LIII Simpósio Brasileiro de Pesquisa Operacional
João Pessoa - PB, 3 a 5 de novembro de 2021

Modelos de Programação Inteira Mista para a resolução de jogos do tipo quebra-cabeça

Suzianne Miranda Tormann
Programa de Pós-graduação em Engenharia de Produção - Universidade Federal de Santa Maria
Av. Roraima nª1000 Cidade Universitária, Santa Maria - RS - Brasil
suzianne.tormann@acad.ufsm.br

Viviane Cátia Köhler
Colégio Técnico Industrial de Santa Maria - Universidade Federal de Santa Maria
Av. Roraima nª1000 Cidade Universitária, Santa Maria - RS - Brasil
viviane@redes.ufsm.br

Olinto César Bassi de Araújo
Programa de Pós-graduação em Engenharia de Produção - Universidade Federal de Santa Maria
Av. Roraima nª1000 Cidade Universitária, Santa Maria - RS - Brasil
olinto@ctism.ufsm.br

RESUMO

O uso de jogos do tipo quebra-cabeça lógicos é uma forma interessante de motivar estudantes de cursos de Programação Matemática no processo de aprendizagem de técnicas de Modelagem Matemática. Uma grande variedade destes jogos, além de instigantes, apresentam regras bem definidas e delimitam adequadamente o escopo do problema a ser resolvido. Nesse trabalho são propostos dois modelos de programação inteira mista para a resolução dos jogos Hitori e Heyawake, muito populares entre apreciadores de jogos lógicos. Estes jogos compartilham regras que não são trivialmente traduzidas em relações matemáticas. Testes computacionais são realizados para validar os modelos propostos considerando instâncias classificadas como difíceis por sites especializados. Os resultados mostram que os modelos, quando resolvidos com o CPLEX em um computador pessoal, fornecem soluções factíveis em décimos de segundos.

PALAVRAS CHAVE. Programação inteira mista, Hitori, Heyawake, Quebra-cabeças.

PM – Programação Matemática, OA – Outras aplicações em PO

ABSTRACT

The use of logic puzzles is an exciting way to motivate students of Mathematical Programming courses in learning Mathematical Modeling techniques. A wide variety of these games are provocative and have well-defined rules that adequately delimit the scope of the problem to be solved. In this work, we propose two mixed integer programming models to solve the puzzles Hitori and Heyawake, which are very popular among logic game enthusiasts. These games share rules that are not trivially translated into mathematical relations. Computational tests were performed to validate the proposed models considering instances classified as hard by specialized websites. The results show that the models, when solved with CPLEX on a personal computer, provide feasible solutions in tenths of seconds.

KEYWORDS. Mixed integer programming, Hitori, Heyawake Pencil-and-paper puzzle.

PM - Mathematical Programming, OA - Other applications in PO

1. Introdução

A aptidão para traduzir fenômenos ou problemas oriundos de situações reais em um conjunto de relações matemáticas (modelo matemático) é uma habilidade chave para os profissionais/pesquisadores da área de Pesquisa Operacional. O estudo de quebra-cabeças pode ser uma ferramenta útil no desenvolvimento dessa habilidade, enquanto propicia, ao mesmo tempo, uma experiência manejável e instigante. Manejável devido ao fato das regras do jogo fornecerem todas as informações necessárias para a criação do modelo, e a familiaridade com os atributos mais importantes poder ser adquirida, em um tempo razoavelmente curto, manipulando o jogo. Instigante porque o gosto por resolução de quebra-cabeças é uma característica comum entre muitos estudantes e profissionais de pesquisa operacional, conforme pode ser observado na entrevista do pesquisador Leandro Coelho no canal *Subject to* do Youtube (Subramanian [2021]).

Neste trabalho são considerados especificamente quebra-cabeças que podem ser modelados a partir de técnicas de programação matemática. Muitos quebra-cabeças elegíveis pertencem à classe NP-completo ([Graham e Parkes, 2008]), o que pode ser considerado um indício de serem significativamente desafiadores. Nos últimos anos, um dos mais emblemáticos exemplos do uso de quebra-cabeça para exercitar a aplicação de técnicas de programação matemática é o Sudoku ([Chlond, 2005], [Coelho e Laporte, 2014], [Takano et al., 2015]. Outros jogos também foram objeto de estudo para o desenvolvimento de modelos matemáticos.

[Trindade et al., 2011] apresentam dois modelos baseados em programação linear inteira mista para a solução do jogo Shisen-Sho, sendo um com o objetivo de encontrar uma sequência factível para retirar todas as peças do tabuleiro e outro com o fim encontrar a sequência que induz ao menor número possível de movimentos. Em [Melkonian, 2016] é proposto um modelo de programação inteira para resolver o jogo Kenken, um quebra-cabeça que tem por objetivo de preencher uma grade com números, similar ao Sudoku.

Considerando trabalhos que modelam jogos lógicos com técnicas de programação matemática, dois trabalhos interessantes são [Hartmann, 2018] e [Hartmann, 2019], nos quais são abordados jogos lógicos disponíveis em aplicativos de celulares. Mais recentemente, [Iranpoor, 2021] apresenta dois modelos para um quebra-cabeça que utiliza peças de xadrez, no caso, cavalos que devem trocar de lugar no tabuleiro obedecendo as regras usuais de movimento. O modelo baseado em programação binária se mostrou ineficiente no que se refere ao tempo computacional e uso de memória. O segundo modelo, baseado em fluxo em rede, por sua vez, apresentou bom desempenho, tanto em tempo computacional como uso de memória. Esses modelos foram utilizados pelo autor para ensinar técnicas de modelagem matemática em um curso de programação matemática.

Neste trabalho são tratados dois quebra-cabeças: Hitori e Heyawake. Os dois jogos foram popularizados no início da década de 90 pela revista especializada em jogos lógicos *Puzzle Communication Nikoli*. Estes jogos, além de serem interessantes por méritos próprios, compartilham regras que não são trivialmente modeladas com técnicas de programação matemática. Estas características, tomadas em conjunto, propiciam uma conjuntura adequada para utilizar estes quebra-cabeças como material didático em um curso de programação matemática.

O trabalho esta organizado na forma que segue. Nas seções 2 e 3 os problemas Hitori e Heyawake são apresentados e os respectivos modelos são desenvolvidos, respectivamente. Os resultados computacionais são apresentados e discutidos na seção 4 e as conclusões na seção 5.

2. Hitori

Hiroti é um quebra-cabeça publicado em março de 1990 na revista *Puzzle Communication Nikoli*, especializada em jogos lógicos. O termo Hitori significa, literalmente, "deixe-me sozi-

nho"que, de certa forma, faz menção ao objetivo do jogo que é eliminar números de uma grade ou tabuleiro. Assim como muitos outros jogos lógicos, Hitori pertence à classe NP-Completo ([Hearn e Demaine, 2009]).

O quebra-cabeça Hitori começa com todos os números em uma grade com dimensões $m \times n$, em que geralmente $m = n$, ou seja, o tabuleiro é quadrado. O objetivo é eliminar algumas números "sombreando"células de modo que não haja nenhum número duplicado em nenhuma fileira ou coluna. O objetivo deve ser alcançado considerando as seguintes regras:

- **Regra 1**: Nenhuma linha ou coluna pode apresentar células com números não sombreados repetidos.
- **Regra 2**: Duas células adjacentes, horizontalmente ou verticalmente, não podem ser sombreadas.
- **Regra 3**: As células não sombreadas devem formar uma área contínua. Ou seja, deve ser possível conectar as células não sombreadas por uma curva composta apenas por segmentos que conectam células adjacentes (horizontalmente ou verticalmente).

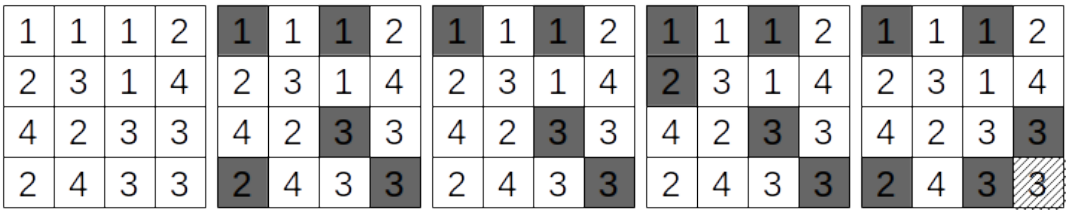


Figura 1: **(a)** Instância de um Hitori 4×4. **(b)** Solução para a instância em (a). **(c)** Regra 1 não é satisfeita, existem dois algarismos 2 na primeira coluna. **(d)** Regra 2 não é satisfeita, existem duas células sombreadas ortogonalmente adjacentes. **(e)** Regra 3 não é satisfeita, um célula está ilhada

A Figura (1) apresenta algumas configurações de uma instância 4×4 que procuram ilustrar as regras considerando situações nas quais estas não são satisfeitas. A instância é exposta em (a) e a única solução possível em (b), enquanto em (c), (d) e (e) as regras 1, 2 e 3 são consideradas, respectivamente.

Das três regras do jogo, a terceira é a mais desafiadora para ser modelada com técnicas de programação matemática. Para garantir que a solução encontrada satisfaça a Regra 3, foram consideradas restrições baseadas em fluxo em rede. Procedimentos de pré-processamento são necessários para identificar as células passíveis de serem sombreadas, ou seja, as células que apresentam números repetidos. As células identificadas dão origem a definição de diferentes conjuntos utilizados na elaboração do modelo proposto, que é especificado a seguir.

Parâmetros:

I: conjunto dos índices das linhas da grade;

J: conjunto dos índices das colunas da grade;

Q: Conjunto das posições $(i, j) \in I \times J$ das células na grade;

S: Conjunto das posições $(i, j) \in I \times J$ das células passíveis de serem sombreadas;

A: conjunto dos números considerados nas células da grade;

L_{ia}: Conjunto das posições das células com o número $a \in A$ na linha $i \in I$;

C_{ja}: Conjunto das posições das células com o número $a \in A$ na coluna $j \in J$;

$N_{(ij)}$: Conjunto das posições das células ortogonalmente adjacentes à célula $(i, j) \in Q$;

(i_0, j_0): posição da célula fonte do fluxo em rede;

M: um número suficientemente grande.

Variáveis

$$x_{(i,j)} = \begin{cases} 1 & \text{se a célula na posição } (i,j) \in S \text{ deve ser sombreada} \\ 0 & \text{caso contrário} \end{cases}$$

$y_{(i,j),(l,k)}$: quantidade de fluxo enviado da célula $(i, j) \in Q$ para a célula $(l, k) \in N_{(i,j)}$.

Modelo Hitori

$$\text{Min} \sum_{(i,j)\in S} x_{(i,j)} \tag{1}$$

Sujeito a:

$$\sum_{(i,j)\in L_{ia}} x_{(i,j)} \geq |L_{ia}| - 1 \qquad \begin{array}{l} \forall i \in I, \\ \forall a \in A : |L_{ia}| > 1 \end{array} \tag{2}$$

$$\sum_{(i,j)\in C_{ja}} x_{(i,j)} \geq |C_{ja}| - 1 \qquad \begin{array}{l} \forall j \in J, \\ \forall a \in A : |C_{ja}| > 1 \end{array} \tag{3}$$

$$x_{(i,j)} + x_{(l,k)} \leq 1 \qquad \begin{array}{l} \forall (i,j) \in Q, \\ \forall (l,k) \in N_{(i,j)} : \\ l \geq i \ \wedge k \geq j \end{array} \tag{4}$$

$$\sum_{(l,k)\in N_{(i_0,j_0)}} y_{(i_0,j_0),(l,k)} = |Q| - 1 - \sum_{(i,j)\in S} x_{(i,j)} \tag{5}$$

$$\sum_{(l,k)\in N_{(i,j)}} y_{(l,k),(i,j)} - \sum_{(l,k)\in N_{(i,j)}\setminus\{(i_0,j_0)\}} y_{(i,j),(l,k)} = \begin{cases} 1\,, & \text{se } (i,j) \notin S \\ 1 - x_{(i,j)}\,, & \text{se } (i,j) \in S \end{cases} \qquad \forall (i,j) \in Q \setminus \{(i_0,j_0)\} \tag{6}$$

$$\sum_{(l,k)\in N_{(i,j)}} y_{(l,k),(i,j)} \leq \left(1 - x_{(i,j)}\right) M \qquad \forall (i,j) \in S \tag{7}$$

$$x_{(i,j)} \in \{0, 1\} \qquad \forall (i,j) \in S \tag{8}$$

$$y_{(i,j),(l,k)} \geq 0 \qquad \begin{array}{l} \forall (i,j) \in Q, \\ \forall (l,k) \in N_{(i,j)} \end{array} \tag{9}$$

A função objetivo em (1) considera a minimização do número de células passíveis de serem sombreadas. É oportuno observar que o modelo não necessariamente demanda uma função objetivo, uma vez que originalmente as regras do jogo exigem que seja encontrada ao menos uma solução factível e não especificamente uma solução com o menor número possível de células sombreadas. As restrição (2) e (3) correspondem a Regra 1, e determinam que um dado número a, com $|L_{ia}|$ ocorrências na linha i e C_{ja} ocorrências na coluna j, apareça apenas uma vez na linha i e coluna j. Para tanto, a restrição impõe a condição que, no mínimo, $|L_{ia}| - 1$ e $|C_{ja}| - 1$ células correspondentes ao número a sejam sombreadas. A restrição (4) refere-se a Regra 2, e impede que células vizinhas ortogonais sejam sombreadas.

As restrições (5),(6) e (7) são as restrições de fluxo em rede referentes a Regra 3. Inicialmente, é necessário considerar como fonte uma célula (i_0, j_0) que sabidamente não será sombreada por não corresponder a um número repetido. Esta fonte injeta na rede determinada pelas células da grade um fluxo de $|Q| - 1 - \sum_{(i,j)\in S} x_{(i,j)}$ unidades. Ou seja, o número de células na grade menos célula fonte e as células sombreadas, restrição (5). As células que não pertencem ao conjunto S absorvem uma unidade de fluxo, enquanto as células em S absorvem uma unidade de fluxo caso não sejam sombreadas, restrição (6). Ainda, uma célula sombreada não pode participar da rede de fluxo. Esta configuração é garantida a partir da proibição de uma célula sombreada receber fluxo, restrição (7). Estas restrições tomadas em conjunto impõem que cada célula não sombreada seja alimentada pela fonte e, por consequência, quaisquer duas células não sombreadas serão conectadas por um caminho até a fonte ou que tenha a fonte como uma das células neste caminho.

Como exemplo, considere que a célula $(1, 2)$ da Figura 1 é a fonte que injeta na rede $16 - 1 - 5 = 10$ unidades de fluxo (16 células menos a célula fonte, menos 5 células sombreadas). Todas as células não sombreadas demandam uma unidade de fluxo que só pode ser suprida a partir de uma conexão em rede com a célula fonte. Com isto, todas as células não sombreadas são conectadas por um caminho até a fonte e, por consequência, para os fins específicos, também são conectadas entre si, formando a área contínua exigida pela Regra 3.

O modelo foi idealizado para tratar tabuleiros quadrados, mas pode ser facilmente estendido para considerar tabuleiros retangulares ou até mesmo outros formatos. O número de variáveis e restrições são dependentes não só da dimensão n do tabuleiro como também do número de células passiveis de serem sombreadas, ou seja, do número de elementos do conjunto S ($|S|$).

Para um tabuleiro com dimensão n, a quantidade de variáveis é limitada pela expressão $5n^2 - 4n$. No que se refere as restrições o limitante superior é dado pela expressão $\frac{9n^2-4n}{2}$.

3. Heyawake

Heyawake é um quebra-cabeça publicado pela primeira vez em em setembro de 1992 na revista especializada em jogos lógicos *Puzzle Communication Nikoli*. Uma tradução possível para Heyawake corresponde a "salas divididas". O jogo é apresentado em uma grade retangular na qual são delimitadas regiões (salas) de tamanhos variados por linhas em negrito. O objetivo reside em determinar quais células devem ser sombreadas ou não, obedecendo algumas regras impostas. Cada região pode conter um número que informa a quantidade de células a serem sombreadas, caso não tenha nenhum número indicado, pode-se sombrear qualquer quantidade de células dentro da região. Em [Holzer e Ruepp, 2007] é apresentada uma prova que o Heyawake pertence à classe NP-Completo. A prova considera a redução ao Problema de Satisfatibilidade Booliana.

As regras para obter uma solução são as que seguem:

- **Regra 1**: As regiões que contêm números devem ter exatamente o número indicado de células

sombreadas;

- **Regra 2**: Regiões que não tiverem nenhum valor indicado podem ter qualquer número de células sombreadas;
- **Regra 3**: As células sombreadas não podem ser adjacentes ortogonalmente, tanto verticalmente quanto horizontalmente;
- **Regra 4**: Uma linha reta ligando células não sombreadas não deve perpassar mais de duas regiões;
- **Regra 5**: As células não sombreadas devem formar uma área contínua. Ou seja, deve ser possível conectar as células não sombreadas por uma uma curva composta apenas por segmentos que conectam células adjacentes (horizontal ou verticalmente).

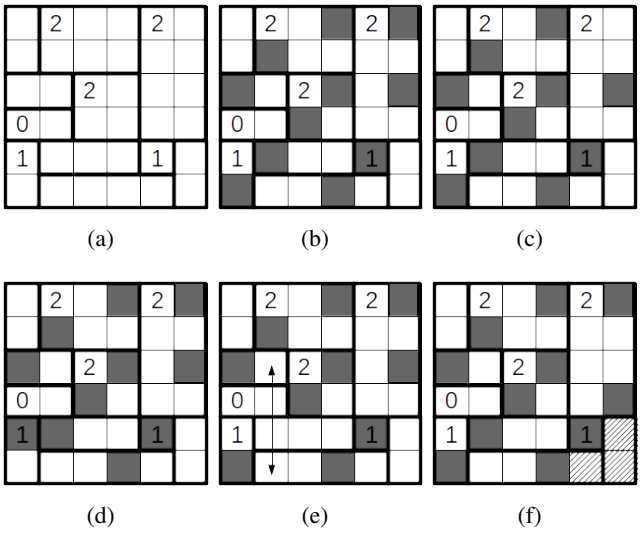


Figura 2: **(a)** Instância de um Heyawake 6×6. **(b)** Solução para a instância em (a). **(c)** Regra 1 não é satisfeita, um região não tem o número exigido de células sombreadas. **(d)** Regra 3 não é satisfeita, existem duas células sombreadas ortogonalmente adjacentes. **(e)** Regra 4 não é satisfeita, é possível determinar uma reta que conecta células não sombreadas que pertencem a 4 regiões distintas. **(f)** Regra 5 não é satisfeita, três células estão ilhadas

A Figura (2) apresenta um problema (instância) do Heyawake em um tabuleiro com dimensões 6×6 e a única solução possível, respectivamente, em (a) e (b). A instância é composta por 11 regiões e exige que, no mínimo, 8 células sejam sombreadas. A solução em (b) conta com 11 células sombreadas devido a Regra 4. Em (c), (d), (e) e (f) são apresentadas situações em que as regras 1, 3, 4 e 5 não são satisfeitas. A Regra 2, por não ser uma regra restritiva, por assim dizer, não apresenta um maior desafio para ser modelada. Na prática, basta suprimir qualquer restrição para o número de células a serem sombreadas para as regiões que não possuem tal número expressamente indicado.

Nota-se que algumas regras do Heyawake se equiparam as regras do Hitori. De fato, as regra 3 e 5 do primeiro correspondem as regras 2 e 3 do segundo, respectivamente. Apesar das

similaridades, algumas definições são reproduzidas na formalização do modelo de forma a deixar a apresentação autocontida. O modelo proposto para o Heyawake é definido da forma que segue.

Parâmetros:

I: Conjunto dos índices das linhas da grade;

J: Conjunto dos índices das colunas da grade;

Q: Conjunto das posições $(i, j) \in I \times J$ das células na grade;

$N_{(ij)}$: Conjunto das posições das células ortogonalmente adjacentes à célula $(i, j) \in Q$;

R: Conjunto dos índices das regiões (salas);

C_r: Conjunto de células $(i, j) \in Q$ dentro de uma região $r \in R$;

n_r: Número de células que devem ser sombreadas na região $r \in R$;

F: Conjunto dos índices das regiões sem indicação de número de células a serem sombreadas;

$L_{i(r_1,r_2,r_3)}$: Conjunto da posição da última célula da região r_1, todas as posições das células da região r_2 e a primeira célula da região r_3 na linha $i \in I$, em que r_1, r_2 e $r_3 \in R$ são regiões contíguas, em sequência, da esquerda para a direita do observador;

$C_{i(r_1,r_2,r_3)}$: Conjunto da posição da última célula da região r_1, todas as posições das células da região r_2 e a primeira célula da região r_3 na coluna $j \in J$, em que r_1, r_2 e $r_3 \in R$ são regiões contíguas, em sequência, de cima para baixo;

M: um número suficientemente grande.

Variáveis

$$x_{(i,j)} = \begin{cases} 1 & \text{se a célula na posição } (i, j) \in Q \text{ for sombreada} \\ 0 & \text{caso contrário} \end{cases}$$

$y_{(ij),(lk)}$: quantidade de fluxo a ser enviada da célula $(i, j) \in Q$ para a célula $(l, k) \in N_{(i,j)}$

$$z = \begin{cases} 1 & \text{se o fluxo total da célula } (1, 1) \text{ é direcionado para a célula } (1, 2) \\ 0 & \text{caso contrário} \end{cases}$$

Modelo Heyawake

$$\text{Min} \sum_{(i,j)\in Q} x_{(i,j)} \tag{10}$$

Sujeito a:

$$\sum_{(i,j)\in C_r} x_{(i,j)} = n_r \qquad \forall r \in R \setminus F \tag{11}$$

$$x_{(i,j)} + x_{(l,k)} \leq 1 \qquad \begin{array}{l} \forall (i, j) \in Q, \\ \forall (l, k) \in N_{(i,j)} : \\ l \geq i \ \wedge k \geq j \end{array} \tag{12}$$

$$\sum_{(i,j)\in L_{i,(r_1,r_2,r_3)}} x_{(i,j)} \geq 1 \qquad \forall r_1, r_2, r_3 \in R, \; \forall i \in I : \left|L_{i,(r_1,r_2,r_3)}\right| > 0 \tag{13}$$

$$\sum_{(i,j)\in C_{j,(r_1,r_2,r_3)}} x_{(i,j)} \geq 1 \qquad \forall r_1, r_2, r_3 \in R, \; \forall j \in J : \left|C_{j,(r_1,r_2,r_3)}\right| > 0 \tag{14}$$

$$\sum_{(l,k)\in N_{(1,1)}} y_{(1,1),(l,k)} = |Q| - 1 - \sum_{(i,j)\in Q\setminus\{(1,1)\}} x_{(i,j)} \tag{15}$$

$$z = 1 - x_{(1,1)} \tag{16}$$

$$y_{(1,1),(1,2)} \geq |Q| - 2 - \sum_{(i,j)\in Q\setminus\{(1,1)\}} x_{(i,j)} - Mz \tag{17}$$

$$\sum_{(l,k)\in N_{(i,j)}} y_{(l,k),(i,j)} - \sum_{(l,k)\in N_{ij}\setminus\{(1,1)\}} y_{(i,j),(l,k)} = 1 - x_{(i,j)} \qquad \forall (i,j) \in Q \setminus \{(1,1)\} \tag{18}$$

$$\sum_{(l,k)\in N_{(i,j)}} y_{(l,k),(i,j)} \leq \left(1 - x_{(i,j)}\right) M \qquad \forall (i,j) \in Q \tag{19}$$

$$x_{(i,j)} \in \{0,1\} \qquad \forall (i,j) \in Q \tag{20}$$

$$y_{(ij),(l,k)} \geq 0 \qquad \forall (i,j) \in Q, \forall (l,k) \in N_{(i,j)} \tag{21}$$

A função objetivo (10) minimiza o número de células sombreadas. Assim como o Hitori, não existe a necessidade estrita de função objetivo, uma vez que as regras do jogo exigem que uma solução factível qualquer seja encontrada. A restrição (11) descreve as regras 1 e 2 do jogo, impondo que a quantidade de células sombreadas dentro de uma região $r \in R$ deve ser igual ao valor indicado, n_r. No caso de não haver indicação para o número de células sombreadas, então nenhuma restrição é imposta e a solução encontrada pode apresentar o número necessário de células sombreadas para estas regiões de modo a satisfazer as demais regras.

A restrição (12) corresponde a Regra 3, e impede que células sombreadas sejam vizinhas ortogonalmente. As restrições (13) e (14) correspondem a Regra 4 considerando linhas e colunas, respectivamente. Caso sejam identificadas células que pertençam a três regiões vizinhas em uma linha ou coluna, uma dessas células, obrigatoriamente, deve ser sombreada. Na Figura (2) (e), na coluna 2, existem duas ocorrências. Na primeira, uma das células nas posições (2,3), (2,4), e (2,5) deve ser sombreada, e na segunda ocorrência, umas das células nas posições (2,4), (2,5) e (2,6) deve ser sombreada. É possível verificar que se a célula (2,4) ou (2,5) for sombreada, a regra é respeitada nas duas ocorrências. Na primeira linha existe uma ocorrência da Regra 4, e, obrigatoriamente, uma das células (1,1), (1,2), (1,3),(1,4) e (1,5) deve ser sombreada. Na solução apresentada na Figura (2) (b) as células (1,4) e (2,5) são sombreada, e satisfazem os exemplos considerados das restrições que correspondem a Regra 3.

As restrições (15) a (19) correspondem a Regra 5. A explicação da sinergia entre essas restrições é muito semelhante ao já exposto para as restrições do modelo Hitori referentes a Regra 3 daquele jogo. No entanto, faz-se necessário esclarecer algumas diferenças. No jogo Heyawake não é possível saber *a priori* qual célula, com certeza, não será sombreada. Por este motivo a célula $(1,1)$ é escolhida como fonte do fluxo em rede, restrição (15). Caso a célula $(1,1)$ seja sombreada, então todo o fluxo é desviado para a célula $(1,2)$ que, devido a Regra 3, certamente não será sombreada,

restrições (16) e (17). Deste modo, a fonte passa a ser a célula $(1, 2)$ e o todo o fluxo ocorre da mesma forma como explicado para o modelo Hitori. Para exemplificar, suponha que a célula $(1, 1)$ seja sombreada, então, pela restrição (16), a variável z assume valor 1 e a restrição (17) assume a forma

$$y_{(1,1),(1,2)} \geq |Q| - 2 - \sum_{(i,j)\in Q\setminus\{(1,1)\}} x_{(i,j)}.$$

Neste caso, é possível observar que todo o fluxo da célula $(1, 1)$ para para a célula $(1, 2)$, já descontado o valor unitário absorvido pela célula $(1, 2)$.

Diferente do modelo do Hitori, a quantidade de variáveis do modelo Heyawake é dependente somente da dimensão n do tabuleiro e limitado pela expressão $5n^2 + 4n$. Para as restrições, que são dependentes das características do tabuleiro além da dimensão n, o limitante superior é dado por $5n^2 - 2n + 2$, um pouco maior se compararmos ao Modelo do Hitori.

4. Resultados computacionais

Os experimentos computacionais foram executados em uma máquina Intel Core i5, 2.5 GHz com 8GB de memória RAM com sistema operacional Ubuntu 20.04.2 LTS. Ambos os modelos foram implementados utilizando a linguagem de programação Julia 1.6.0 através da biblioteca JuMP v0.21.8 e o *solver* IBM ILOG CPLEX 12.10, com o conjunto padrão de parâmetros.

As instâncias para o jogo Hitori foram obtidas no site https://es.puzzle-hitori.com/ e pertencem ao nível difícil, de acordo com a classificação indicada no site. As instâncias utilizadas e o código fonte em Julia para construção do modelo estão disponíveis no endereço https://github.com/SuzianneTormann/Hitori .

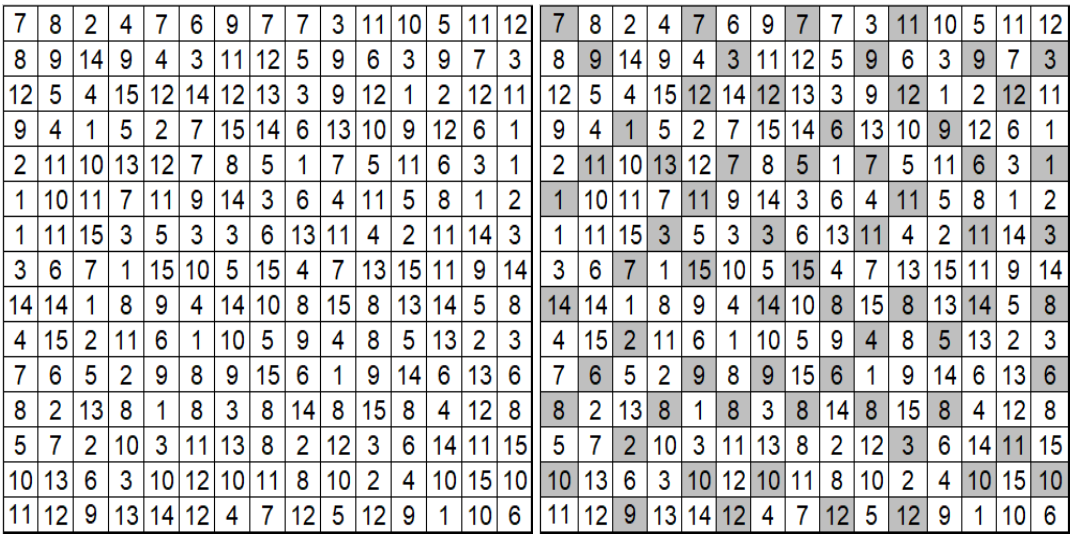
A Tabela 1 apresenta as instâncias (coluna 1), o número de variáveis (colunas 2 e 3), o número de restrições (coluna 4) a função objetivo e o tempo computacional (colunas 5 e 6, respectivamente). É possível ver que a quantidade de variáveis cresce quadraticamente em função da dimensão n do tabuleiro, assim como a quantidade de restrições. No entanto, o mesmo crescimento não é observado no tempo computacional. De fato, para todas as instâncias a ordem de grandeza do tempo computacional não é maior do que décimos de segundos. A função objetivo apresenta o número mínimo de células que devem ser sombreadas para se obter uma solução factível. De uma certa forma, o valor da função objetivo mensura o esforço manual necessário para finalizar o jogo, uma vez que quanto maior a quantidade de células que devem ser sombreadas, pode-se imaginar que maior será o tempo para construir uma solução factível.

Resultados Computacionais Hitori

Instância	**# variáveis**		**# restrições**	**Função objetivo**	**Tempo computacional (s)**
	$x_{(ij)}$	$y_{(ij)(lk)}$			
Hitori-15x15	128	840	525	66	0,17
Hitori-20x20	225	1520	931	120	0,06
Hitori-30x30	524	3480	2187	271	0,34
Hitori-35x35	701	4760	2902	369	0,33

Tabela 1: Especificações das instâncias Hitori e tempo computacional.

A Figura 3 apresenta a configuração inicial e a solução encontrada para a instância Hitori-15x15. A solução apresenta 66 células hachuradas de cinza, que corresponde ao valor da função objetivo apresentado na Tabela 1.



(a) (b)

Figura 3: Exemplo do Problema e Solução do Hitori-15x15 (a) Problema Hitori-15x15 e (b) Solução Hitori-15x15.

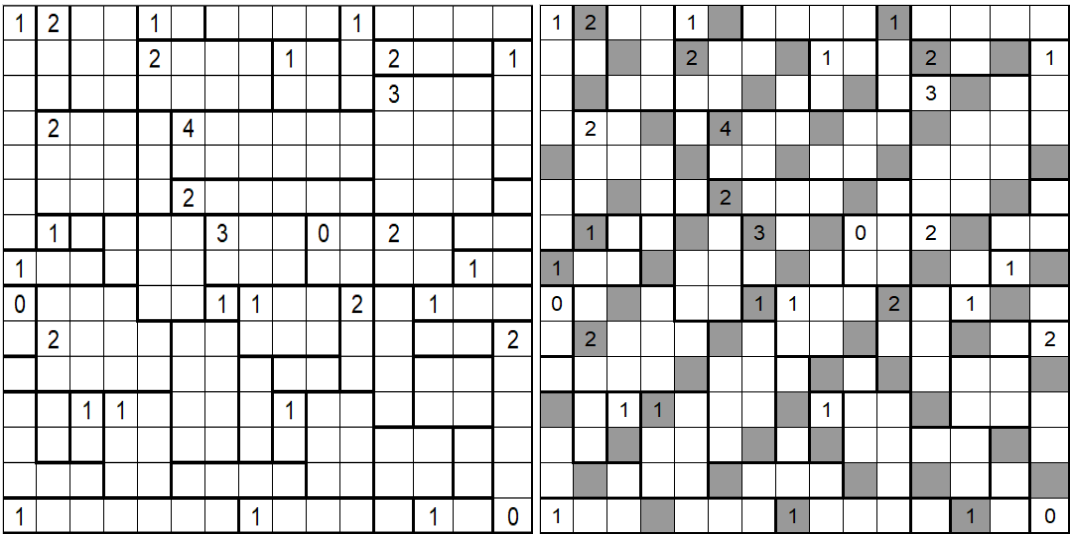
Os testes computacionais para o Modelo Heyawake seguiram a mesma lógica utilizada nos testes para o Modelo Hitori. As instâncias foram obtidos no site https://es.puzzle-Heyawake.com/ e pertencem ao nível difícil, segundo a classificação do próprio site. As instâncias e o código fonte em Julia para construção do modelo estão disponíveis no endereço https://github.com/SuzianneTormann/Heyawake.

A Tabela 2 apresenta as instâncias utilizadas para os testes do modelo Heyawake, o número de variáveis, o número de restrições geradas, o valor da função objetivo, que corresponde a quantidade de células sombreadas, e o tempo computacional, nesta ordem. É possível observar, considerando tabuleiros de mesma dimensão, que a quantidade de variáveis $\mathbf{x}$ é maior do que a do modelo Hitori devido ao fato de todas as células serem passíveis de sombreamento. O número de restrições também é superior. Isso é facilmente explicável devido ao número maior de regras do Heyawake em relação ao Hitori. Interessante observar que a quantidade de células sombreadas (função objetivo) são semelhantes nos dois jogos. O tempo computacional, também neste caso, é da ordem de décimos de segundos.

Resultados Computacionais Heyawake					
Instância	**# variáveis**		**# restrições**	**Função objetivo**	**Tempo computacional (s)**
	$x_{(ij)}$	$y_{(ij)(lk)}$			
Heyawake-15x15	225	840	1087	63	0,34
Heyawake-20x20	400	1520	1914	116	0,28
Heyawake-30x30	900	3480	4469	266	0,23
Heyawake-35x35	1225	4760	5990	344	0,58

Tabela 2: Especificações das instâncias Heyawake e tempo computacional.

A Figura 4 apresenta a configuração inicial da instância Heyawake-15x15 e a respectiva solução. Na solução, 63 células células são hachuradas de cinza, em acordo com o valor apresentado para função objetivo da Tabela 2.



(a) (b)

Figura 4: Exemplo do Problema e Solução do Heyawake-15x15 (a) Problema Heyawake-15x15e (b) Solução Heyawake-15x15.

Observando as Figuras 3 e 4 é possível perceber que os tabuleiros possuem a mesma dimensão e uma quantidade de células hachuradas (função objetivo) semelhante. No caso do Hitori existem 66 células hachuradas, que corresponde percentualmente a 29% do total de células, e a solução Heyawake apresenta 63 células hachuradas, que corresponde a 28% do total de células. Analisando o restante dos problemas testados nesse trabalho podemos observar que a quantidade de células hachuradas mantém-se entre 25% e 30% .

5. Conclusão

Neste trabalho foram propostos dois modelos de programação inteira mista para a resolução dos quebra-cabeças Hitori e Heyawake. Estes jogos apresentam regras em comum que não podem ser trivialmente traduzidas em relações matemáticas, demonstrando serem exemplos interessantes para uso em cursos que se destinam a ensinar Modelagem Matemática. Adicionalmente, a elaboração dos modelos demanda procedimentos de pré-processamento para identificar algumas características necessárias para a correta interpretação e modelagem das regras do jogo.

Os testes computacionais demonstraram que os modelos resolvem adequadamente os quebra-cabeças considerados, e apresentam uma quantidade de variáveis e restrições manejáveis para resolver instâncias em frações de segundos do tempo necessário para a resolução manual.

Por fim, como proposta para estudos futuros, pretende-se explorar o uso de modelos matemáticos para lançar luz sobre quais características das configurações iniciais são determinantes para definir a dificuldade do jogo. Também, pretende-se utilizar modelos matemáticos para gerar instâncias com diferentes características e graus de dificuldade.

Referências

Chlond, M. J. (2005). Classroom exercises in ip modeling: Su doku and the log pile. *Informs Transactions on Education*, 5(2):77–79.

Coelho, L. C. e Laporte, G. (2014). A comparison of several enumerative algorithms for sudoku. *Journal of the Operational Research Society*, 65(10):1602–1610.

Graham, K. e Parkes, A. (2008). A survey of NP-Complete puzzles. *ICGA Journal*, 31(1):13–34.

Hartmann, S. (2018). Solving smartphone puzzle apps by mathematical programming. *INFORMS Transactions on Education*, 18(2):127–141.

Hartmann, S. (2019). Puzzle — more logic puzzle apps solved by mathematical programming. *INFORMS Transactions on Education*, 20(1):1–55.

Hearn, R. A. e Demaine, E. D. (2009). *Games, puzzles, and computation*. CRC Press.

Holzer, M. e Ruepp, O. (2007). The troubles of interior design-a complexity analysis of the game heyawake. *In: Crescenzi P., Prencipe G., Pucci G. (eds) Fun with Algorithms. FUN 2007. Lecture Notes in Computer Science*, 4475:198–212.

Iranpoor, M. (2021). Knights exchange puzzle — teaching the efficiency of modeling. *INFORMS Transactions on Education*, 21(2):0–7.

Melkonian, V. (2016). An Integer Programming Model for the KenKen Problem. *American Journal of Operations Research*, 06(03):213–225.

Subramanian, A. (2021). Subject to: Leandro coelho. *Youtube*, https://youtu.be/-T0XfXBLzHo, acessado em de 25 maio de 2021.

Takano, K., de Freitas, R., e de Sá, V. (2015). O jogo de lógica sudoku: modelagem teórica, NP-Completude e estratégias algorítmicas exatas e heurísticas. In *Anais do XXXIV Concurso de Trabalhos de Iniciação Científica da SBC*, p. 71–80, Porto Alegre, RS, Brasil. SBC.

Trindade, R. S., Dhein, G., Muller, F. M., e de Araujo, O. C. (2011). Mixed integer linear programming models to solve the Shisen-Sho puzzle. *Proceedings - 2011 Workshop-School on Theoretical Computer Science, WEIT 2011*, p. 108–112.