12º Congresso de Inovação, Ciência e Tecnologia do IFSP - 2021

METODOLOGIA PARA A CRIAÇÃO DE RESTRIÇÕES PARA PROBLEMAS DE OTIMIZAÇÃO INTEIRA: UMA APLICAÇÃO AO SUDOKU

ARTHUR R. PEREIRA[1], GLAUBER R. COLNAGO[2], JOSÉ R. COLOMBO JUNIOR[3]

[1] Graduando em Engenharia de Controle e Automação, Bolsista PIBIFSP, IFSP, Câmpus Cubatão ramos.pereira@aluno.ifsp.edu.br.
[2] Docente do IFSP Câmpus Cubatão, glauber.colnago@ifsp.edu.br.
[3] Docente do Instituto Técnico da Aeronáutiva, colombojrcj@ita.br.

Área de conhecimento (Tabela CNPq): 3.08.02.02-4 Programação Linear, Não-Linear, Mista e Dinâmica

RESUMO: O presente trabalho propôs um modelo de programação inteira capaz de criar restrições para a resolução de outros problemas de otimização inteira, buscando diminuir a quantidade de restrições e variáveis do problema em que as restrições são aplicadas. A metodologia foi testada no Sudoku, obtendo-se variações da formulação do problema com relação à formulação tradicional. Investigou-se como a quantidade de restrições ou variáveis afeta o desempenho computacional na resolução do problema. Dentre as formulações propostas, a que apresentou um número significativamente menor de variáveis foi a que apresentou menor tempo na resolução, sugerindo que o número de variáveis é mais importante que a quantidade de restrições no caso estudado. A pesquisa, ainda em desenvolvimento, continuará investigando a relação entre o tamanho do problema e desempenho computacional.

PALAVRAS-CHAVE: Pesquisa Operacional; Otimização; Programação Inteira; Sudoku.

METHODOLOGY FOR CREATING CONSTRAINTS FOR INTEGER OPTIMIZATION PROBLEMS: AN APPLICATION TO SUDOKU

ABSTRACT: This work proposed an integer programming model capable of creating constraints for solving other integer optimization problems, aiming to reduce the number of constraints and variables in which the constraints are applied. The method was tested in Sudoku, obtaining alternative formulations to the traditional one. It was investigated how the number of constraints or variables affects the computational performance of the problem. The formulation with a significantly smaller number of variables than the traditional one presented the shortest time, suggesting that the number of variables is more important than the number of constraints in the case studied. The research, in development, will continue to investigate the relationship between problem size and computational performance.

KEYWORDS: Operations Research; Optimization; Integer Programming; Sudoku.

INTRODUÇÃO

O Sudoku é um jogo que consiste, tradicionalmente de uma tabela de tamanho 9×9 células, cujo objetivo é completar todas as células com os números inteiros de 1 a 9, sem que haja repetição de números nas linhas, colunas e nas submatrizes de tamanho 3×3. A Figura 1 apresenta uma combinação de Sudoku válida e a indicação de cada uma das submatrizes.

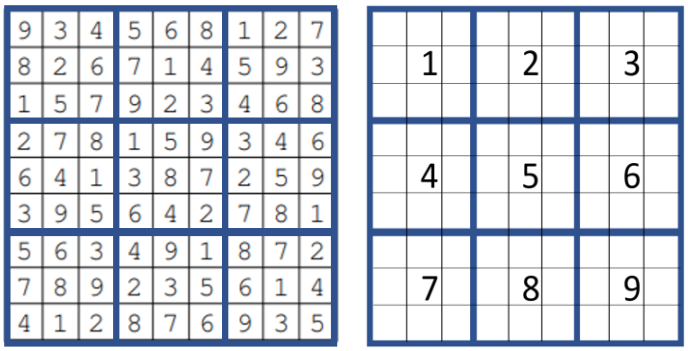
Simonis (2005) generaliza o Sudoku de ordem *n*, como uma matriz de tamanho $n^2 \times n^2$, onde deve-se colocar números de 1 a n^2. Para o Sudoku de ordem $n = 3$, existe um número grande de combinações válidas, a saber, na ordem de $6x10^{21}$combinações. O jogo pode ser formulado como um

problema de otimização de factibilidade (BARTLETT et al., 2008), ou seja, onde há o interesse em encontrar uma solução factível e todas possuem a mesma importância.

Existem diversas técnicas para resolver o Sudoku: Simonis (2005) lida como um problema de restrição, Deng e Yong (2013) resolvem com um algoritmo genético híbrido e Bartlett et al. (2008) formulam o problema como um programa linear inteiro binário, como o apresentado nas equações 1-5.

Este trabalho propõe uma formulação de um programa linear inteiro binário alternativa à formulação tradicional do Sudoku, através de uma metodologia para a criação de restrições de igualdade. Esta metodologia pode ser usada para outros problemas de otimização e aqui é testada no Sudoku. O processo busca criar um número reduzido de restrições, buscando investigar como esse número menor de restrições afeta a performance computacional.

Figura 1 – Exemplo de um tabuleiro Sudoku completo (à esquerda) e as nove submatrizes (à direita).



Fonte: Adaptado de Bartlett et al. (2008)

MATERIAL E MÉTODOS

Tradicionalmente, a formulação matemática do Sudoku é dada pelas equações (1)-(5) (BARLETT et al., 2008):

$$Max\ 0 \tag{1}$$

s.a

$$\sum_{j=1}^{9} x_{ijk} = 1 \;\; i,k \in \{1,\dots,9\} \tag{2}$$

$$\sum_{i=1}^{9} x_{ijk} = 1 \;\; j,k \in \{1,\dots,9\} \tag{3}$$

$$\sum_{k=1}^{9} x_{ijk} = 1 \;\; i,j \in \{1,\dots,9\} \tag{4}$$

$$\sum_{i=i_0}^{i_0+2} \sum_{j=j_0}^{j_0+2} x_{ijk} = 1 \; k \in \{1,\dots,9\} \; e \; i_0, j_0 \in \{1,4,7\} \tag{5}$$

A variável x_{ijk} é binária e indica se o número k está na linha i e coluna j ($x_{ijk} = 1$), ou não ($x_{ijk} = 0$). A Equação (1) é a função objetivo, que não é relevante no caso do Sudoku, pois se trata de um problema de factibilidade. O número *k* deve aparecer uma vez em cada linha (Equação (2)) e uma vez em cada coluna (Equação (3)). Já a restrição (4) garante que cada célula será preenchida com um número apenas. A restrição (5) garante que um número *k* aparecerá uma única vez em cada submatriz 3x3.

A metodologia para criação de variáveis foi aplicada para substituir as equações (2) e (3). O processo para criar restrições de otimização inteira parte da enumeração de combinações de variáveis de um modelo, indicando quais combinações são factíveis e quais são infactíveis. Na aplicação, foram listadas combinações das variáveis de ocorrência de cada número em três filas (linhas ou colunas) da tabela do Sudoku, buscando substituir as 3 restrições de linha ou coluna por um número menor de restrições (1 ou 2).

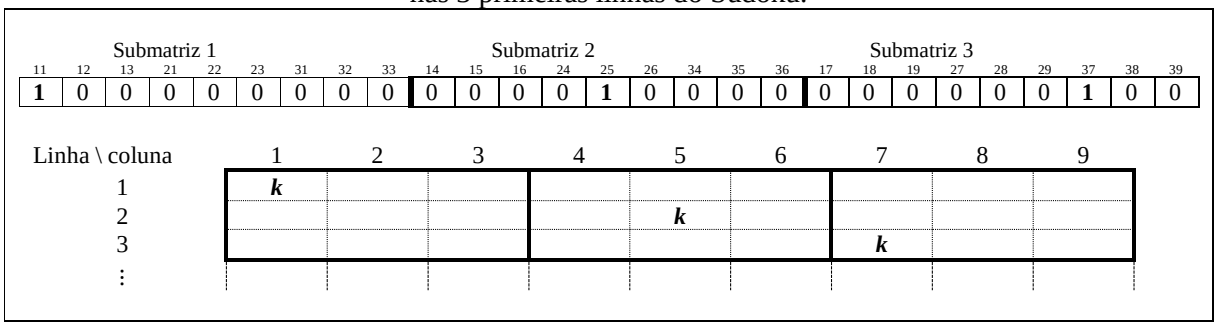
Para isto, foi necessário transformar três linhas (referentes a três submatrizes) em um vetor de 27 posições, sendo que cada posição está associada a uma célula das três submatrizes. Uma posição do vetor está associada à ocorrência ou não de um determinado número na posição referente.

Exemplificando para as 3 primeiras linhas, que estão associadas às submatrizes 1, 2 e 3, considere o vetor da Figura 2. As primeiras 9 posições do vetor referem-se à submatriz 1, com as posições de (1,1) até (3,3), as 9 posições seguintes referem-se à submatriz 2 ((1,4) até (3,6)) e as últimas 9 posições referem-se à submatriz 3 ((1,7) até (3,9)). O número 1 no vetor (referente à ocorrência de um número k) aparece nas posições (1,1), (2,5) e (3,7). O tabuleiro da Figura 2 representa a ocorrência do número k nas posições citadas. Verifica-se que essa combinação é factível para as restrições de linha, pois não há repetição do número k nas 3 primeiras linhas.

Já a Figura 3 apresenta a ocorrência de uma combinação inválida, pois o número k aparece duas vezes na segunda linha, nas posições (2,1) e (2,6).

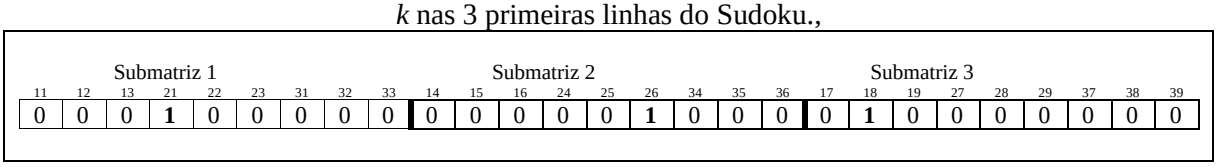
Ao todo, são 729 combinações de ocorrência de um número k uma única vez em cada submatriz. Foram desconsideradas todas as demais combinações, onde um número aparece mais de uma vez em cada submatriz, pois isto é infactível para a Equação (5), que foi mantida na formulação alternativa, e além disto, representaria um número muito grande de possibilidades (2^{27}), o que é proibitivo para a listagem de todas as combinações.

Figura 2. Vetor referente às submatrizes 1 a 3 e ocorrência de uma combinação válida de um número k nas 3 primeiras linhas do Sudoku.



Fonte: Autoria própria

Figura 3. Vetor referente às submatrizes 1 a 3 e ocorrência de uma combinação inválida de um número k nas 3 primeiras linhas do Sudoku.,



Fonte: Autoria própria

Este raciocínio vale para as demais linhas, com as submatrizes 4-5-6 e com 7-8-9, e também vale para as colunas, vinculando as submatrizes 1-4-7, 2-5-8 e 3-6-9.

Com esse arranjo, as combinações foram separadas em dois conjuntos: V para combinações factíveis, e F para as infactíveis, servindo de entrada para o algoritmo que cria as restrições. Ao todo são 162 combinações factíveis e 567 infactíveis.

O modelo para criar restrições é um problema de programação linear inteira mista, cuja formulação é a (6)-(12) e possui as seguintes variáveis e parâmetros:

- w_j: Variável binária - indica se j é restrição válida do problema ($w_j = 1$) ou não ($w_j = 0$);
- u_{jq}: Variável binária - vale 1 se a combinação $q \in F$ não satisfaz a restrição j ou 0 se satisfaz;
- y_{jq}: Variável binária – caso $u_{jq} = 1$ então y_{jq} vale 1 para indicar que a combinação q violou para cima a restrição de igualdade j ou vale 0 caso a combinação q viole a igualdade j para baixo. O valor da variável é indiferente quando $u_{jq} = 0$;
- c_{ij}: Variável real ou inteira - coeficientes da restrição j limitados a L_{inf} e L_{sup};
- n: número de variáveis inteiras que são para a criação de restrições;
- $i \in \{0,1,2, \dots, n\}$: índice das n variáveis da aplicação das restrições mais a posição 0 relacionada ao termo independente de cada restrição;
- r: número máximo de restrições (valor previamente estabelecido);
- $j \in \{1,2, \dots, r\}$: índice de restrição;
- m: quantidade de combinações listadas das n variáveis;

- $q \in \{1,2,\ldots,m\}$: índice das combinações;
- L_{inf}: limite inferior para os coeficientes c_{ij};
- L_{sup}: limite superior para os coeficientes c_{ij};
- M: um número grande.

$$Min \sum_{j=1}^{r} w_j \tag{6}$$

s.a.

$$\forall q \in F \quad \sum_{j=1}^{r} u_{jq} \geq 1 \tag{7}$$

$$\forall q \in F, j \in \{1,2,\ldots,r\} \quad u_{jq} \leq w_j \tag{8}$$

$$\forall i \in \{0,1,\ldots n\}, j \in \{1,2,\ldots,r\} \quad L_{inf} w_j \leq c_{ij} \leq L_{sup} w_j \tag{9}$$

$$\forall q \in F, j \in \{1,2,\ldots,r\} \quad \left[c_{0j} + \left(\sum_{i=1}^{n} c_{ij}\, x_{ij}\right)\right] + u_{jq} - (M+1) y_{jq} \leq 0 \tag{10}$$

$$\forall q \in F, j \in \{1,2,\ldots,r\} \quad \left[c_{0j} + \left(\sum_{i=1}^{n} c_{ij}\, x_{ij}\right)\right] - u_{jq} - (M+1) y_{jq} + (M+1) \geq 0 \tag{11}$$

$$\forall q \in V, j \in \{1,2,\ldots,r\} \quad c_{0j} + \left(\sum_{i=1}^{n} c_{ij}\, x_{ij}\right) = 0 \tag{12}$$

A função objetivo em (6) minimiza a quantidade de restrições do problema. A restrição em (7) garante que uma combinação infactível não satisfaça pelo menos uma das restrições válidas. Já (8) faz com que uma restrição não válida (w_j = 0) force as variáveis u_{jq} a serem 0 também com objetivo de inutilizá-las. Por outro lado, caso uma combinação infactível não satisfaça a restrição *j* (u_{jq}=1) então a restrição deve ser válida (w_j = 1). Em (9) ocorre a limitação dos coeficientes para as restrições válidas do problema. As restrições (10) e (11) garantem que uma combinação infactível não satisfaça a igualdade representada por (12). Caso a restrição seja válida, então $w_j = 1$ e u_{jq} pode ser 0 ou 1. No caso de u_{jq} = 0 então a restrição *q* não necessariamente violou a restrição *j* e y_{jq} pode assumir qualquer valor, mas no caso de u_{jq}= 1, então a igualdade será violada para cima se $y_{jq} = 1$ ou para baixo se $y_{jq} = 0$. As restrições (10) e (11) garantem isto. Por último, a restrição (12) é aplicada apenas para as combinações factíveis, garantindo que a igualdade seja satisfeita.

Com esta formulação foram obtidos coeficientes para restrições que substituíram as restrições tradicionais (2) e (3) do Sudoku pelas novas restrições (13) e (14):

$$i_0 \in \{1,4,7\}, k \in \{1,\ldots,9\} \tag{13}$$

$$c_0 + \sum_{j_0 \in \{i_0, i_0+3, i_0+6\}} \sum_{i=i_0}^{i_0+2} \sum_{j=j_0}^{j_0+2} c_{[(i-i_0)\times 3+(j-j_0)+1+(j_0-i_0)\times 3]} x_{ijk} = 0$$

$$j_0 \in \{1,4,7\}, k \in \{1,\ldots,9\} \tag{14}$$

$$c_0 + \sum_{i_0 \in \{j_0, j_0+3, j_0+6\}} \sum_{j=j_0}^{j_0+2} \sum_{i=i_0}^{i_0+2} c_{[(j-j_0)\times 3+(i-i_0)+1+(i_0-j_0)\times 3]} x_{ijk} = 0$$

As restrições (13) e (14) varrem todas as linhas e colunas, respectivamente, garantindo que em cada linha/coluna deverá haver os inteiros de 1 a 9 sem repetição.

Os coeficientes dependem dos limitantes L_{inf} e L_{sup} definidos. Por exemplo, para o caso de valores de $L_{inf} = -4$ e a $L_{sup} = 4$, foram obtidos coeficientes dando origem à restrição (15) para as três primeiras linhas ou submatrizes ($i_0 = 1$ e $j_0 = 1$):

$$\begin{array}{llll} 3 & -4x_{11} - 4x_{12} - 4x_{13} & -1x_{21} - 1x_{22} - 1x_{23} & -2x_{31} - 2x_{32} - 2x_{33} \\ & +0x_{14} + 0x_{15} + 0x_{16} & +3x_{24} + 3x_{25} + 3x_{26} & +2x_{34} + 2x_{35} + 2x_{36} \\ & -4x_{11} - 4x_{12} - 4x_{13} & -1x_{21} - 1x_{22} - 1x_{23} & -2x_{31} - 2x_{32} - 2x_{33} = 0 \end{array} \tag{15}$$

RESULTADOS E DISCUSSÃO

Foram obtidos vários conjuntos de coeficientes para as restrições (13) e (14) através da definição dos parâmetros L_{inf} e L_{sup}, variando desde $\pm 0{,}5$ até ± 10, conforme a Tabela 1.

O Sudoku foi então implementado no CPLEX 12.10.0.0 em sua formulação tradicional (1) - (5) e na formulação proposta (1), (4), (5), (13), (14) com os diferentes conjuntos de coeficientes obtidos.

Originalmente, a formulação tradicional possui 324 restrições, sendo que cada uma das Equações de (2)-(5) possui 81 restrições. A formulação alternativa compartilha das restrições (4) e (5), mas substitui as 81 restrições da (2) por 27 ou 54 e substitui as 81 da (3) por 27 ou 54, implicando em 216 ou 270 restrições, um número menor que a tradicional, conforme esperado.

As simulações foram feitas em um computador com processador Ryzen 51600 e 16gb de memória RAM. O log do CPLEX indicou os resultados das três últimas colunas da Tabela 1: a quantidade de variáveis, de não zeros e o tempo de solução. A quantidade de variáveis é inicialmente é 729 mas este número pode ser alterado, uma vez que o CPLEX faz uma série de simplificações, eliminando, possivelmente variáveis e restrições antes de realizar a otimização. Nota-se que houve simplificação em apenas algumas combinações, podendo-se destacar a referente aos limitantes ± 1, que reduziu sua quantidade para 504 variáveis. Comparando os tempos de execução, verifica-se que a formulação tradicional apenas foi maior que a formulação referente aos limitantes ± 1. Possivelmente isto aconteceu por conta do tamanho do problema, que foi reduzido com a eliminação de variáveis. Verifica-se que a quantidade de restrições por si só não implicou necessariamente em uma diminuição do tempo de execução. Isto será investigado na continuação da pesquisa.

Tabela 4. Comparação de tempo entre os coeficientes.

Limitantes dos coeficientes de igualdade			Resultados das formulações Sudoku	
L_{inf}	L_{sup}	Formulação Sudoku	Quantidade de Variáveis	Tempo (s)
-10	10	(1), (4), (5), (13), (14)	729	18.698
-9	9		729	19.496
-8	8		729	3.067
-7	7		729	203.250
-6	6		729	3.210
-5	5		729	2.810
-4	4		720	0.993
-3	3		648	0.134
-2	2		720	0.041
-1,5	1,5		729	0.290
-1	1		504	0.014
-0,5	0,5		729	0.042
Formulação tradicional		(1), (2), (3), (4), (5)	729	0.024

CONCLUSÕES

A metodologia para a criação de restrições é uma contribuição deste trabalho que está em desenvolvimento ainda em uma Pesquisa de Iniciação Científica. A proposta visa investigar como o tamanho (quantidade de restrições e variáveis) afeta o tempo de execução de problemas de otimização inteira. Foram obtidas várias formulações com menos restrições que a formulação tradicional do Sudoku, mas pôde-se verificar que esta métrica não foi significativa na obtenção de menores tempos. Já a quantidade de variáveis dá indício de ser uma melhor métrica para o caso aplicado. A relação das métricas citadas, assim como outras possíveis, continuarão sendo investigadas na pesquisa.

AGRADECIMENTOS

Agradecemos ao IFSP pelo fomento da pesquisa através do PIBIFSP.

REFERÊNCIAS

BARTLETT, Andrew et al. An integer programming model for the Sudoku problem. **Journal of Online Mathematics and its Applications**, v. 8, n. 1, 2008.

DENG, Xiu Qin; DA LI, Yong. A novel hybrid genetic algorithm for solving Sudoku puzzles. **Optimization Letters**, v. 7, n. 2, p. 241-257, 2013.

SIMONIS, Helmut. Sudoku as a constraint problem. In: **CP Workshop on modeling and reformulating Constraint Satisfaction Problems**. Citeseer, p. 13-27, 2005.