Resolução de um problema de decisão usando Programação em Lógica com Restrições: Gold Star

Filipe Recharte[up201806743] and Rita Peixoto[up201806257]

FEUP-PLOG, Turma 3MIEIC05, Grupo Gold Star_2

Faculdade de Engenharia da Universidade do Porto

Resumo Descrição da resolução de um problema de decisão que consiste num puzzle, Gold Star, usando programação em lógica com restrições, com o desenvolvimento da solução em SICStus Prolog. Exploram-se as variáveis de decisão e as restrições aplicadas, com descrição da solução desenvolvida e análise das conclusões dos diversos testes. Foram testadas diferentes dimensões e estratégias de pesquisa.

Keywords: Prolog · Restrições · SICStus · Problemas de Decisão · clpfd · Gold Star

1 Introdução

No âmbito da unidade curricular de Programação em Lógica, foi proposta a construção de um programa na linguagem Prolog para a resolução de um problema de decisão, com recurso à programação em lógica com restrições, utilizando a biblioteca clpfd.

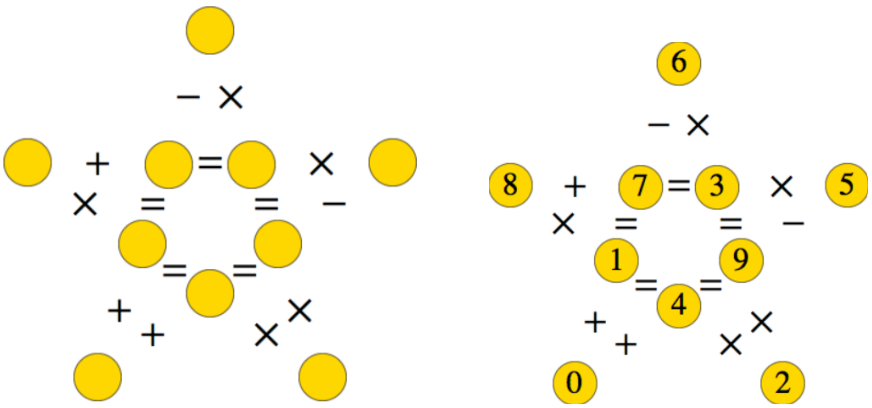
O grupo escolheu o puzzle Gold Star, que consiste num conjunto de equações válidas exibidas em forma de estrela. Este trabalho divide-se em duas grandes partes que são a pesquisa da solução para problema e a geração aleatória de problemas.

O presente artigo começa por descrever o problema, passando para a abordagem na resolução do mesmo, onde é possível encontrar as variáveis de decisão, as restrições e a geração de problemas. Após isto aborda-se a visualização das soluções, é feita a análise dimensional do problema e são analisadas diferentes estratégias de pesquisa. Terminando com as conclusões a partir dos dados obtidos.

2 Descrição do Problema

O problema da Gold Star consiste num puzzle que exibe um conjunto de equações em forma de estrela cujo objetivo é preencher os círculos vazios das equações de modo a que estas se tornam verdadeiras quando lidas da esquerda para a direita.

A resolução do problema é preencher com operadores aritméticos (+, - , *, /) e dígitos de 0 a 9 os círculos vazios da estrela, sendo que cada dígito é utilizado uma única vez. Sendo o caso base uma estrela de 5 pontos, existem 5 equações que partilham as 10 varáveis numéricas e os 10 operadores aritméticos, obtendo uma estrela como a que se apresenta na Figura 1 .



(a) não resolvida (b) resolvida

Figura 1. Exemplo da configuração da estrela de 5 pontas

3 Abordagem

O puzzle Gold_Star corresponde a um problema de satisfação de restrições (PSR). A solução de um problema de satisfação de restrições consiste na atribuição de um valor a cada variável de decisão, dentro do seu domínio, de forma a que todas as restrições sejam satisfeitas.

Para solucionar este problema, a estrela foi convertida num sistema de equações. Recebendo um determinado conjunto de operadores e tendo em atenção que há variáveis partilhadas entre equações, é necessário determinar valores numéricos, que substituídos nos locais vazios tornam todas as equações verdadeiras.

3.1 Variáveis de decisão

A solução do problema é constituída por duas listas, uma composta pelos operadores aritméticos, que são um de entre os seguintes: [+, -, *, /] e outra pelos dígitos cujo domínio é [0,9].

Inicia-se a solução com a obtenção da lista de operadores do problema através do predicado getOperatorsRestricted/2 ou getOperatorsNotRestricted/2, sendo que para tomar fazer uso da programação em lógica com restrições estes são convertidos numa representação numérica passando o seu dominío a ser [1,4], uma

vez que a biblioteca clpfd apenas funciona com domínios finitos. Este predicado tem 10 variáveis de decisão, correspondendo a cada operador, no caso base do problema que são 5 pontas.

O predicado star/4 recebe a lista de operadores referida acima, traduzida para sua representação real através do predicado numberOperatorsToSignals(Ops,Operators)/2, e encontra, se possível, a solução do problema com este conjunto de operadores. Este predicado possui outras 10 variáveis de decisão, no caso base do problema, correspondendo a cada dígito.

3.2 Restrições

Todas as restrições deste problema são rígidas. De modo a responder ao problema proposto, foi utilizado o predicado all_distinct/1 para garantir que cada valor numérico era apenas utilizado uma única vez. Para além disso, foi implementado o predicado restrictions/6 que recebe os 2 operadores e os 4 dígitos que vão constituir uma equação e garante que esta é verdadeira com esses valores.

Sendo o problema em forma de estrela, há várias formas de gerar simetrias, desde eixos de reflexão a rotações, e portanto, tentar encontrar as soluções para este problema sem restrições iria gerar um conjunto de soluções que eram semelhantes tendo apenas uma organização diferente. Podendo estas soluções ser obtidas a partir de uma só destas soluções, por via de rotações e reflexões, é, então, escusado guardar estas soluções.

As restrições implementadas foram:

- O primeiro operador da lista de operadores é o que tem o valor mais alto da lista na sua representação numérica:
  Com recurso ao predicado check/2, que garante que não são admitidas listas de operadores de soluções que apenas são shifts ou reflexões de outra solução, admitindo apenas a solução cujo primeiro operador é o mais elevado em termos de representação numérica.
  Exemplo:
  1)
  Ops: [3,1,1,1,1,1,3,1,1,1]
  Vars:[4,7,2,3,8,6,0,5,9,1]
  2)
  Ops: [1,1,3,1,1,1,1,1,3,1]
  Vars: [4,7,2,3,8,6,0,5,9,1]
  3)
  Ops: [1,1,1,1,1,3,1,1,1,3]
  Vars: [9,5,0,6,8,3,2,7,4,1]
  Neste exemplo, utilizando a restrição check, apenas a primeira opção seria aceite.
- Todas as divisões calculadas são inteiras, não sendo permitidos arredondamentos para o inteiro mais próximo. Numa fase inicial deste trabalho, foram encontradas soluções que não eram exatamente verdadeiras, uma vez que eram efetuados arredondamentos, sendo obtidas "falsas"igualdades.

Ambas estas restrições mostram reduzir bastante o espaço de solução e tempo de procura de soluções em comparação com apenas ter implementado as restrições base do problema, como é possível analisar na Tabela 1.

Tabela 1. Valores dos testes de restrições para uma estrela de 5 pontas

	Tempo de execução(ms)	**Número de soluções obtidas**
sem restrições	234494	1516944
com check	417620	784598
com divisão	1118504	5071
com check e divisão	80254	838

3.3 Geração de problemas

Como referido anteriormente, este trabalho integra, também,a geração de problemas a resolver. Neste caso, a implementação é em tudo semelhante diferindo apenas a forma como são geradas as listas de operadores, que passam a ter uma geração pseudo-aleatória. Tal acontece dando uso ao predicado getOperators-Random/2 que gera os operadores, depois é testado o solver e se tiver solução termina, senão vai gerar uma nova configuração de operadores e tenta arranjar uma solução para esta configuração, dando uso ao predicado repeat.

Tabela 2. Tempo médio de execução da procura de uma solução para uma configuração de operadores gerada de forma pseudo-aleatória

Nº de pontas	**Tempo de execução(ms)**
3 not restricted	0,666666667
3 restricted	0,666666667
4 not restricted	9
4 restricted	34,33333333
5 not restricted	24
5 restricted	28,66666667
6 not restricted	129
6 restricted	197,3333333

Na Tabela 2, analisando o tempo médio das diferentes tentativas para o mesmo caso, é possível concluir que com o aumento da dimensão do problema aumenta o tempo de execução da procura.

4 Visualização da Solução

O predicado print_gold_star/2 é utilizado para a representação de uma solução quando a estrela é constituída por 5 pontas, uma vez que é o tamanho padrão

deste problema (Figura 2). Para dimensões superiores, seria necessário fazer uma função por tamanho, não sendo relevante para este trabalho.

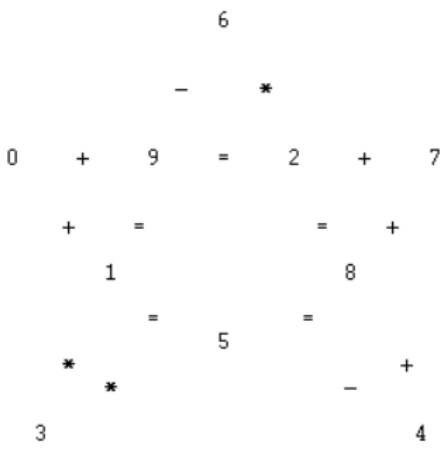


Figura 2. Exemplo do output na consola do predicado print_gold_star para uma solução de uma estrela de 5 pontas

Quando se pretende encontrar solução para uma estrela com tamanho diferente de 5, são impressas duas listas, uma com os operadores, outra com os operandos que correspondem a uma solução válida, recorrendo ao predicado print_solution/2 (Figura 3).

```
] ?- getASolution(6,0).
Operators: [+,+,+,+,+,+,+,+,+,+,+,+]  Digits: [0,5,11,1,4,10,2,8,6,3,7,9]
yes
```

Figura 3. Exemplo do output na consola do predicado print_solution para uma solução de uma estrela de 6 pontas

A organização da lista de operadores e da lista de operandos está de acordo com a Figura 4 .

O programa contém dois predicados gerais, um para obter soluções e estas serem escritas para a consola, run/3, que recebe o predicado a ser chamado, bem como a dimensão do problema e se deve ser aplicada a restrição de check ou não, e outro para escrever para um ficheiro as soluções obtidas, save/4, que recebe o predicado a ser chamado, o nome do ficheiro em que vão ser guardadas as soluções, a dimensão do problema e se deve ou não ser aplicada a restrição check. O predicado a ser chamado pode ser getASolution/2 que obtém uma solução para uma configuração de operaores, getConfigurationSolution/2 que

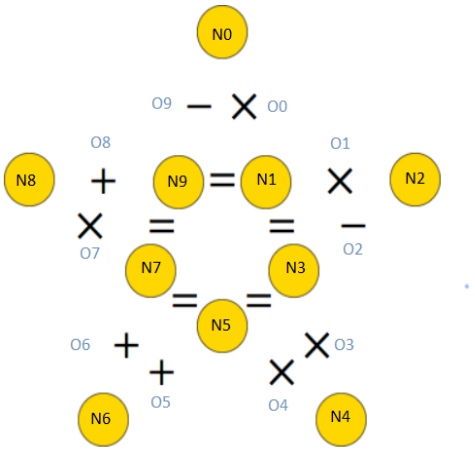


Figura 4. Organização da lista de operadores e da lista de operandos para uma estrela de 5 pontas

obtém uma solução para cada configuração de operadores possível, getAllSolutions que obtém todas as soluções para cada configuração de operadores possível e getRandomSolution que obtém uma solução para a configuração aleatória de operadores com solução.

5 Experiências e resultados

5.1 Análise Dimensional

Após chegar à ideia base deste problema de o traduzir num sistema de equações, facilmente se pode aumentar a dimensão do problema, que, em termos visuais, é o acréscimo de uma ponta à estrela e, em termos do sistema de equações, é o acréscimo de uma equação. A adição desta nova equação exige, por sua vez, a adição de uma nova restrição.

Para responder à análise dimensional para além das restrições fixas encontradas, que são o caso do sistema de equações caso seja uma estrela de 3 pontas, que é a dimensão mais pequena possível, são também impostas restrições que variam com o tamanho da estrela, podendo desde já se concluir que o número de restrições aumenta linearmente com a dimensão.

De modo a implementar estas restrições foram criados os predicados fixed_restrictions/3 e growing_restrictions/3, em que o primeiro aplica as mesmas restrições em qualquer caso e o segundo lida com este aumento de uma equação por ponta adicionada.

Na Figura 5, tendo em atenção que as letras representam os operandos e os números os operadores, é possível perceber quais são as restrições fixas e quais

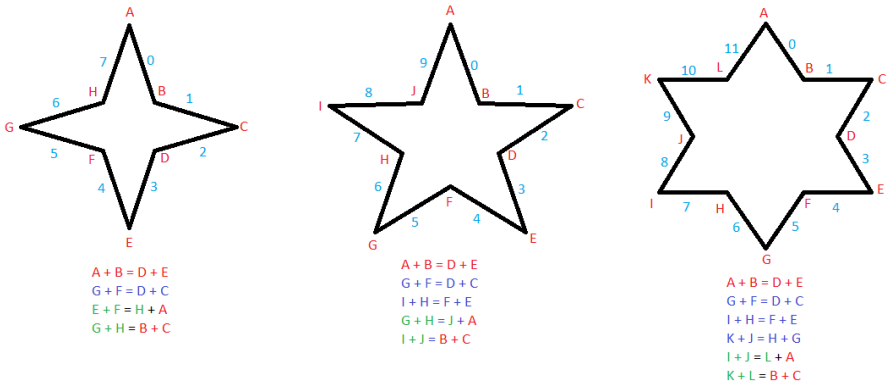


Figura 5. Esquema descritivo da implementação de restrições

as que variam com a dimensão da estrela. As restrições fixas encontram-se a vermelho, a verde estão as restrições que são fixas com o aumento do tamanho, ou seja, são letras que se encontram sempre no mesmo índice relativo na lista de operandos, e a azul as restrições que são acrescentadas com o acréscimo de uma ponta, sendo também visível que se tornam fixas na dimensão acima. Foi com esta análise que foi implementado o predicado growing_restrictions/3.

Tabela 3. Tempo de execução e número de soluções obtidas com diferentes dimensões

N º de pontas	Tempo de Execução(ms)	Número de soluções obtidas
3 unrestricted	563	264
3 restricted	136	132
4 unrestricted	15846	738
4 restricted	3424	186
5 unrestricted	558125	5071
5 restricted	80254	838
6 unrestricted	23416725	66334
6 restricted	2270679	15703

Na Figura 6 e na tabela 3 é possível verificar que o aumento do tempo de execução é exponencial com o aumento do número de pontas, assim como o número de soluções, visível na figura 8 Por esta razão não foram efetuados teste de dimensão superior a 6 pontas.

5.2 Estratégias de pesquisa

A eficiência de um programa em lógica com restrições depende tanto de algoritmos de propagação adequados como de heurísticas apropriadas de seleção da

próxima variável a instanciar no processo de enumeração e do valor a atribuir a essa variável.

Foram realizados alguns testes com diferentes combinações heurísticas de ordenação de variáveis, seleção de valores e ordenação de valores para a procura da mesma solução: estrela de 5 pontas com restrições.

Os testes abrangeram todas as combinações possíveis de argumentos de labeling, tanto na parte dos operadores como dos operandos. Os resultados obtidos encontram-se nas tabelas 5 e 6 e nos gráficos 11 e 12, respetivamente.

Nos resultados obtidos para os operandos, observa-se que a configuração de heurísticas que torna a procura mais eficiente é [min, median, up], aproximadamente 33 segundos mais rápido que a configuração predefinida que é [leftmost,step,up], que melhorou o tempo de execução em 42.5%; conclui-se também que as configurações consideravelmente menos eficientes são as que usam a heurística occurrence e ffc na ordenação de variáveis, sendo a configuração menos efieciente [occurence, middle, down], tendo demorado cerca de 4 minutos, o que piora o tempo de execução em 68,4%. Nestes resultados, é notável a diferença entre o uso de diferentes configurações no labeling.

Para os operadores, não se verifica grandes diferenças entre o uso de diferentes configurações no labeling. É possível concluir que a configuração [leftmost,enum,up] é a que torna a procura mais eficiente, tendo melhorado o tempo de execução em 12,7% e que a configuração [min, bisect,up] é a que torna a procura menos eficiente, tendo piorado o tempo de execução em 65,1%.

6 Conclusões e trabalho futuro

Este trabalho contribuiu para aplicação dos conhecimentos teóricos de programação em lógica com restrições abrangendo diversos tópicos.

Note-se que a solução apresentada poderia ser melhorada e optimizada, de modo a torná-la mais eficiente. Com o aumento da dimensão, o tempo de execução aumenta consideravelmente.

No desenvolvimento deste trabalho foram encontradas dificuldades em remover simetrias e reflexões, tendo sido feitos alguns testes sem quaisquer efeitos, uma vez que o problema tem uma forma complexa. Pensa-se que a restrição aplicada surte bastante efeito, sendo possível haver melhores restrições para diminuir ainda mais o espaço de resposta, mas que o grupo não encontrou, deixando para trabalho futuro.

Conclui-se que o trabalho foi concluído com sucesso, foi possível encontrar tanto soluções para problema base, bem como para problemas de diferentes dimensões e foi possível gerar aleatoriamente problemas a serem resolvidos.

Referências

1. Gold Star Puzzles, https://erich-friedman.github.io/puzzle/star/.Last accessed 4 Jan 2021
2. SWI-Prolog, https://www.swi-prolog.org/. Last accessed 4 Jan 2021

7 Anexos

A Tabelas de dados

Tabela 4. Resultados da procura de uma solução para cada configuração de operadores

N º de pontas	**Tempo de execução(ms)**	**Número de soluções obtidas**
3 not restricted	108	264
3 restricted	97	23
4 not restricted	14761	397
4 restricted	2905	70
5 not restricted	734696	3567
5 restricted	100215	546

Tabela 5: Resultados dos testes de todas as combinações de heurística possíveis para encontrar configurações de operadores

Heurísticas			Tempo(ms)
Ord. Variáveis	Sel.Valores	Ord.Valores	
anti first fail	bisect	down	1724
		up	1588
	enum	down	1541
		up	2299
	median	down	1773
		up	2008
	middle	down	1582
		up	1808
	step	down	2082
		up	1686
ff	bisect	down	1562
		up	1931
	enum	down	1365
		up	1504
	median	down	1589
		up	2227
	middle	down	1633
		up	1660
	step	down	1824
		up	1999
ffc	bisect	down	1586
		up	1495
	enum	down	1547
		up	1411

Tabela 5 – Continuação da página anterior

	median	down	1529
		up	1525
	middle	down	1786
		up	1569
	step	down	1418
		up	1521
leftmost	bisect	down	1573
		up	1483
	enum	down	1508
		up	1327
	median	down	1520
		up	1403
	middle	down	1516
		up	1394
	step	down	1464
		up	1521
max regret	bisect	down	1601
		up	1790
	enum	down	1958
		up	1660
	median	down	1951
		up	2394
	middle	down	1881
		up	2177
	step	down	1808
		up	1824
max	bisect	down	1775
		up	2012
	enum	down	1719
		up	2255
	median	down	1943
		up	2124
	middle	down	2034
		up	2096
	step	down	2541
		up	1894
min	bisect	down	2904
		up	4363
	enum	down	1741
		up	1486
	median	down	1922
		up	2141
	middle	down	1980
		up	1678

Tabela 5 – Continuação da página anterior

	step	down	1629
		up	1457
occurrence	bisect	down	1567
		up	1553
	enum	down	1513
		up	1642
	median	down	1477
		up	1460
	middle	down	1516
		up	1365
	step	down	1664
		up	1892

Tabela 6: Resultados dos testes de todas as combinações de heurística possíveis para encontrar configurações de operandos

Heurísticas			Tempo(ms)
Ord. Variáveis	Sel.Valores	Ord.Valores	
anti first fail	bisect	down	68110
		up	67805
	enum	down	110356
		up	110528
	median	down	100118
		up	58400
	middle	down	100202
		up	58442
	step	down	100317
		up	58395
ff	bisect	down	70404
		up	70409
	enum	down	72123
		up	72317
	median	down	79438
		up	67557
	middle	down	79558
		up	68288
	step	down	79577
		up	67623
ffc	bisect	down	157914
		up	157018
	enum	down	166338
		up	166124
	median	down	185947

Tabela 6 – Continuação da página anterior

		up	142456
	middle	down	185793
		up	142257
	step	down	185268
		up	142376
leftmost	bisect	down	76841
		up	77075
	enum	down	82302
		up	82691
	median	down	88116
		up	77568
	middle	down	88070
		up	77777
	step	down	89105
		up	79003
max regret	bisect	down	76353
		up	75964
	enum	down	83341
		up	83673
	median	down	89541
		up	75222
	middle	down	89612
		up	75121
	step	down	89937
		up	75307
max	bisect	down	109743
		up	109668
	enum	down	108344
		up	108539
	median	down	90090
		up	97558
	middle	down	90022
		up	97293
	step	down	90219
		up	97639
min	bisect	down	71441
		up	71185
	enum	down	94675
		up	94997
	median	down	100104
		up	45466
	middle	down	100333
		up	45503
	step	down	100248

Tabela 6 – Continuação da página anterior

		up	45662
occurrence	bisect	down	198288
		up	196999
	enum	down	233549
		up	233531
	median	down	248790
		up	200385
	middle	down	249684
		up	200394
	step	down	248703
		up	200454

B Gráficos

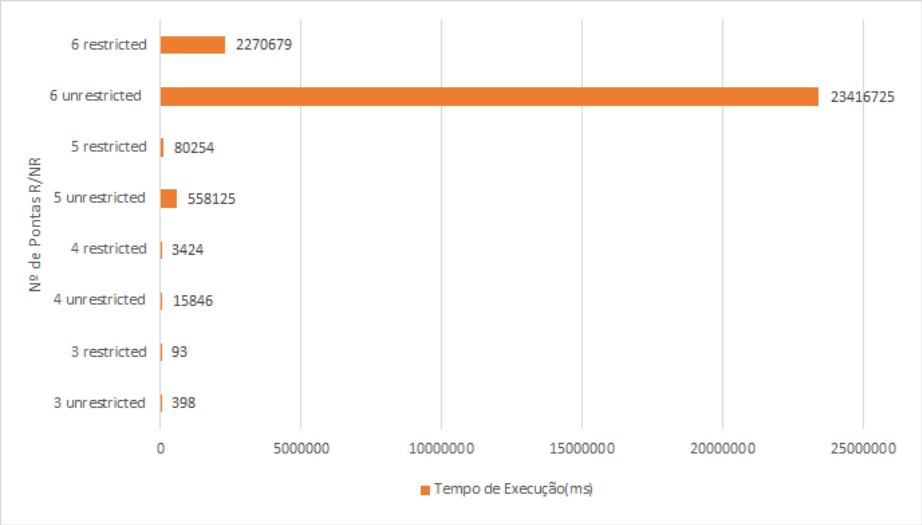


Figura 6. Gráfico correspondente ao tempo de execução da procura de todas as soluções possíveis para diferentes dimensões e restrições

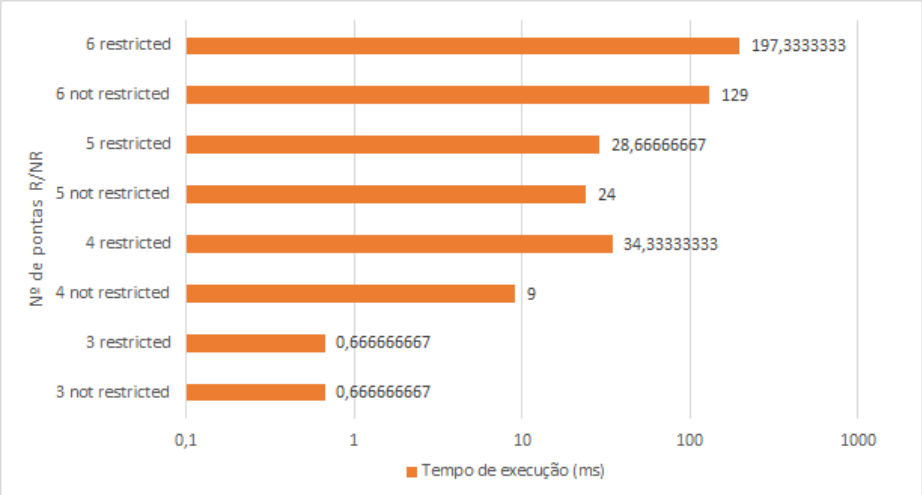


Figura 7. Gráfico correspondente ao tempo médio de execução da procura de uma solução para uma configuração de operadores gerada de forma pseudo-aleatória

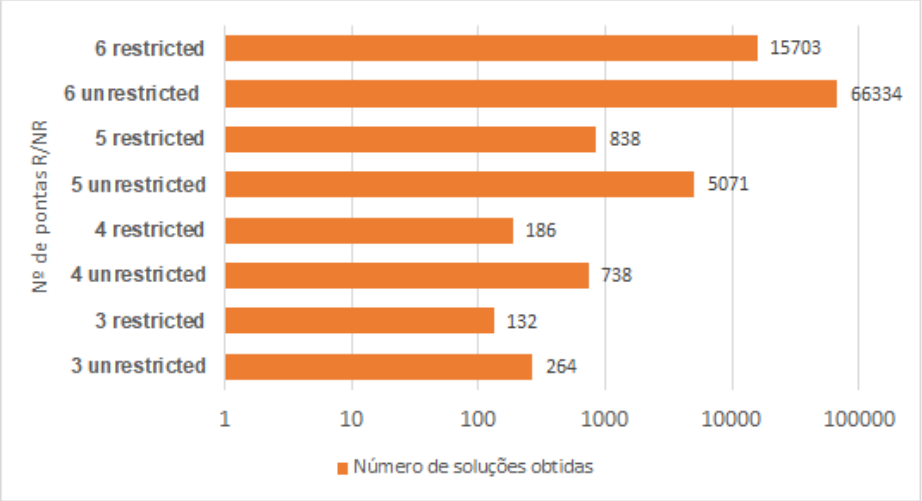


Figura 8. Gráfico correspondente ao número de soluções da procura de todas as soluções possíveis

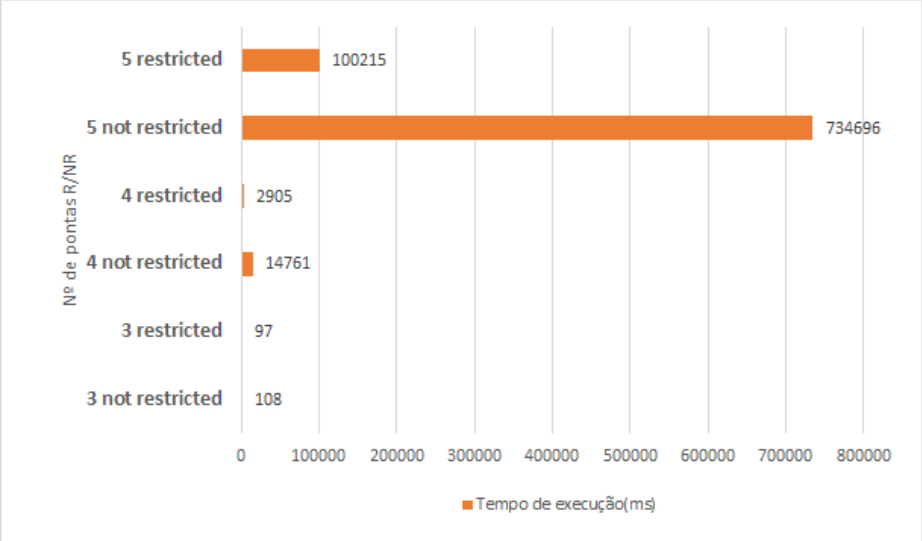


Figura 9. Gráfico correspondente ao tempo de execução da procura de uma solução para cada configuração de operadores

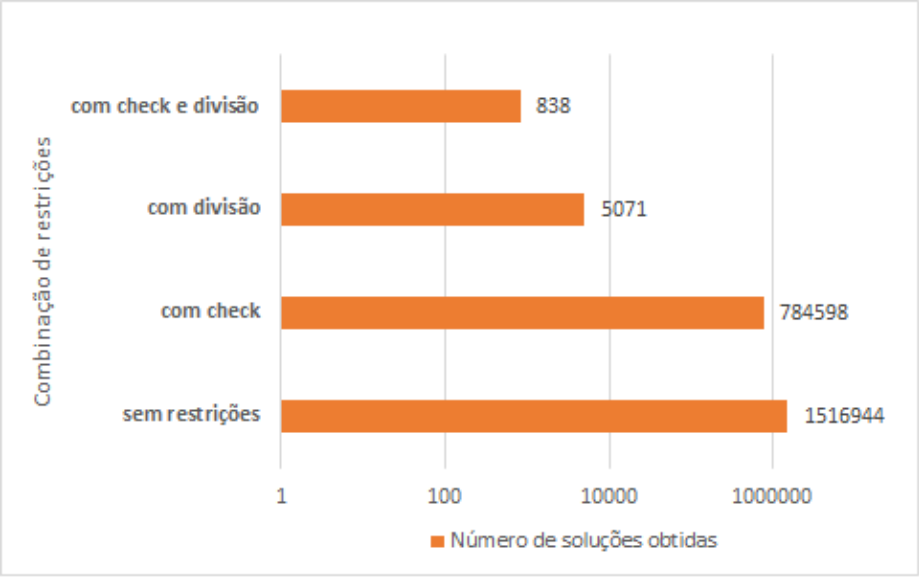


Figura 10. Gráfico correspondente ao número de soluções para cada combinação de restrições numa estrela de 5 pontas

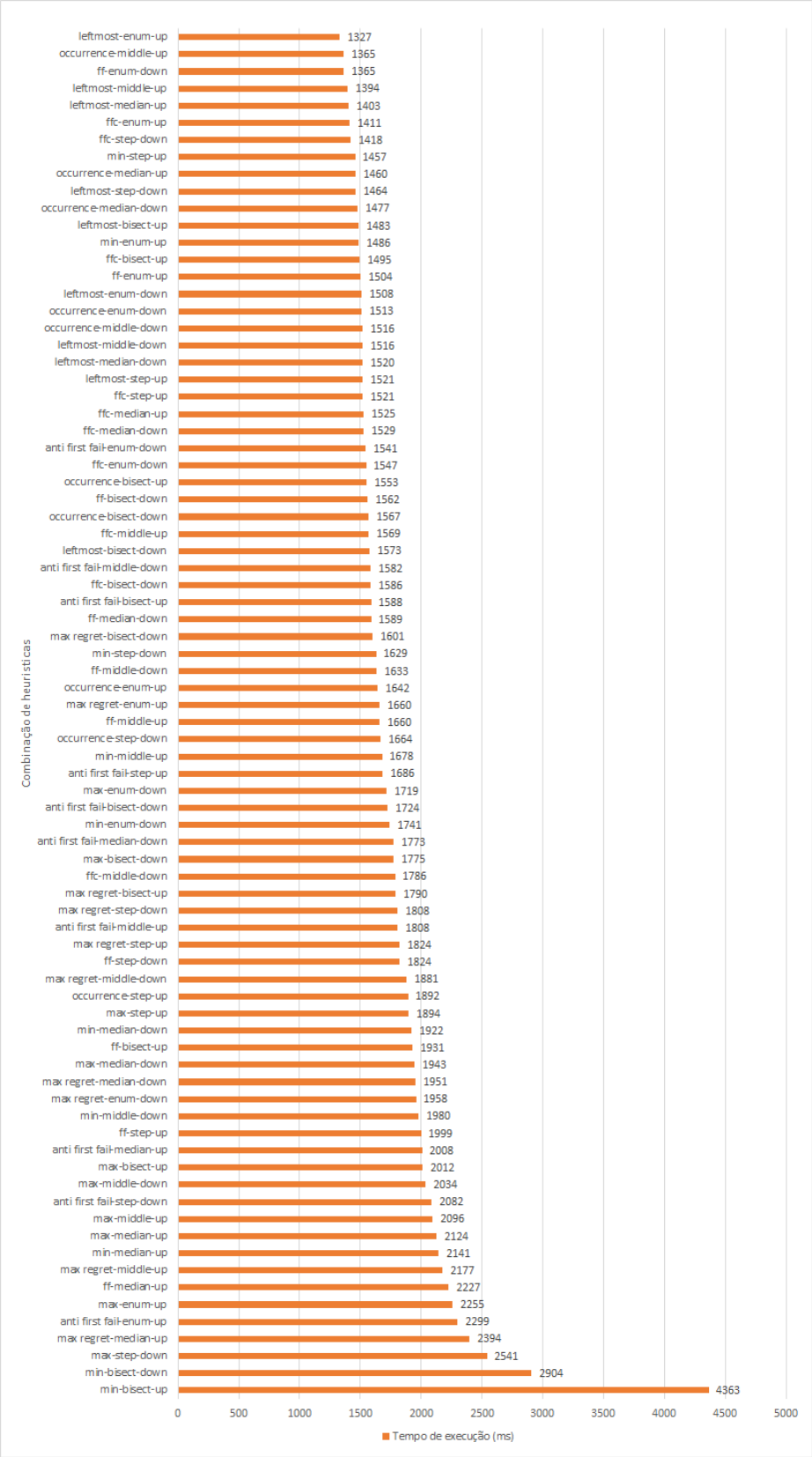


Figura 11. Gráfico correspondente ao tempo de execução da procura restrita de todas as combinações de operadores para cada combinação de heurísticas

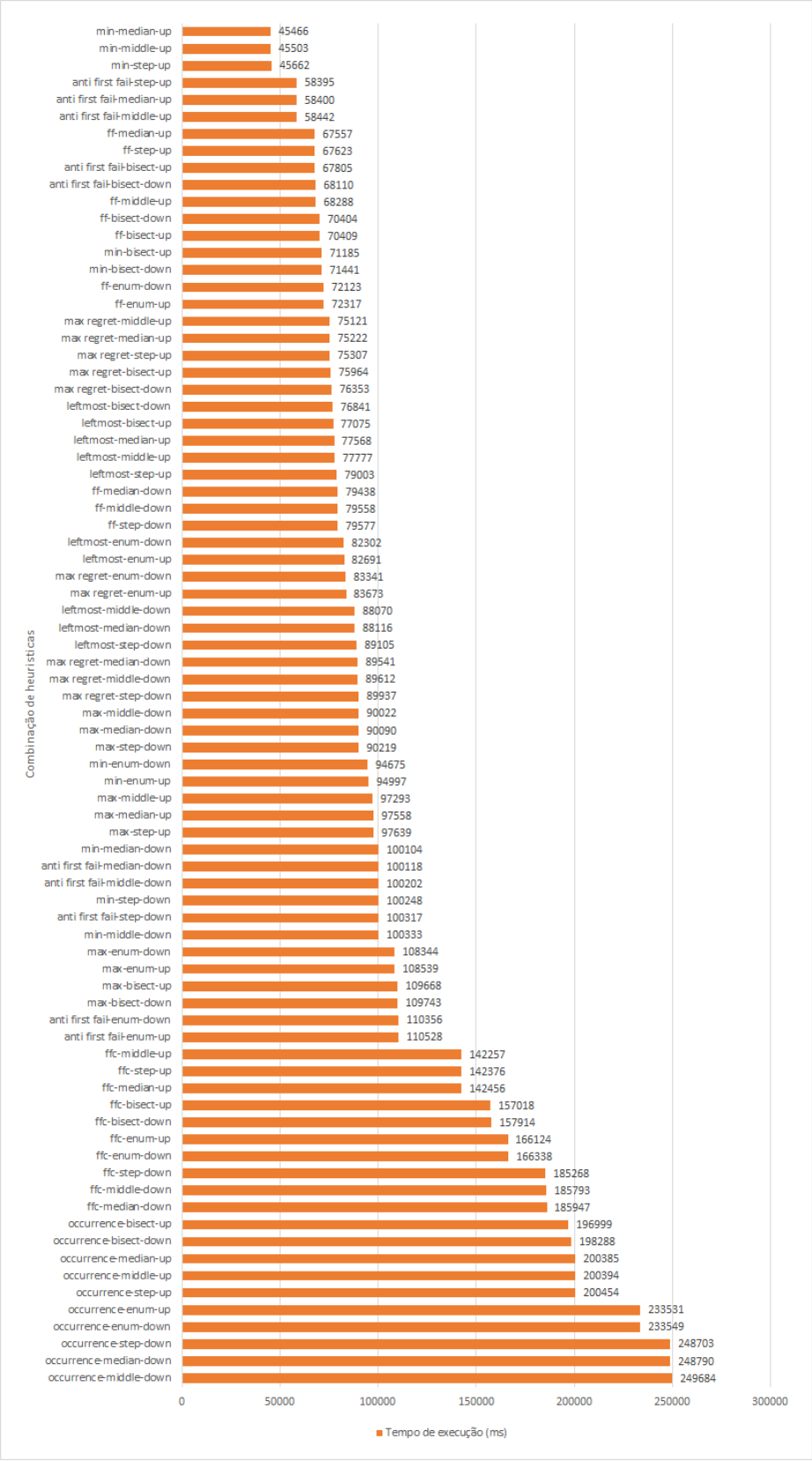


Figura 12. Gráfico correspondente ao tempo de execução da procura restrita de todas as soluções possíveis de uma estrela de 5 pontas para cada combinação de heurísticas