Programação matemática aplicada à resolução de um jogo tipo quebra-cabeça

Renan S. Trindade[1] Olinto C. B. de Araujo[2] Guilherme Dhein[2]

[1]Curso de Ciência da Computação, Centro Universitário Franciscano, Brasil
[2]Colégio Técnico Industrial de Santa Maria (CTISM), Universidade Federal de Santa Maria, Brasil

Resumo

Shisen-Sho ou Four Rivers é um jogo com informação perfeita cujo objetivo é remover todas as peças de um tabuleiro. Neste trabalho é proposto um modelo de programação inteira mista para resolução de uma variante do Shisen-Sho na qual é necessário otimizar a sequência de movimentos segundo um critério definido pelos autores. Este modelo pode ser utilizado como ferramenta didática para o ensino de programação matemática na disciplina de Pesquisa Operaconal. Resultados computacionais são apresentados considerando diferentes configurações iniciais e dimensões de tabuleiro.

Palavras-chave: programação matemática; jogos tipo quebra-cabeça; otimização

Contato autores:
renanspencer@gmail.com
olinto@ctism.ufsm.br
gdhein@redes.ufsm.br

1. Introdução

Muitos problemas de cunho prático podem ser interpretados como jogos de quebra-cabeça. Um exemplo disto é o problema de carregamento de contêiner, no qual, semelhante a um tetris tridimeniosnal, deve-se carregar caixas em um ou mais contêineres de forma a ocupar o maior volume possível. Ainda que instâncias reais desse problema, até o presente momento, só possam ser adequadamente tratadas com abordagens heurísticas, modelos matemáticos são propostos na literatura de modo a propiciar um estudo adequado da estrutura e complexidade subjacentes [Chen, 1995; Junqueira et al., 2010; Junqueira et al., 2011]. De fato, uma forma de encorajar a criatividade e melhorar a habilidade dos profissionais que constroem modelos matemáticos consiste no treinamento na resolução de jogos de tabuleiro [Sniedovich, 2002 ; DePuy e Taylor, 2007].
Neste trabalho é proposto um modelo matemático de programação inteira mista para resolução de uma variante do jogo Shisen-Sho clássico na qual é necessário otimizar a sequência de movimentos para remover todas as peças do tabuleiro.

2. Trabalhos relacionados

Um estudo sobre jogos combinatórios, considerando a complexidade dos algoritmos de resolução e evidenciando aqueles cuja complexidade ainda não foi apropriadamente determinada, é apresentado em Demaine (2001). Em Takes e Kosters (2009) é proposto um método de resolução para o jogo SameGame bem como uma variação deste denominada Chessbord variant. Uma variante do jogo Peg Solitaire, conhecido na língua portuguesa por "Resta Um", é estudada em Jefferson et al. (2006) a partir de modelos matemáticos de programação inteira e técnica de propagação de restrições. Um modelo de programação linear inteira para resolução do jogo Rummikub, que combina elementos de *rummy* (jogo de cartas), dominó, mahjong e xadrez, é proposto em Den Hertog e Hulshof (2006). Em Lewis (2007) é proposta uma metaheurística simulated annealing para tratar o popular jogo Sudoku.

3. O jogo Shisen-Sho

Shisen-sho, também conhecido por Four Rivers, é um jogo quebra-cabeça de estratégia cujo desafio é identificar peças do mesmo tipo que possam ser associadas e removidas do tabuleiro. Duas peças só podem ser removidas do tabuleiro se puderem ser conectadas com no máximo três segmentos de reta verticais ou horizontais sem intersecção com as demais peças. O objetivo do jogo é remover todas as peças do tabuleiro, muito embora existam configurações para as quais isso é impossível. Este jogo é dito ter informação perfeita porque todas as informações necessárias no momento de tomar uma decisão são conhecidas. Na Figura 1 é apresentado um tabuleiro típico com uma dada configuração inicial. Em (a) são identificados dois movimentos permitidos (1 e 2) e um movimento (3) que não satisfaz as regras do jogo, uma vez que utiliza um caminho formado por mais de três segmentos de reta. Após realizar os movimentos possíveis, as peças envolvidas nesses movimentos são removidas do tabuleiro permitindo que outros movimentos sejam executados, como exemplificado em (b) com os movimentos 4 e 5. Usualmente as peças são as mesmas do popular e tradicional jogo chinês Mahjong.

Para definir uma configuração do jogo Shisen-Sho é necessário considerar as dimensões do tabuleiro, em que m representa o número de linhas e n representa o número de colunas, e a quantidade de tipos diferentes de peças, T. De modo a ser possível retirar todas as peças do tabuleiro, a partir de associações de peças do mesmo tipo duas a duas, o quociente $(m \cdot n)/T$ deve ser um número inteiro e par.

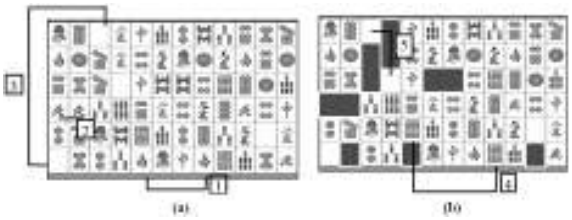

Figure 1: Tabuleiro Shisen-Sho

O número de caminhos possíveis entre duas peças do mesmo tipo é dependente das dimensões do tabuleiro e da posição relativa das peças, e existem, no máximo, m+n+2 caminhos possíveis de associação para remover estas peças do tabuleiro.

A resolução do Shisen-Sho clássico não apresenta um desafio computacional relevante, uma vez que o objetivo é a busca por uma sequência de movimentos qualquer que remova todas as peças do tabuleiro. Neste trabalho é tratada uma variação com conotação de otimização do jogo Shisen-Sho, no qual o objetivo é executar o mais cedo possível o maior número de movimentos sem precedência necessários para remoção de todas as peças do tabuleiro. Movimentos sem precedência são aqueles que podem ser executados de forma independente um do outro, ou seja, podem ser executados ao mesmo tempo ou na mesma rodada, conforme exemplificado na Figura 1 (b).

4. Modelo Matemático

Um modelo para resolução da variação considerada do jogo Shisen-Sho é apresentado a seguir. A posição de cada peça no tabuleiro é indexada por números de 1 a $m \cdot n$ e os parâmetros utilizados são descritos a seguir.

Parâmetros:

I: conjunto de índices da posição das peças no tabuleiro.

E: conjunto de pares de índices $\langle i, j \rangle$ das posições das peças de um mesmo tipo, $i, j \in I$ e $i < j$.

$K_{i,j}$: conjunto de índices dos caminhos que ligam peças nas posições i e j, $\langle i, j \rangle \in E$. Dado um par de peças nas posições i e j, o elemento $k \in K_{i,j}$ representa o caminho de índice k que liga as peças nas posições i e j.

$P_{i,j,k}$: conjunto de índices das posições das peças que compõem o caminho k entre as peças nas posições i e j, $\langle i, j \rangle \in E$, $k \in K_{i,j}$

M: constante suficientemente grande.

Variáveis

$$p_{\substack{i,j,k \\ (i,j)\in E}} = \begin{cases} 1, & \text{se o movimento que retira as peças nas posições de índices } i \text{ e } j \text{ for executado a partir do caminho } k \\ 0, & \text{caso contrário} \end{cases}$$

$c_i \geq 0$, determina a iteração do jogo na qual o movimento envolvendo a peça i é executado.

Modelo $\quad \text{Min} \sum_{i \in I} c_i \qquad (1)$

$$\sum_{\substack{j \in I \\ i<j}} \sum_{k \in K_{i,j}} p_{i,j,k} + \sum_{\substack{j \in I \\ j<i}} \sum_{k \in k_{i,j}} p_{j,i,k} = 1, \ \forall i \in I \qquad (2)$$

$$c_i \geq c_h + 1 + (p_{i,j,k} - 1)M, \quad \forall \langle i, j \rangle \in E, \forall k \in K_{i,j}, \forall h \in P_{i,j,k} \qquad (3)$$

$$c_j \geq c_h + 1 + (p_{i,j,k} - 1)M, \quad \forall \langle i, j \rangle \in E, \forall k \in K_{i,j}, \forall h \in P_{i,j,k} \qquad (4)$$

$$c_i \geq 0, \qquad \forall i \in I \qquad (5)$$

$$p_{i,j,k} \in \{0,1\}, \qquad \langle i, j \rangle \in E, \ \forall k \in K_{i,j} \qquad (6)$$

A função objetivo (1) minimiza o número total de rodadas ponderadas compostas por movimentos sem precedência necessários para remoção de todas as peças do tabuleiro. A restrição (2) determina que somente um caminho, dentre os caminhos possíveis para remover duas peças do tabuleiro, pode ser escolhido para executar um movimento. Devido a esta restrição, caso não seja possível retirar todas as peças do tabuleiro o jogo é considerado sem solução. As restrições (3) e (4) determinam em qual iteração um dado movimento é liberado para ser executado. Estas restrições são semelhantes àquelas utilizadas em problemas de roteamento de veículos para evitar subciclos ou subrotas. As restrições (5) e (6) determinam o domínio das váriaveis de decisão. É interessante observar que, em princípio, as variáveis c deveriam ser definidas como inteiras, no entanto, essas variáveis sempre assumem valores inteiros na solução ótima, dai a definição destas como não negativas. Para este problema, é possível definir $M = (m \cdot n)/2$.

Uma vez que as rodadas não são consideradas explicitamente no modelo, ciclos podem ocorrer quando movimentos com precedência são executados ao mesmo tempo, conforme pode ser observado na Figura 3. Em (a) é apresentada uma configuração na qual os movimentos utilizados para remover as peças do tabuleiro apresentam uma relação de precedência. Desta forma, para executar o movimento que remove as peças do tipo A é necessário remover as peças do

tipo *B*, mas ocorre que para remover as peças do tipo *B* é necessário remover as peças do tipo *A*. Isto caracteriza um ciclo de tamanho 2. Em (b) é apresentada uma configuração na qual os movimentos determinam um ciclo de tamanho 3.

Para a completa resolução, nem todos os $m+n+2$ caminhos entre duas peças quaisquer do mesmo tipo precisam ser considerados.

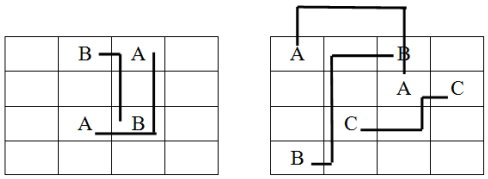

Figure 3: Exemplo de movimentos que determinam um ciclo

Definição: Dado duas peças *i* e *j* é possível que existam caminhos $k, k' \in K_{i,j}$, de modo que $P_{i,j,k} \subset P_{i,j,k'}$. Neste caso, se estabelece uma relação denominada relação de dominância, sendo *k* definido como caminho dominante e *k'* definido como caminho dominado.

A Figura 4 ilustra esta relação. As peças de tipo *A* podem ser ligadas pelos caminhos $k = \{B, C\}$ e $k' = \{B, B, C\}$. O caminho *k* é dominante, pois exige a remoção de apenas duas peças para que se torne possível o movimento envolvendo as peças do tipo *A*. O caminho *k'* exige a remoção de uma peça além das necessárias para o caminho *k*. No caso particular em que duas peças possuem posições adjacentes, todos os demais caminhos são considerados dominados (veja o movimento 2 da Figura 1(a)) e as variáveis que identificam esses caminhos podem ser excluídas do modelo.

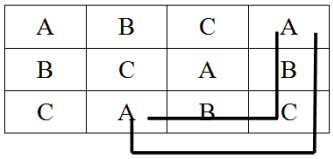

Figure 4: Exemplo de um caminho dominado

5. Resultados Computacionais

Os testes computacionais foram realizados em um computador com processador Intel Quad-Core Xeon X3360 2.83 Ghz, o modelo foi implementado na linguagem de modelagem ZIMPL [Koch, 2004] e como otimizador foi utilizado CPLEX 12.1 com a configuração padrão.

Resultados a partir de exemplos gerados com dados aleatórios demonstraram que o CPLEX não é hábil para tratar instâncias de médio e grande porte.

O uso dos caminhos dominados reduziu em torno de 70% o número de variáveis do modelo matemático. Para obter os dados da Tabela 1 o tempo computacional de resolução foi arbitrariamente limitado em 30 min. O sinal "*" determina que não foi possível provar a otimalidade para todos os casos estudados para a configuração em foco. Os *gaps* de otimalidade foram calculados como *gap* = (melhor limitante – melhor valor) / melhor limitante ·100%.

Tabela 1. Tempo computacional em segundos

m	*n*	*T*	Modelo Matemático
3	4	3	0,06
4	5	5	116,20
5	8	10	1800,00*
5	12	15	1800,00*
6	6	9	1659,51*

Para a configuração 5x8 e 10 tipos de peças em todos os casos foram encontradas soluções factíveis, mas sem a prova de otimalidade. O *gap* médio neste caso foi de 60,89%. Uma solução factível para uma única instância foi obtida para os tabuleiros 5x12 e 15 tipos de peças com um *gap* de 72,0%. Para a configuração 6x6 e 9 tipos de peças, foi provada a otimalidade da solução de 1 instância e encontradas soluções factíveis para as demais 4 com um *gap* médio de 59,1%.

Figura 5. Solução de um tabuleiro 6x6 com nove tipos de peças

A Figura 5 apresenta uma configuração específica para um tabuleiro 6x6 e 9 tipos de peças. No canto superior esquerdo e no canto inferior direito peças do tipo "C" e "G" são destacadas. A disposição destas peças foi idealizada de modo que pelo menos um dos movimentos trivialmente identificáveis no canto inferior não faça parte de qualquer solução factível. Isoladamente, as peças no canto superior formam um interbloqueio (*deadlock*), e pelo menos um movimento conectando as peças do canto superior e do canto inferior deve constar na solução ótima. A figura apresenta todas as rodadas da solução factível obtida pelo CPLEX utilizando o Modelo 2. Não foi possível provar a otimalidade com 30 min de processamento e o *gap* encontrado após o tempo limite foi maior que 66%. O fato dos movimentos em destaque na parte inferior serem atrativos em conjunto como um *gap* superior a média, sugere que o uso de ciclos ou

interbloqueios que obstruam movimentos trivialmente identificáveis pode ser uma forma de identificar configurações com diferentes níveis de dificuldade, semelhante a estudos realizados para o jogo Sudoku [Henz e Truong, 2009; Mantere e Koljonen, 2007].

6. Conclusões

Neste trabalho foi proposto um modelo matemático baseado em programação linear inteira mista para resolução de uma variação do jogo Shisen-Sho. Os testes computacionais demonstram que o modelo é consistente e representa adequadamente o problema estudado e, no atual estágio da pesquisa, somente instâncias de pequeno porte podem ser resolvidas na otimalidade. No entanto, o modelo proposto pode ser útil como motivação para futuros trabalhos explorando métodos de decomposição, métodos baseados em relaxação lagrangeana, propagação de restrições e abordagens heurísticas, de modo a tratar instâncias de maior porte. Ainda, é possível utilizar este problema como motivação em sala de aula, dada a conotação lúdica do jogo e o fato do estudande não depender de fontes externas ou conhecimento prévio para aprofundar os conhecimentos relativos ao problema em si.

A partir da estrutura evidenciada no modelo matemático foi possível inferir características que podem ser utilizadas para identificar configurações com diferentes níveis de dificuldade. Este resultado sugere a possibilidade de estudar, em um trabalho futuro, uma forma de parametrizar e controlar o nível de dificuldade do jogo. Outro desdobramento deste trabalho é a formulação de novas variantes do jogo, como a minimização do número de rodadas, e a construção de uma interface para auxiliar o usuário a jogar as variações do jogo propostas.

Referências

CHEN C. (1995). An analytical model for the container loading problem, *European Journal of Operational Research,* 80 (1), 68-76.

DEMAINE, E. D. (2001). Playing Games with Algorithms: Algorithmic Combinatorial Game Theory, in *Proceedings of the 26th Symposium on Mathematical Foundations in Computer Science*, Lecture Notes in Computer Science, 2136, 18–32.

DEPUY, G. W. E TAYLOR, G. D. (2007). Using Board Puzzles to Teach Operations Research, *INFORMS Trans.* Ed. 7(2), 160-171.

DEN HERTOG D.; HULSHOF P. B. (2006). Solving Rummikub problems by integer linear programming, *Computer Journal*, 49(6), 665-669.

HENZ, M. E TRUONG, H. (2009). SudokuSat-A Tool for Analyzing Difficult Sudoku Puzzles.*In Collection of Tools and Applications with Artificial Intelligence*, 25-35.

JEFFERSON, C., MIGUEL, A., MIGUEL, I. E TARIM, A. (2006). Modelling and Solving English Peg Solitaire, *Computers and Operations Research*, 33(10), 2935-2959.

JUNQUEIRA, L., MORABITO, R. E YAMASHITA, D. S. (2010). Modelos de otimização para problemas de carregamento de contêineres com considerações de estabilidade e de empilhamento. *Pesquisa Operacional*, 30 (1), 73-98.

JUNQUEIRA, L., MORABITO, R. E YAMASHITA, D. S. (2011). Three-dimensional container loading models with cargo stability and load bearing constraints, *Computers & Operations Research,* aprovado para publicação.

LEWIS, R. (2007). Metaheuristics can solve sudoku puzzles, *Journal of Heuristics*, 13 (4), 387-401.

KOCH, T., Rapid Mathematical Programming, *PhD thesis*, Technische Universität Berlin, 2004.

MANTERE, T. E KOLJONEN, J. (2007). Solving, rating and generating Sudoku puzzles with GA, *IEEE Congress on In Evolutionary Computation*, 1382-1389.

SNIEDOVICH, M. (2002). OR/MS Games: 1. A Neglected Educational Resource, *INFORMS Trans. of Ed.*, 2(3), 86-95.

TAKES, F. W. E KOSTERS, W.A. (2009). Solving SameGame and its Chessboard Variant, *21th Benelux Conference on Artificial Intelligence*, 249-256