Pesquisa Operacional para o Desenvolvimento
Vol. 10, n. 1, p. 1-8, 2018

ISSN 1984-3534

PROGRAMAÇÃO POR RESTRIÇÕES: UM BREVE TUTORIAL[1]

Luiz Henrique Cherri *
Otimized Decision Making (ODM), São Carlos-SP
Brasil

Recebido 02/05/2018, aceito 21/05/2018

RESUMO

Existem diversas estratégias para modelar matematicamente problemas de otimização combinatória e cada uma dessas estratégias leva ao uso de diferentes métodos de resolução dos modelos desenvolvidos. Este tutorial apresenta a Programação por Restrições, uma técnica de modelagem matemática para a representação de problemas de otimização combinatória. São também apresentados os conceitos básicos do método de solução comumente empregado na resolução destes modelos. Um exemplo de aplicação da técnica na modelagem e resolução de um problema é apresentado. Ao final deste tutorial, o leitor estará apto a construir modelos utilizando a Programação por Restrições.

Palavras-chave: Programação por restrições, Técnica de modelagem, Programação matemática.

ABSTRACT

There are several strategies to mathematically model a combinatorial optimization problem and each of these strategies leads to the use of different solution methods. This tutorial presents the Constraints Programming, a mathematical modeling technique for the representation of combinatorial optimization problems. It also presents the basic concepts of the solution method commonly employed in solving these models. An example of the technique usage to modeling and solving a problem is presented. At the end of this tutorial, the reader will be able to build constraint programming models.

Keywords: Constraint programming, Modeling strategy, Mathematical programming

* Autor para correspondência. E-mail: lhcherri@icmc.usp.br
DOI: 10.4322/PODes.2018.001

[1] *Todos os autores assumem a responsabilidade pelo conteúdo do artigo.*

1. Introdução

Modelagem matemática pode ser descrita como a tentativa de representar matematicamente sistemas e fenômenos do mundo real e é comumente utilizada em diversas áreas do conhecimento, tais como, física, engenharia, economia, biologia, química, entre outras. Dada a natureza do problema, este pode ser modelado utilizando diferentes técnicas. Dependendo da técnica utilizada para modelar o problema, alguns métodos de solução podem ser empregados em sua resolução. Neste tutorial, apresentamos uma técnica de modelagem matemática e resolução de problemas combinatórios bastante versátil e poderosa, conhecida como programação por restrições.

A programação por restrições é formada pela união de dois paradigmas computacionais, a programação lógica e a resolução de restrições. A técnica tem uma origem bastante interdisciplinar, que começou a surgir dentro da área de inteligência artificial nas décadas de 60 e 70 (Kelley, 1961; Geoffrion, 1976). Na área de modelagem de problemas de otimização combinatória, a técnica começou a se popularizar com o trabalho de Jaffar e Lassez (1987), em que os autores definem a programação por restrições como uma classe de linguagem de programação. Os primeiros *softwares* capazes de resolver modelos de programação por restrições foram Prolog III, CLP (R) e CHIP.

Mesmo sendo bastante recente face a outras estratégias de modelagem, a programação por restrições tem se mostrado promissora na resolução de diversos problemas de otimização combinatória. Exemplos destes problemas podem ser encontrados em Clautiaux et al. (2008) para a resolução do problema de corte e empacotamento, em Zhou (1996) na resolução de problemas de *job-shop* e em (Trojet et al., 2011) para a resolução de problemas de *scheduling*. Métodos que integraram a programação por restrições com outras técnicas de modelagem também apresentaram bastante sucesso. Alguns destes métodos integrados melhoram expressivamente os resultados da literatura, como ocorreu em Hooker e Osorio (1999), Jain e Grossmann (2001) e Bollapragada et al. (2001) em que os métodos de solução propostos chegaram a ser 1000 vezes mais rápidos que os apresentados anteriormente na literatura. Outros exemplos de métodos integrados podem ser encontrados em Hooker (2011).

Utilizar programação por restrições na modelagem de problemas de otimização combinatória pode proporcionar diversas vantagens. Uma destas vantagens é a flexibilidade para representar as restrições do problema, pois, estas podem ser construídas utilizando equações lineares e não lineares, operadores lógicos e condições lógicas. Esta flexibilidade possibilita a exploração das características intrínsecas de alguns problemas de formulação mais intuitiva e com uma forma de resolução mais eficaz.

Pela sua natureza, os métodos utilizados na resolução de modelos de programação por restrição são vantajosos para explorar domínios, logo são eficientes para encontrar soluções factíveis para os problemas. Por outro lado, estes métodos não focam na prova de otimalidade da solução, o que pode comprometer a utilização da técnica em alguns casos.

Este tutorial apresenta alguns princípios básicos de Programação por restrições e, destina-se ao público com conhecimentos básicos da área de pesquisa operacional. Esperamos que este texto consiga facilitar e instigar o uso da técnica pelos leitores.

2. Programação por Restrições

Nesta seção, serão apresentadas as principais características de um modelo de programação por restrições e do método de resolução empregado na resolução de modelos.

2.1. Modelagem

Como na maioria dos modelos de otimização combinatória, um modelo de programação por restrições pode ser representado por um conjunto de variáveis x que têm um domínio finito pertencente a $\mathcal{D}$, um conjunto de restrições $C(x)$ e uma função objetivo $E(x)$. Este modelo de

programação por restrições pode ser formulado como em (1)-(3).

$$\text{minimizar:} \quad E(x), \tag{1}$$
$$\text{sujeito a:} \quad C(x), \tag{2}$$
$$x \in \mathcal{D}. \tag{3}$$

No modelo (1)-(3), a estrutura das restrições $C(x)$ é um dos maiores diferenciais na modelagem de um problema via programação por restrições. A razão para isto é que estas restrições podem ser representadas por igualdades, desigualdades e condições lógicas. Por exemplo, as seguintes sentenças são restrições válidas em um modelo de programação por restrições:

(a) $x_1 + 5x_2 - x_3 = 42$,

(b) $x_1^{x_2} + \log(x_3) \leq 17$,

(c) `todosDiferentes` $\{x_1, x_2, x_3\}$,

(d) `Se` $(\log(x_1) \leq 0.5)$ `então` $(x_2 > 10)$.

No exemplo acima (a) é uma equação linear e (b) exemplifica uma equação não-linear. Os itens (c) e (d) representam restrições lógicas. Em (c), os valores das variáveis x_1, x_2 e x_3 devem ser diferentes entre si. A restrição (d) diz que quando a desigualdade $\log(x_1) \leq 0.5$ for satisfeita então a condição $x_2 > 10$ também será. Os softwares de resolução de modelos de programação por restrições possuem algumas das condições lógicas mais utilizadas já implementadas e dão suporte a criação de novas restrições. Na Seção 4, discutimos um pouco mais sobre este tipo de restrições.

Note que esta versatilidade na representação das características de um problema torna a modelagem de diversos problemas mais flexível. Assim, a etapa de modelagem de problemas fica mais fácil e intuitiva.

Dada uma solução x que satisfaça as restrições em $C(x)$ de um problema, a função objetivo $E(x)$ faz a avaliação deste valor. Esta função também pode ter qualquer estrutura desde que consiga mensurar o valor de uma dada solução x. Existem alguns casos em que o objetivo é apenas encontrar uma solução factível para o problema. Nestes casos, a função objetivo pode ser ausente e o método de solução chegará ao fim quando encontrar uma solução x que satisfaça $C(x)$.

A maior limitação quanto a modelagem de um problema utilizando programação por restrições está na definição no domínio $\mathcal{D}$ das variáveis de decisão, que deve ser discreto. Isto pode dificultar a modelagem de problemas de natureza contínua. Entretanto, vale ressaltar que o domínio das variáveis deve ser discreto e não obrigatoriamente inteiro, ou seja, o domínio de uma variável pode ser definido, por exemplo, como {0.5, 1, 1.25, 2.5}, o que pode levar a boas aproximações para vários problemas.

2.2. Método de Solução

A resolução de um problema de programação por restrições é baseada em três elementos: a filtragem, a ramificação e o *backtracking*.

Na fase de filtragem, as restrições do problema são analisadas a fim de reduzir o domínio das variáveis. Após a fase de filtragem, se algumas variáveis ainda não foram fixadas na solução atual, a fase de ramificação é desenvolvida. Na ramificação é escolhido um valor para uma variável e, desta forma, é possível realizar novamente a fase de filtragem para a redução do domínio das demais variáveis. A fase de filtragem e ramificação são iteradas até que todas as variáveis tenham seu domínio fixado. Ao final do processo, uma solução factível para o problema é encontrada

ou, o domínio de uma ou mais variáveis se torna nulo e portanto solução do ramo investigado é infactível.

Note que quando uma ramificação é realizada, é iniciado um processo semelhante ao de busca em árvore. Assim, o *backtracking* é utilizado para regredir para as decisões tomadas durante a fase de ramificação, possibilitando a mudança de tomada de decisão. Isto garante que todas as escolhas possíveis a serem tomadas para o valor de uma variável sejam averiguadas. Caso a função objetivo $E(x)$ não seja nula, a otimalidade da solução é provada quando não há mais soluções a serem averiguadas.

Em sua forma mais básica, este método de solução não utiliza limitantes inferiores para a solução do problema no processo de busca. Portanto, a única estimativa que se tem para este valor em cada ramo é o valor da função objetivo E em na solução x ainda não completa. Esta característica leva a um método que explora rapidamente o domínio de soluções e que, em alguns casos, deixa de provar a otimalidade desta solução por falta de uma informação mais avançada sobre a qualidade das soluções ainda não exploradas do domínio.

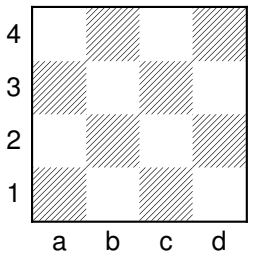
3. Um Pequeno Exemplo

Para que os conceitos da técnica de programação por restrições possam ser melhor compreendidos, introduzimos agora um pequeno exemplo.

4-Rainhas: considere um tabuleiro, como o de xadrez, com 4 linhas e 4 colunas, e rainhas que podem se atacar na vertical, horizontal e diagonal. Queremos saber como é possível posicionar quatro rainhas no tabuleiro sem que um par de rainhas possa se atacar.

Para resolver este problema, considere o tabuleiro representado com linhas enumeradas de '1' a '4' e colunas de 'a' a 'd'. Como cada rainha precisará ser alocada em uma coluna, considere quatro vaiáveis Qa, Qb, Qc e Qd que representam respectivamente a linha em que a rainha das colunas a, b, c e d será alocada. A Figura 1 ilustra o tabuleiro e o domínio inicial das variáveis.

Figura 1: Representação do tabuleiro e possíveis valores para as variáveis.



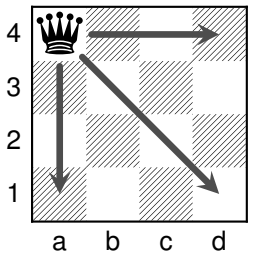
$$Qa = \{1a, 2a, 3a, 4a\}$$
$$Qb = \{1b, 2b, 3b, 4b\}$$
$$Qc = \{1c, 2c, 3c, 4c\}$$
$$Qd = \{1d, 2d, 3d, 4d\}$$

Fonte: Imagem criada pelo autor.

Podemos incluir a primeira rainha em qualquer posição do tabuleiro sem causar infactibilidade, portanto inicialmente a fase de filtragem de domínios não reduz o domínio das variáveis. Desta forma, devemos escolher o valor para uma das variáveis (ramificar). Escolhemos então que uma rainha estará no canto superior esquerdo do tabuleiro, ou seja, $Qa = 4a$. Note que este posicionamento impede que outras rainhas sejam posicionadas na linha 4, na coluna a e na diagonal principal do tabuleiro que contém a posição $4a$. Assim, o domínio das variáveis Qb, Qc e Qd podem ser filtrados. A inserção da peça no tabuleiro e os novos domínios das variáveis são mostrados na Figura 2.

Após a filtragem do domínio das variáveis, podemos notar que ainda é necessário ramificar outra variável do problema para encontrarmos uma solução. Posicionamos então uma rainha na linha 2 da coluna b. Com esta decisão, na fase de filtragem de domínios verificamos que não há

Figura 2: A esquerda é ilustrado o tabuleiro com uma rainha e seus possíveis movimentos. A direita se encontra o novo domínio das variáveis.

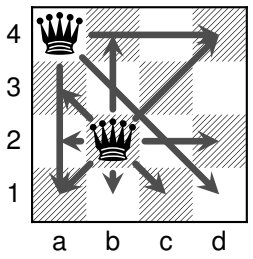


$Qa = \{4a\}$
$Qb = \{1b, 2b\}$
$Qc = \{1c, 2c, 3c\}$
$Qd = \{2d, 3d\}$

Fonte: Imagem criada pelo autor.

valores possíveis para a variável Qc, i.e., com este posicionamento das rainhas nas colunas a e b, não é possível inserir uma rainha na coluna c do tabuleiro como ilustrado na Figura 3.

Figura 3: A esquerda se encontra a nova configuração do tabuleiro com duas rainhas e a direita o domínio das variáveis. Note que não ha possibilidade para o posicionamento de uma nova rainha na coluna c, tornando o domínio de Qc nulo.



$Qa = \{4a\}$
$Qb = \{2b\}$
$Qc = \{\emptyset\}$
$Qd = \{3d\}$

Fonte: Imagem criada pelo autor.

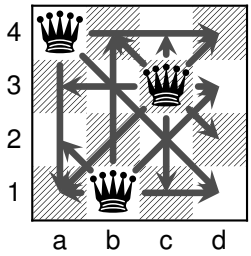
Como a solução que construímos se mostrou infactível, é necessário efetuar o *backtracking*. Assim, voltamos a última decisão tomada e a revertemos (removemos a rainha da linha 2 coluna b), logo a variável Qb tem seu domínio desfixado. Dando continuidade ao método, inserimos agora a rainha da coluna b na linha 1 ($Qb = \{1b\}$) (por que?). Note que com esta decisão a rainha que deve ser posicionada na coluna c somente poderá ser inserida na linha 3 ($Qc = \{3c\}$). Esta escolha de valores de domínios impossibilita a inserção de uma rainha na coluna d ($Qd = \{\emptyset\}$), ou seja, a solução é infactível. A Figura 3 apresenta este caso.

Por conseguinte, devemos efetuar o *backtracking* e continuar nossa busca por uma solução factível. Observe que, para o caso em que há uma rainha na posição $4a$ do tabuleiro, todas as possibilidades de posicionamento das outras rainhas foram verificadas. Assim, mudamos a posição da rainha da coluna a para a linha 3. Com esta alocação, a rainha da coluna b somente poderá ser inserida na linha 1. A nova configuração inicial do tabuleiro é ilustrada na Figura 5.

Agora, a única posição possível para se alocar a rainha da coluna c é na linha 4, o que impossibilita a rainha da coluna d de ser alocada na linha 4, como apresentado na Figura 6(a). Desta forma, só existe a possibilidade de inserir uma rainha na coluna d se esta estiver na linha 2. Portanto, uma solução factível é obtida para o problema como apresentado nas Figuras 6(b) e 6(c).

A solução apresentada na Figura 6(b) é uma das soluções factíveis do problema. Se quisermos encontrar as outras soluções factíveis do problema, podemos efetuar novamente o *backtracking* e continuar a busca por novas soluções. O exercício de encontrar uma nova solução factível para o problema fica a cargo do leitor.

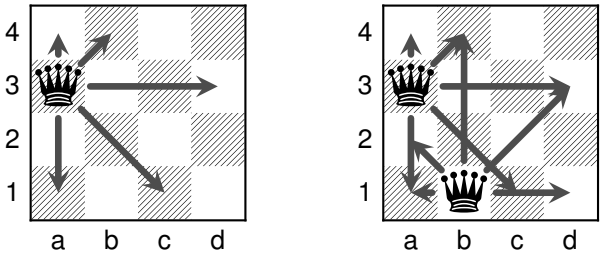
Figura 4: A esquerda se encontra a configuração do tabuleiro e, a direita o domínio das variáveis. Novamente, a solução é infactível pois não há a possibilidade para o posicionamento de uma rainha na coluna d, tornando o domínio de Qd nulo.



$$Qa = \{4a\}$$
$$Qb = \{1b\}$$
$$Qc = \{3c\}$$
$$Qd = \{\emptyset\}$$

Fonte: Imagem criada pelo autor.

Figura 5: Nova configuração do tabuleiro apresentando o posicionamento das rainhas nas colunas a e b à esquerda e o domínio das variáveis à direita.



$$Qa = \{3a\}$$
$$Qb = \{1b\}$$
$$Qc = \{4c\}$$
$$Qd = \{2d, 4d\}$$

Fonte: Imagem criada pelo autor.

Para a resolução deste exemplo, utilizamos a lógica que os resolvedores de modelos de programação por restrições empregariam, entretanto, ainda não definimos um modelo formal para o problema que acabamos de resolver. Neste ponto, instigamos o leitor a, antes de ver a modelagem que começa subsequentemente a este paragrafo, pensar em como seria possível modelar o problema apresentado para um tabuleiro com n linhas e n colunas no qual n rainhas devem ser inseridas sem a possibilidade de se atacarem.

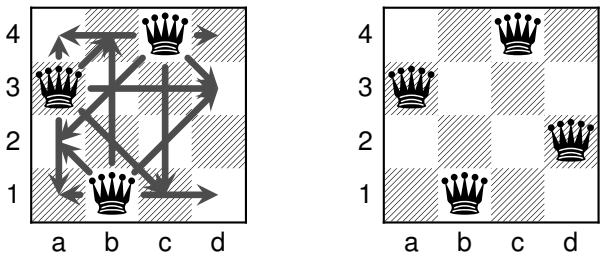
Modelagem do Problema

Considere as variáveis Q_i, $i = 1, ..., n$ com o domínio {1,2,...,n}, que representam as n colunas do tabuleiro e cujo valor indica a linha que a rainha deve ser posicionada na coluna do tabuleiro. Utilizando esta definição de variáveis, uma das restrições do problema é automaticamente resolvida pois, como a variável só pode assumir um único valor na solução, nunca haverão duas rainhas em uma mesma coluna. As outras restrições podem ser formuladas com o uso do operador lógico "`todosDiferentes`". Especificamente, para evitar que duas peças estejam na mesma linha, podemos exigir que o valor de cada variável Q_i, $i = 1, ..., n$ seja distinto, ou seja,

$$\texttt{todosDiferentes}(Q_i, \quad i = 1, ..., n)$$

Ainda resta impedir que haja duas rainhas em uma mesma diagonal. Vamos primeiro pensar no caso da diagonal principal do tabuleiro. Note que, quando duas rainhas estão em uma mesma diagonal principal a diferença do número da linha em que estão alocadas pelo índice da coluna é a mesma para ambas. Por exemplo, se $Q_1 = 2$ e $Q_3 = 4$ temos que as rainhas estão em uma mesma diagonal e que a subtração do valor da linha pelo índice da coluna de ambos é -1 (faça mais

Figura 6: Em (a) é apresentada a configuração do tabuleiro antes da última rainha ser inserida e em (b) é apresentada uma solução factível para o problema. O domínio das variáveis na solução factível é apresentado em (c).



$$Qa = \{3a\}$$
$$Qb = \{1b\}$$
$$Qc = \{4c\}$$
$$Qd = \{2d\}$$

(a) (b) (c)

Fonte: Imagem criada pelo autor.

testes para verificar!). Tomando proveito deste fato, a restrição "`todosDiferentes`" pode ser utilizada para evitar que duas rainhas estejam em uma mesma diagonal principal pela restrição:

$$\texttt{todosDiferentes}(Q_i - i, \quad i = 1, ..., n)$$

Para eliminar não existirem duas rainhas numa mesma diagonal secundaria, o mesmo raciocínio pode ser empregado, levando à seguinte restrição,

$$\texttt{todosDiferentes}(Q_i + i, \quad i = 1, ..., n)$$

4. Restrições Globais

Para alguns problemas, as restrições lógicas previamente definidas em softwares para a resolução de modelos de programação por restrições não permitem que o domínio das variáveis do problema seja reduzido com a maior eficiência possível. Para estes casos, a definição de uma ou mais restrições globais pode trazer um grande impacto na qualidade do modelo de programação por restrições.

As restrições globais geralmente são criadas para utilizar características do problema a ser resolvido pelo modelo, substituindo um conjunto de restrições mais gerais que seriam necessárias para representar esta característica. Assim, a propagação dos domínios das variáveis pode ser feita de forma mais eficiente.

Diversas restrições globais já existem e, as mais comuns podem ser encontradas em softwares comerciais e não comerciais para a resolução de modelos de programação por restrições. Grande parte das restrições globais propostas na literatura estão organizadas em um catálogo de restrições globais (*Global Constraint Catalog*[1]) (Beldiceanu et al., 2007).

5. Conclusões e Perspectivas

Nesse tutorial, foram apresentados conceitos básicos de programação por restrições e um exemplo de aplicação da técnica. Frente a outras estratégias de modelagem de problemas de otimização combinatória, programação por restrições é bastante recente. Como consequência disso, constantemente surgem abordagens que podem ser incorporadas na resolução destes modelos bem como novas restrições globais para a representação de condições lógicas específicas de determinadas aplicações. Assim, investigar abordagens que utilizem a técnica para a resolução de novos

[1] `sofdem.github.io/gccat`

problemas pode ser uma linha promissora de pesquisa.

Agradecimentos. O Autor agradece o apoio da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), processo número 2015/24987-4 e 2013/07375-0.

Referências

Beldiceanu, N., Carlsson, M., Demassey, S. e Petit, T. Global constraint catalogue: Past, present and future. *Constraints*, v. 12, n. 1, p. 21–62, 2007.

Bollapragada, S., Ghattas, O. e Hooker, J. N. Optimal design of truss structures by logic-based branch and cut. *Operations Research*, v. 49, n. 1, p. 42–51, 2001.

Clautiaux, F., Jouglet, A., Carlier, J. e Moukrim, A. A new constraint programming approach for the orthogonal packing problem. *Computers & Operations Research*, v. 35, n. 3, p. 944–959, 2008.

Geoffrion, A. M. The purpose of mathematical programming is insight, not numbers. *Interfaces*, v. 7, n. 1, p. 81–92, 1976.

Hooker, J. e Osorio, M. Mixed logical-linear programming. *Discrete Applied Mathematics*, v. 96-97, p. 395–442, 1999. ISSN 0166-218X.

Hooker, J. N. *Integrated Methods for OptimizationIntegrated Methods for Optimization*: Springer Publishing Company, Incorporated 2nd edition. ISBN 9781461418993, 2011.

Jaffar, J. e Lassez, J.-L. Constraint logic programming. In: *Proceedings of the 14th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, POPL '87, Munich, West Germany. ACM, 1987. p. 111–119.

Jain, V. e Grossmann, I. E. Algorithms for Hybrid MILP/CP Models for a Class of Optimization Problems. *INFORMS Journal on Computing*, v. 13, n. 4, p. 258–276, 2001.

Kelley, J. E. Critical-path planning and scheduling: Mathematical basis. *Operations Research*, v. 9, n. 3, p. 296–320, 1961.

Trojet, M., H'Mida, F. e Lopez, P. Project scheduling under resource constraints: Application of the cumulative global constraint in a decision support framework. *Computers & Industrial Engineering*, v. 61, n. 2, p. 357–363, 2011.

Zhou, J. *A constraint program for solving the job-shop problem*. In: Freuder, E. C. (ed.) *Principles and Practice of Constraint Programming — CP96, CP 1996*. Lecture Notes in Computer Science, vol. 1118. Berlin, Heidelberg: Springer. ISBN 978-3-540-70620-5, 1996.