781
		up	93,219
	enum	down	112,406
		up	112,766
	median	down	67,984
		up	90,781
	middle	down	67,969
		up	90,625
	step	down	67,875
		up	90,875
max_regret	bisect	down	82,093
		up	82,344
	enum	down	101,203
		up	101,157
	median	down	107,594
		up	75,578
	middle	down	105,922
		up	76,187
	step	down	108,062
		up	76,156
min	bisect	down	72,485
		up	71,937
	enum	down	99,891
		up	107,109
	median	down	102,875
		up	38,234
	middle	down	102,813
		up	36,406
	step	down	109,625
		up	39,156
occurrence	bisect	down	129,125
		up	127,813
	enum	down	152,375
		up	153,453
	median	down	153,844
		up	128,421
	middle	down	152,125
		up	121,735
	step	down	161,672
		up	129,734
variable()	bisect	down	110,578
		up	111,11
Continua na próxima página			

Tabela 5 – Continuação da página anterior

Next Var	Next Value	Value Choice	Seconds
	enum	down	107,484
		up	105,453
	median	down	141,11
		up	108,937
	middle	down	141,141
		up	108,515
	step	down	136,156
		up	104,469

Tabela 6. Resultados de pesquisa de soluções para estrelas de diferentes dimensões sem argumentos no labeling

Tips Restriction	Solutions	Seconds	Records
3 restricted	one	0,063	23
	all	0,047	132
unrestricted	one	0,152	76
	all	0,172	264
4 restricted	one	1,750	82
	all	1,703	239
unrestricted	one	7,187	397
	all	7,203	738
5 restricted	one	61,297	522
	all	61,453	781
unrestricted	one	341,469	3 567
	all	348,125	5 071
6 restricted	one	2049,312	4 996
	all	2099,297	15 886
unrestricted	one	13901,078	34 346
	all	unfinished	26669...

Tabela 7. Resultados de execu o com uso de heur stica `select()` no labeling do predicado `gold_star`

Tips	Restriction	Solutions	Seconds	Records
3 restricted	one	0,203	23	
	all	0,172	123	
unrestricted	one	0,547	75	
	all	0,562	247	
4 restricted	one	4,906	82	
	all	5,016	194	
unrestricted	one	16,968	351	
	all	17,079	634	
5 restricted	one	142,640	462	
	all	140,953	650	
unrestricted	one	563,532	2 822	
	all	563,234	3 739	

B Gráficos

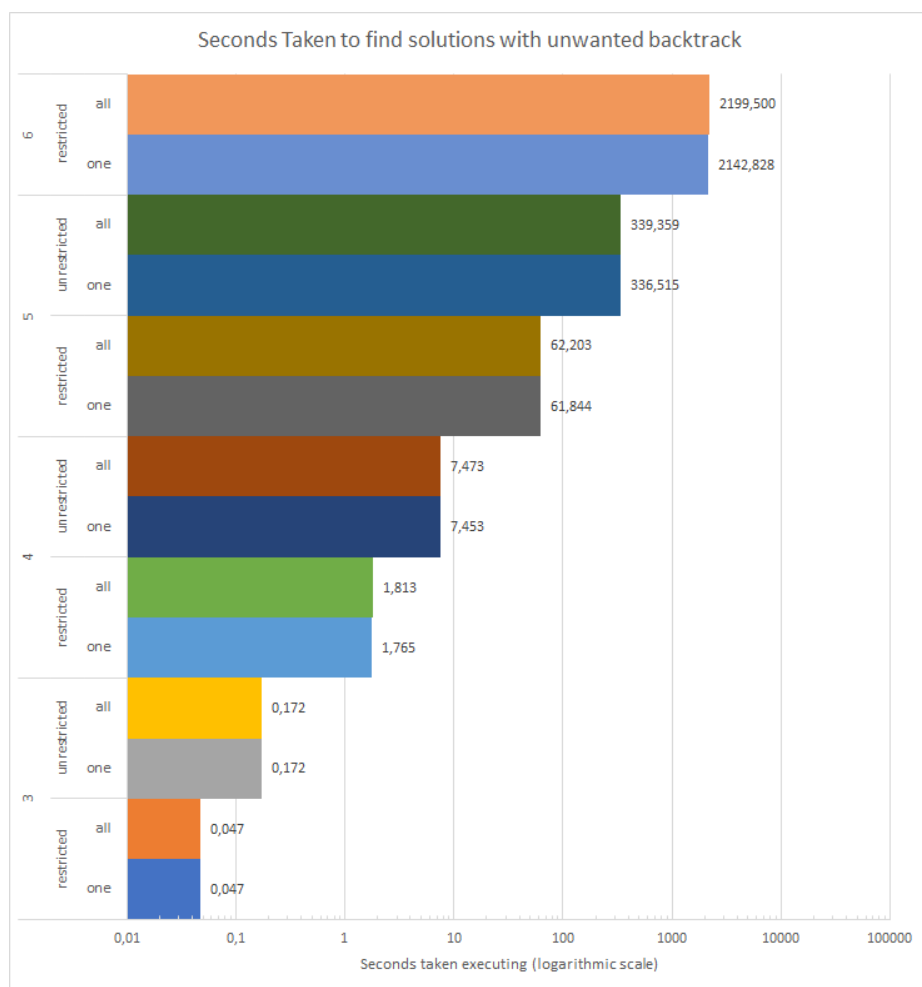


Figura 4. Gráfico correspondente aos dados da tabela 2 com resultados de execução com backtracking indesejado

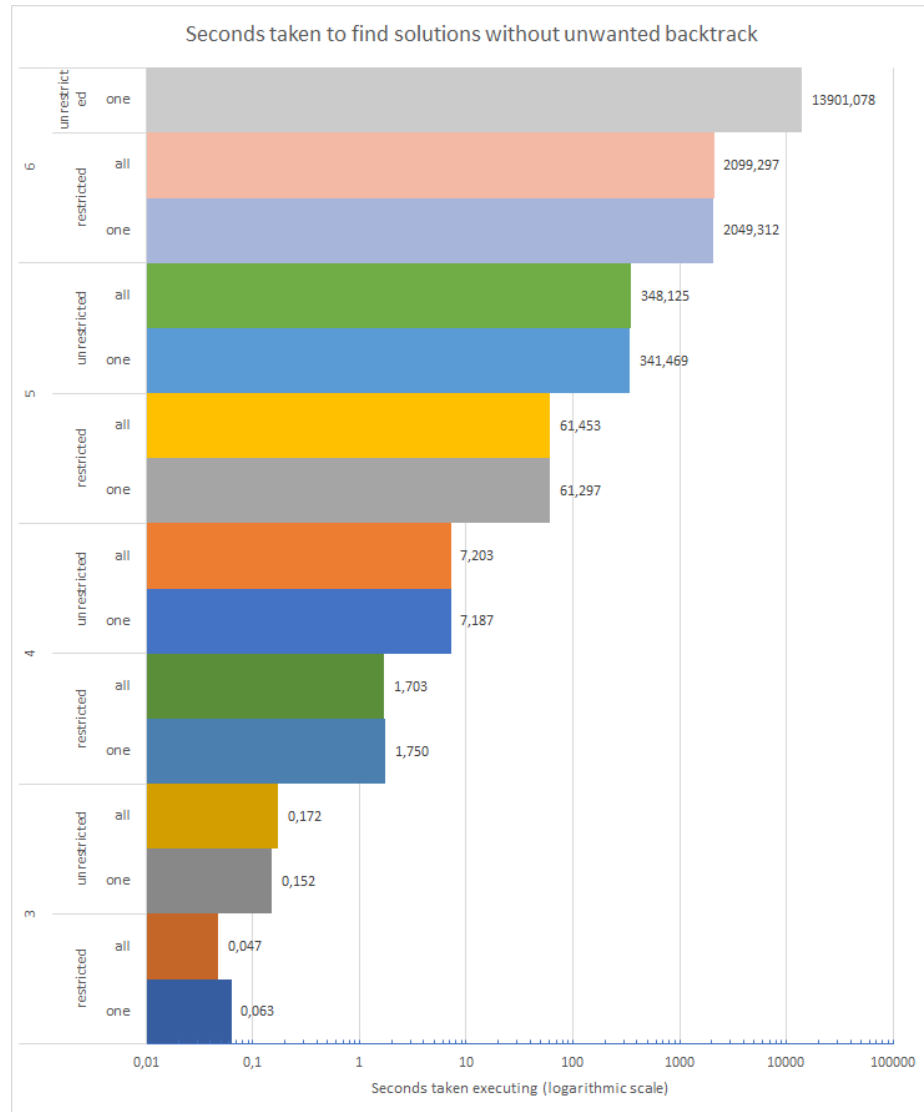


Figura 5. Gráfico correspondente aos dados da tabela 6 com resultados de execução sem backtracking indesejado

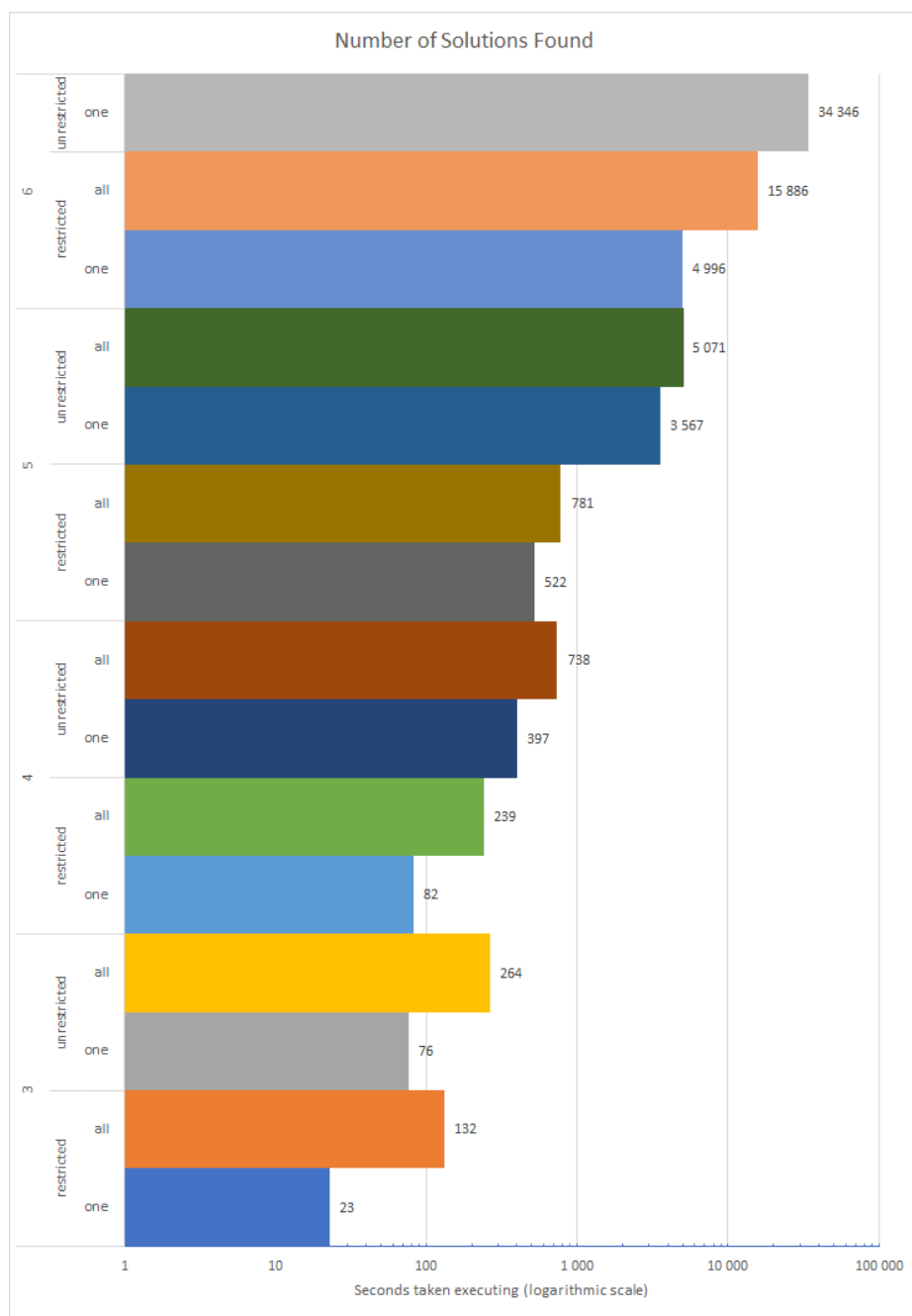


Figura 6. Numero de soluções encontradas para cada tipo de pesquisa. Estes números são comuns a todas as experiências

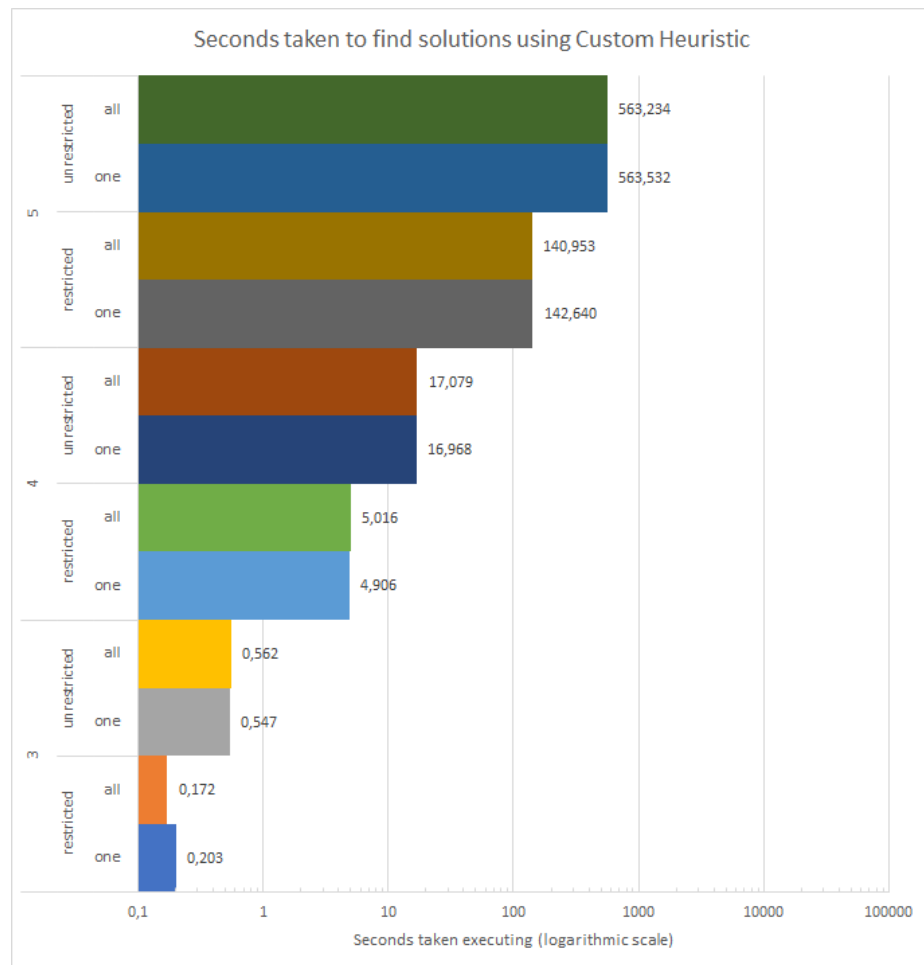


Figura 7. Gráfico com resultados da tabela 7 correspondentes aos dados obtidos com uso de uma heurística variable(Sel)

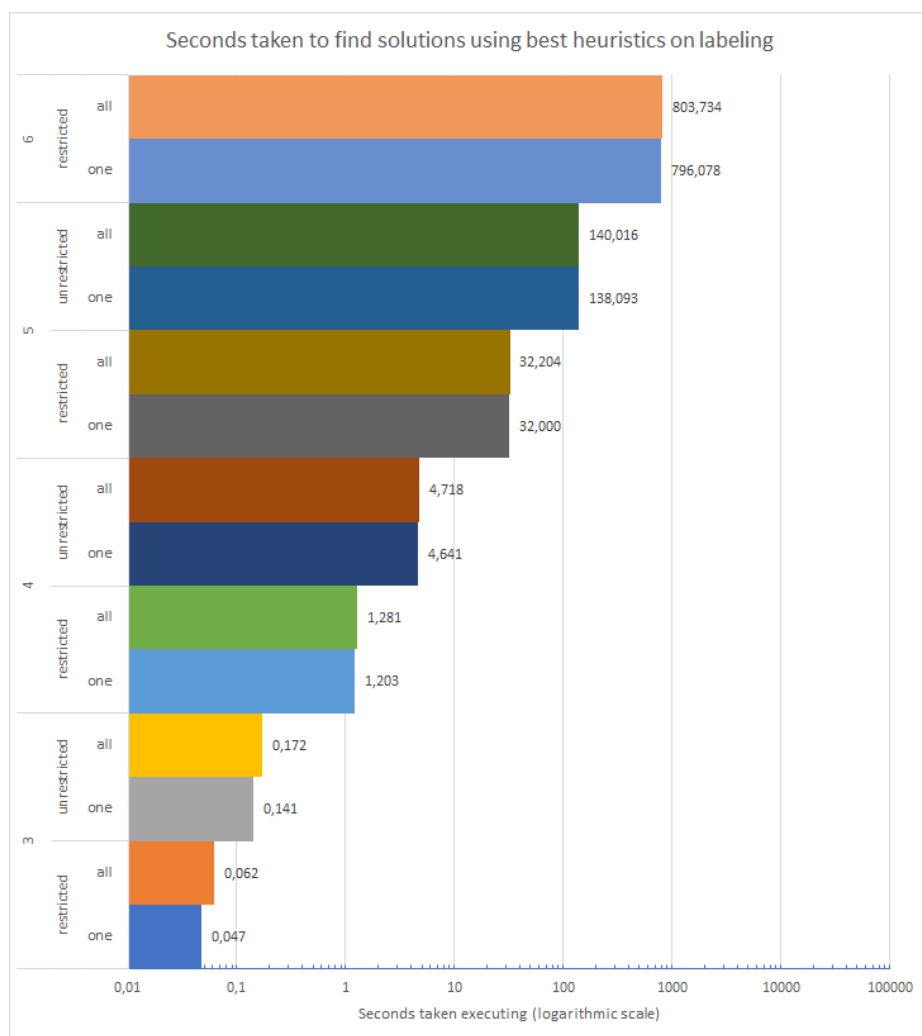


Figura 8. Gráfico com resultados da tabela 3 correspondentes ao uso da melhor combinação heurística

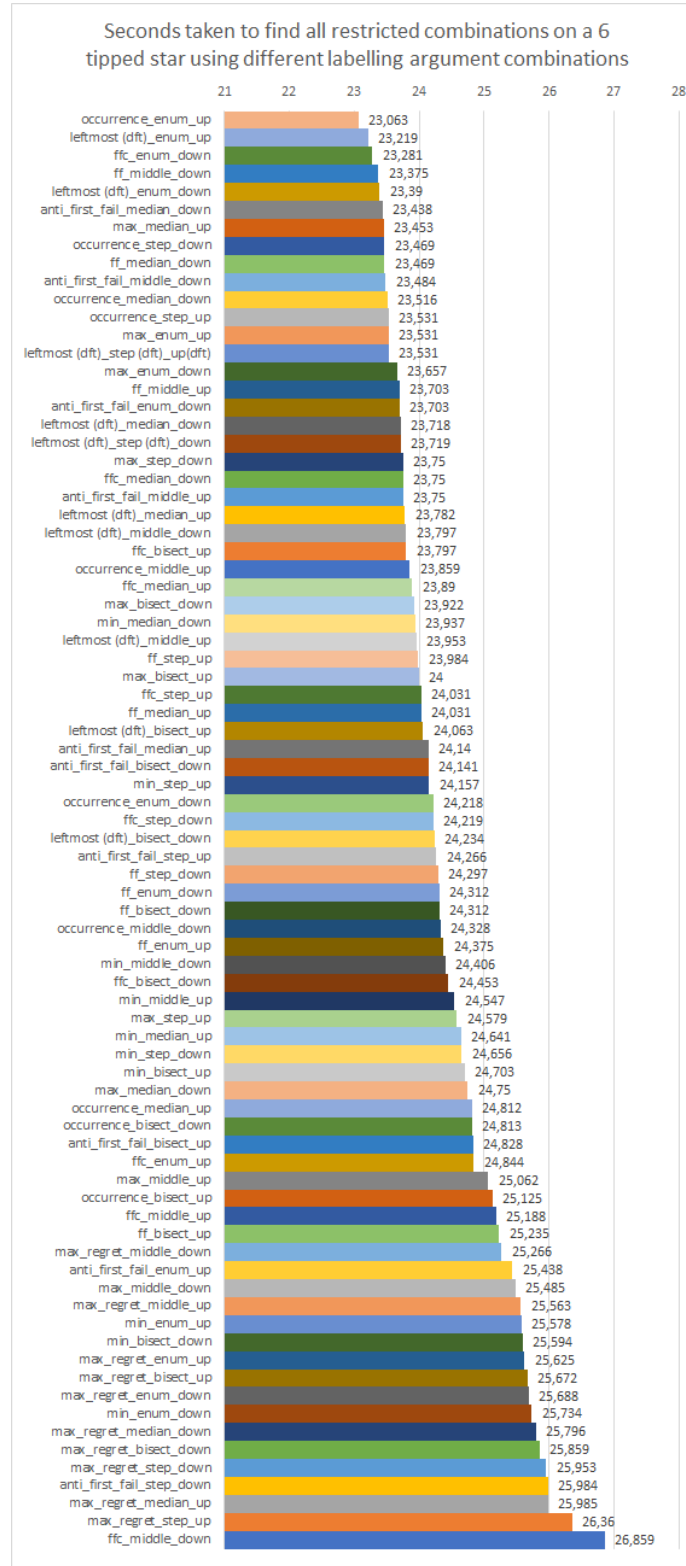


Figura 9. Resultados de teste de diferentes heur sticas no labeling de operadores

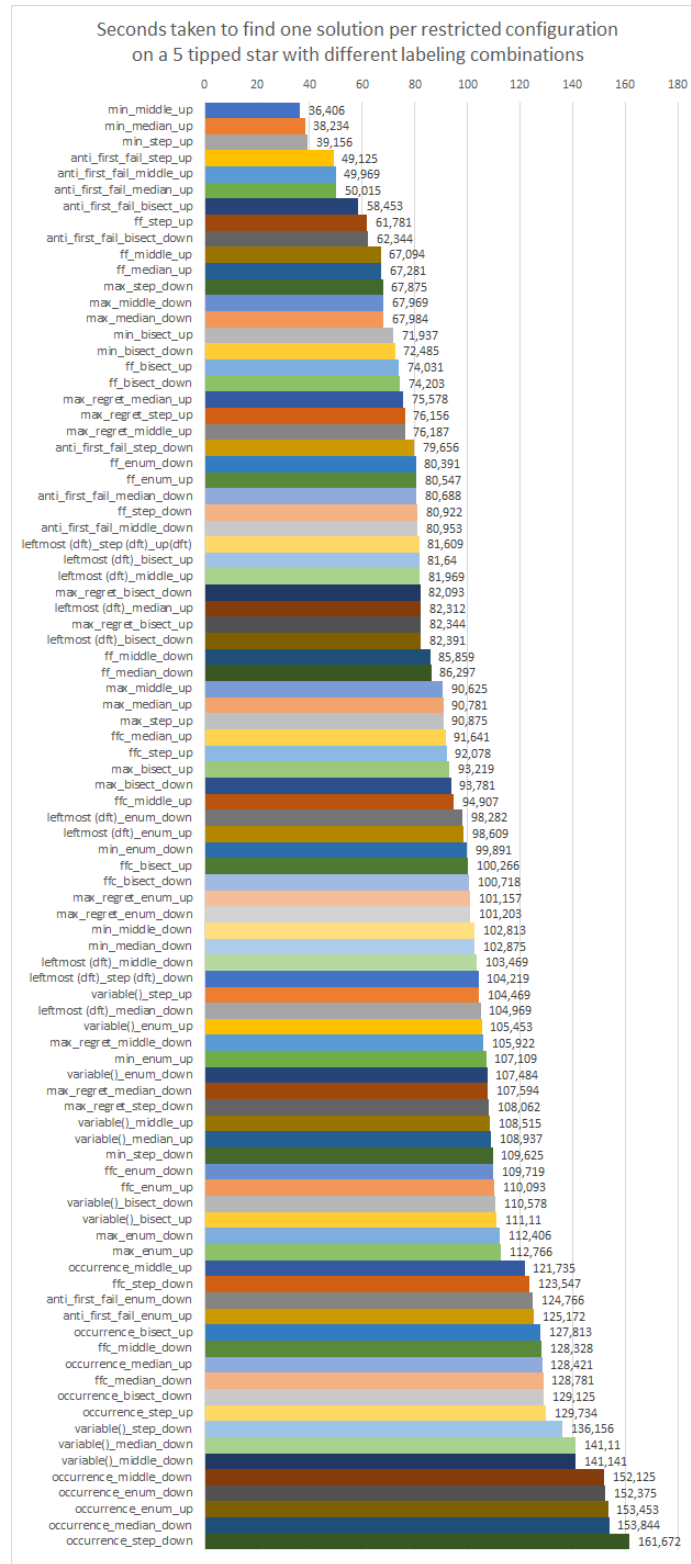


Figura 10. Resultados de teste de diferentes heurísticas no labeling da solução de puzzles

C Anexos

```

apply_restriction(Op0i, C, A, Op7i, I, F),
apply_restriction(Op1i, A, D, Op4i, G, J),
apply_restriction(Op9i, B, C, Op2i, D, E),
apply_restriction(Op8i, B, F, Op5i, H, J),
apply_restriction(Op3i, G, E, Op6i, I, H).

% predicate to apply the equation restriction
apply_restriction(Op1, Var1, Var2, Op2, Var3, Var4):-
    apply_restriction(Op1, Var1, Var2, Value),
    apply_restriction(Op2, Var3, Var4, Value).
apply_restriction(+, Var1, Var2, Value):-
    Var1+Var2 == Value.
apply_restriction(-, Var1, Var2, Value):-
    Var1-Var2 == Value.
apply_restriction(*, Var1, Var2, Value):-
    Var1*Var2 == Value.
% in case of division, the operands must be integers
apply_restriction(/, Var1, Var2, Value):-
    % Var1/Var2 == Value
    Var1 == Value*Var2.

```

Figura 11. Nas primeiras 5 linhas est  uma vers o desactualizada de como se estavam a definir as restri  es dos operandos, usando o predicado `apply_restriction`/6 fica simples aplicar as restri  es a cada equa  o

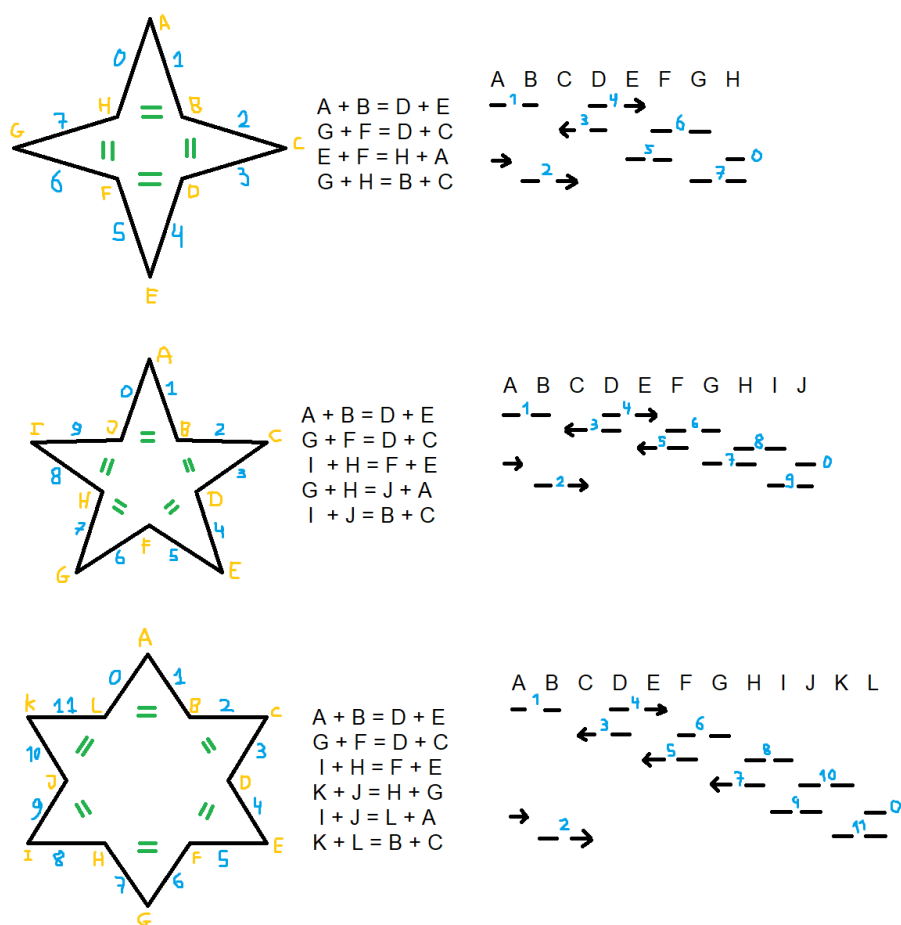


Figura 12. Método encontrado para declarar restrições de forma iterativa. Estrela de equações á esquerda, lista de equações a meio e conjunto de equações direccionais á direita

```

| ?- print_star([/,*,-,+,+/, -, +, +], [1, 9, 0, 2, 7, 4, 8, 5, 6, 3]).

      1
      /  *
6      +      3      =      9      -      0
      +      =      =      -
      5      =      =      2
      -      4      +
      /      +
      8      7
[/,*,-,+,+/, -, +, +][1, 9, 0, 2, 7, 4, 8, 5, 6, 3]

```

Figura 13. Representa o de uma estrela de 5 pontas atrav s do predicado `print_star/2` e representa o da mesma estrela em forma de listas atrav s do predicado `print_result/2`