Algoritmos e Complexidade para Dois Jogos de Blocos

André Castro Ramos[1], Rudini Sampaio[1]

Departamento de Computação, Universidade Federal do Ceará
{andrecr,rudini}@lia.ufc.br

Resumo

É de conhecimento geral que os jogos eletrônicos movimentam um gigantesco mercado e atraem o interesse das pessoas. Além disso, já é antiga a exploração matemática de objetos de entretenimento como o xadrez. Esse tipo de estudo, nas últimas décadas aplicado aos jogos eletrônicos, pode trazer benefícios tanto para a teoria em si quanto para a produção dos próprios jogos. Neste trabalho trazemos resultados sobre problemas extraídos dos jogos Bloxorz e Bobbin 3D. Tanto de algoritmos encontrados por nós para as versões mais simples dos problemas quanto a prova de NP-completude do problema mais geral vindo do Bloxorz.
ÁREA: Teoria e Algoritmos em Grafos (TAG).

Abstract

It's widely known that eletronic gaming is a huge market and a magnet for people's interest. Besides, since long ago people explore entertainment in a mathematical sense like they did with chess. That kind of study, that evolved to be applied to videogames in the last decades, may be beneficial not only for the theory itself, but also for game design. In this work, we present results about problems taken from Bloxorz and Bobbin 3D, two games.We present algorithms for simple versions of the problems and also a NP-completeness proof for the general problem taken from Bloxorz.
AREA: Graph Theory and Algorithms.

1 Introdução

Um dos benefícios de estudar a complexidade computacional de problemas é o enriquecimento da base matemática que se tem para o próprio estudo de novos problemas. Existem aqueles de aplicação prática evidente, mas, mesmo que um dado problema não a tenha, a afirmação inicial torna esse estudo potencialmente bastante frutífero, já que é conhecido o fundamental papel do raciocínio de redução no que se refere a trazer resultados teóricos na área. Neste trabalho há inicialmente uma exposição do que existe de principal a respeito da complexidade computacional de problemas de uma inspiração inusitada: os jogos eletrônicos. Por um lado, tal inspiração traz dificuldade em termos de precisão das definições, mas, por outro, carrega a intuitividade e a tendência a despertar interesse em quem tenha contato com o assunto que é típica de áreas mais práticas do conhecimento. Outro benefício em potencial, no caso em especial de problemas inspirados em jogos, é a melhoria no desenvolvimento desses produtos em termos do ajuste de dificuldade. Em seguida, a partir da próxima seção, são apresentados nossos resultados sobre uma família específica de problemas desse tipo. Os nomes protegidos por direito autoral utilizados aqui o são com propósito acadêmico, portanto de forma legal.

O estudo teórico de objetos entedidos como recreativos tem já um longo histórico. Um caso clássico é o do problema das oito damas, uma questão baseada nas regras do xadrez originalmente proposta por Max Bezzel [Wikipedia, 2013] e estudada por vários matemáticos, incluindo Gauss. No que se refere a estabelecer problemas computacionais a partir de jogos eletrônicos e analisar a complexidade deles sob a ótica da teoria de algoritmos, existem trabalhos relevantes como [Dor e Zwick, 1999], [Demaine et al., 2002] e [Aloupis et al., 2012].

[Aloupis et al., 2012] é um exemplo onde, mesmo não sendo natural traduzir a forma implementada dos jogos, no caso uma coletânea de sucessos licenciados pela Nintendo, em um modelo formal, foi possível estabelecer regras para os problemas e obter resultados. Especificamente, os jogos-base foram os das franquias Mario, Donkey Kong, Legend of Zelda, Metroid e Pokémon e os resultados obtidos foram a NP-completude de cada problema e, em particular naqueles referentes a Zelda, também é apresentado um resultado de $PSPACE$-completude, o que confere a eles, em termos de complexidade computacional, um grau de dificuldade ainda maior. O artigo traz uma caracterização geral para o tipo de problema tratado. Para cada um há regras sobre o que é uma fase e sobre os movimentos válidos e a definição do problema é a pergunta: "para uma dada fase, existe uma sequência de movimentos validos que levam o personagem controlado pelo jogador ao fim dela?".

Já em [Demaine et al., 2002], artigo relacionado ao Tetris, famoso jogo eletrônico, temos teoremas mais clássicos do gênero, que se tratam de provar a NP-completude e a inaproximabilidade para além do fator $p^{(1-e)}$, onde p é o número de peças envolvidas, de versões generalizadas do Tetris. Trata-se de uma situação que favorece um modelo bem mais preciso como o que o artigo em questão apresenta. Em linhas gerais, esse modelo coloca como entrada uma matriz de dimensões variáveis que representa a tela do jogo, que não precisa estar inicialmente em branco, e uma sequência de peças de Tetris, como saída uma sequência de movimentos válidos, noção também definida no modelo, e especifica um problema de acordo com a função objetivo que associa uma saída a um valor. Em particular, o artigo traz 4 funções objetivo: número de linhas eliminadas, número de peças utilizadas até um movimento que cause a derrota, número de movimentos que eliminem 4 linhas simultaneamente e altura máxima de uma posição ocupada da matriz ao longo da sequência de movimentos, sendo que apenas nesse último exemplo o problema é tratado como de minimização. Os principais raciocínios

apresentados no trabalho podem ser entendidos a partir da primeira função.

Em [Dor e Zwick, 1999], associado ao Sokoban, também um jogo bastante conhecido, temos alguns dos primeiros resultados relacionados ao tema. Nesse artigo, existe o cuidado explícito de separar o problema, que imediatamente é classificado como um problema de planejamento de movimentos, do jogo em si. É provada a NP-completude da abstração considerada e a $PSPACE$-completude de uma versão alternativa onde é possível mover 2 blocos de uma vez.

Existe uma conferência, inicialmente trienal e mais recentemente bienal, onde se concentram os trabalhos que carregam usos da teoria de algoritmos que fogem ao formato convencial, o Fun with Algorithms, o que dá mais respaldo a esses esforços.

2 Bloxorz

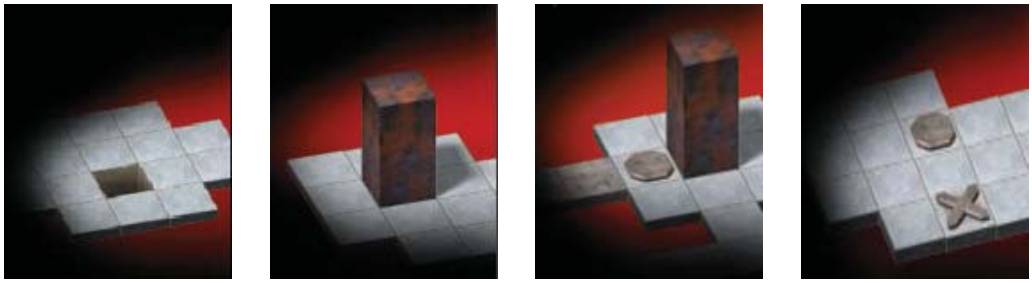
Bloxorz é um jogo eletrônico desenvolvido originalmente para a plataforma Flash pela DX Interactive onde o jogador controla um paralelepípedo de dimensões na escala 2:1:1 e deve levá-lo entre 2 pontos de um cenário. Embora a visualização implementada seja uma projeção que dá noção de profundidade, a mecânica envolvida é totalmente bidimensional. A Figura 1 exemplifica o que pode-se esperar ver no jogo.

Figura 1: Uma visão geral do jogo

O objetivo é fazer o bloco, ou seja, o paralelepípedo, cair no buraco no final, que é ilustrado na Figura 2a, de cada fase. O bloco, mostrado na Figura 2b, pode ser rolado para frente, para trás, para esquerda ou para a direita, mas não pode sair dos limites do piso da fase. Caso isso aconteça, a fase reinicia.

As fases podem ter chaves e pontes como na Figura 2c. As chaves são ativadas quando pressionadas pelo bloco. Pontes então podem mudar de estado e se manter no novo estado, fechadas, por exemplo, mesmo se o bloco for movido. Existem 2 tipos básicos de chaves: pesadas ou leves. As leves, representadas da Figura 2d pela circular, são ativadas quando qualquer parte do bloco as pressiona. As pesadas, que aparecem em forma de "x" como na Figura 2d, são ativadas apenas se o bloco estiver em pé sobre elas.

Quando ativadas, as chaves podem apresentar diferentes comportamentos. Algumas alternarão o estado de pontes entre fechada e aberta a cada uso e outras mudarão o estado de



(a) Final da fase (b) Bloco (c) Chave e ponte (d) Tipos de chave

(e) Madeira (f) Teletransporte (g) Bloco pequeno

Figura 2: Elementos de uma fase

pontes permanentemente.

Piso de madeira, ilustrado na Figura 2e, é mais frágil que o resto do cenário e, se o bloco ficar em pé em cima dele, ele irá cair, tirando o bloco dos limites do piso da fase.

Também existem nas fases as chaves especiais de teletransporte como a que aparece na Figura 2f. Elas teletransportam o bloco para dois lugares diferentes e assim o dividem em dois blocos cúbicos menores, um para cada ponto do cenário. Eles podem ser controlados individualmente e se tornam um bloco normal caso colocados um ao lado do outro.

Blocos pequenos como os da Figura 2g podem ativar chaves leves, mas não chaves pesadas. Além disso, blocos pequenos não podem passar pelo buraco final da fase.

O problema computacional que é estudado nesta seção corresponde à seguinte pergunta:

Dada uma fase hipotética de Bloxorz, ou seja, uma construção arbitrária que respeite as regras expostas acima sobre o que pode haver em uma fase do jogo, mas sem que se considerem possíveis restrições práticas, existe uma sequência de jogadas que leve o jogador a passar de fase?

Exemplos de soluções, ou seja, sequências de jogadas, para fases concretas do jogo estão ilustrados nas Figuras 3 e 4.

Na Figura 3, o número $i > 1$ marca um movimento que vem em sequida do marcado com $i - 1$ na sequência, 1 marca o primeiro movimento, se i está entre dois elementos, marca um movimento que termina no bloco deitado sobre eles dois e, se está centralizado sobre um elemento, marca um que termina em um bloco em pé sobre ele.

Aqui será descrito um algoritmo polinomial para o caso onde não há pontes nem as chaves que as ativam, mas podem existir chaves de teletransporte além de todos os outros elementos. Para resolvê-lo,transformamos o problema em um caso do problema de menor caminho em um grafo direcionado ponderado. A Figura 5 ilustra esse algoritmo e a Figura 6 demonstra a aplicação em uma fase concreta do jogo.

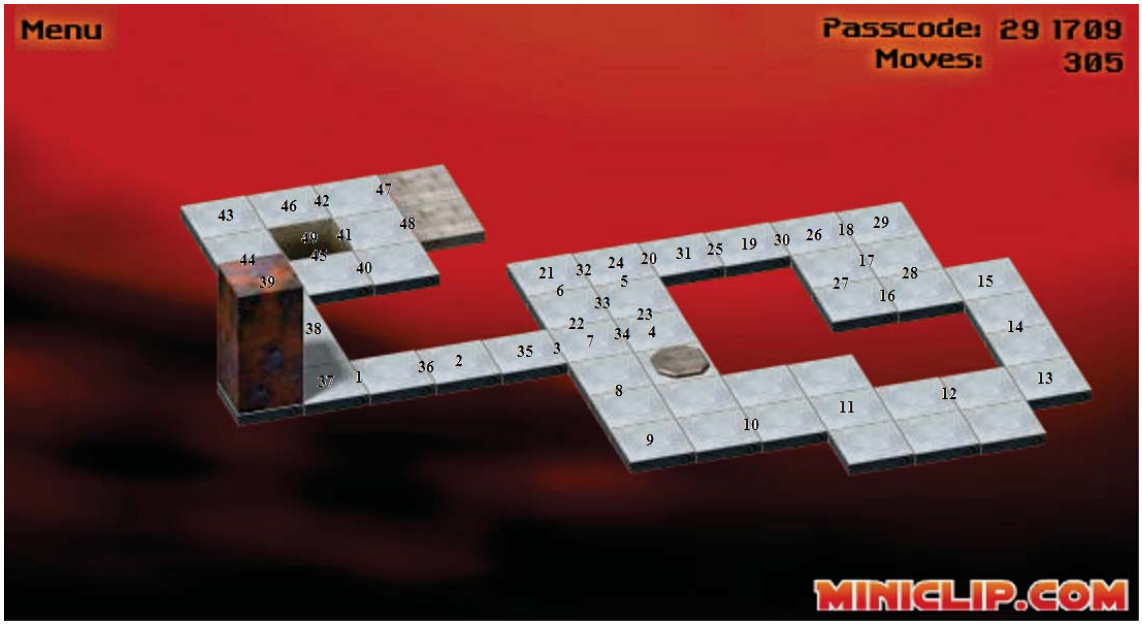


Figura 3: Curioso caso com pontes

Na Figura 5, em azul claro está representado o vertice a correspondente ao bloco em pé sobre o teletransporte, que está ilustrado pelas duas curvas escuras, os vértices para os quais existem arcos que saem de a estão representados em azul escuro, em laranja o valor de w, ou seja, o peso, para cada um desses arcos e em verde limão há indicações das duas posições onde surgirão blocos pequenos quando o teletransporte for ativado. O restante do grafo gerado para uma dada fase são vértices correspondentes a elementos de Bloxorz a não ser pelo piso laranja e arcos que representam os passos possíveis para um bloco de Bloxorz.

Nosso algoritmo constrói um grafo direcionado ponderado a partir da fase. Os vértices são os mesmos, incluindo um vértice para cada chave de teletransporte, interpretaremos aqui todos os vértices como blocos. Os arcos sãos os seguintes:

- $e = (u, v)$ com $w(e) = 1$ $\forall (u, v) \in E^{--}$, onde E^{--} é o conjunto de arcos mapeáveis a partir de movimentos válidos, para um bloco normal, que não envolvam blocos em pé sobre um telestransporte. Esses arcos podem ser entendidos como sendo aqueles que formam uma estrutura análoga a ligar por arestas os movimentos numerados na Figura 4 com a observação de que aqui precisamos contemplar todas as possibilidades sem repetições de blocos, pois um movimento que deixa o bloco em uma situação idêntica a uma anterior pode ser eliminado sem prejuízo, de sequências de movimentos a partir do bloco inicial e a imagem mostra apenas aquela que indica a resposta do problema.

- $e = (u, v)$ com $w(e) = 1$ tal que (u, v) é um arco como no item anterior, mas v está em pé sobre uma chave de teletransporte.

- $e = (u, v)$ com $w(e) = c$ tal que u está em pé sobre uma chave de teletransporte e v está deitado, onde c é o número de passos necessários para, dadas as posições dos blocos pequenos gerados, esses 2 blocos se encontrarem e formarem v, esse número pode ser infinito.

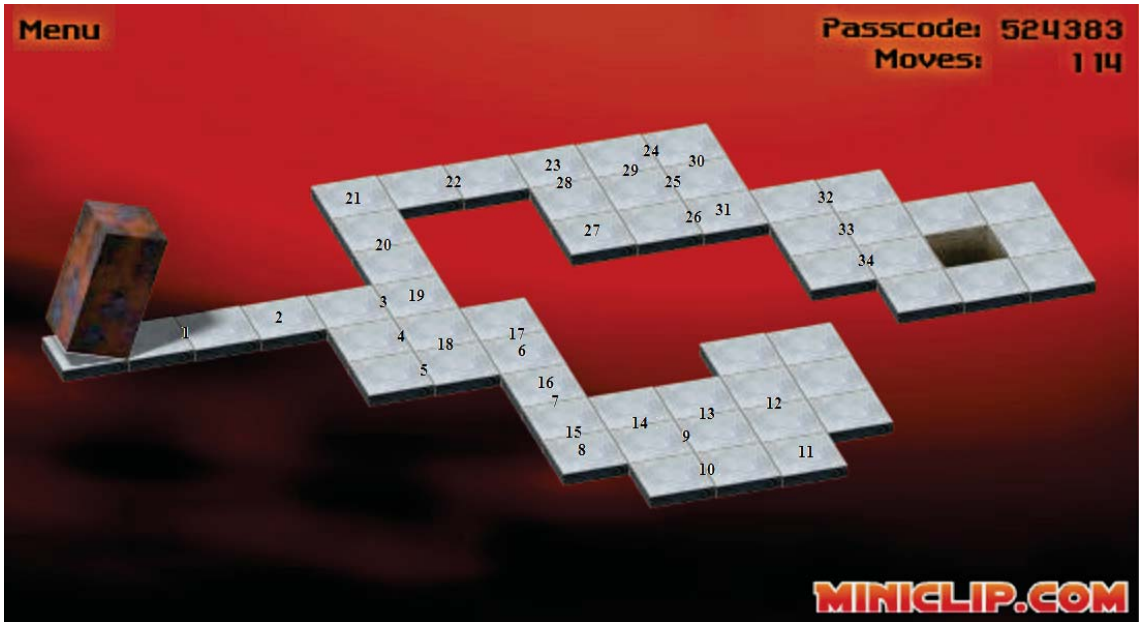


Figura 4: Passos até o final da fase

Construído esse grafo, pode-se aplicar o algoritmo de [Dijkstra, 1959] para encontrar o menor caminho e portanto um caminho de custo finito se existir. Isso conclui nosso algoritmo com a observação de que o cálculo de c para os arcos saídos de um bloco em pé sobre uma chave de teletransporte é feito por uma "subrotina"' que mede as distâncias entre as posições dos blocos cúbicos e as ocupadas pelo bloco final e gera o padrão notado na figura a não ser no caso especial onde não há caminho que os blocos pequenos possam percorrer até as posições finais, nesse caso $c = \infty$. Esse último adendo leva a uma necessidade de resolver um problema similar, mas envolvendo blocos pequenos. Uma abordagem é construir um subgrafo induzido de um grid cujos vértices são correspondentes à posição de cada elemento. Na Figura 6, os devidos caminhos nesse grafo estão representados em verde e amarelo. Observa-se que pode-se trocar a posição final dos blocos de forma que o trajeto de um bloco não interfira no do outro e que existe sempre como evitar encontros dos blocos no trajeto escolhendo movimenos de um antes dos do outro. Portanto pode-se tratar cada um em separado ignorando o outro e testando as duas possibilidades (a não ser que as posições iniciais definidas pela chave para os blocos pequenos sejam vizinhas). Se existe um caminho de custo finito entre o bloco inicial e o bloco em pé sobre o final, o algorimo retorna sim, caso final ele retorna não.

Uma alteração que pode ser feita no algoritmo é calcular primeiramente os custos usando uma estratégia de programação dinâmica. Nossa tabela teria uma célula para cada trio formado por duas posições distintas ocupadas por elementos e um par não-ordenado de elementos vizinhos. O número de pares possíveis de posições distintas é $O(n^2)$ e o de pares de elementos vizinhos $O(n)$, o que leva a uma tabela de tamanho $O(n^3)$, o que também é o limitante assistótico para o tempo de construção dessa tabela. Como segunda parte do algoritmo, restaria construir $O(n^2)$ arestas, parte delas consultando a tabela e parte delas apenas as restrições de movimentos válidos. Logo, nosso algoritmo final executa em tempo $O(n^3)$.

Para finalizar a seção trataremos do problema geral inicialmente descrito nela, que aqui será chamado de $BLOXORZ$. O tópico é o seguinte teorema:

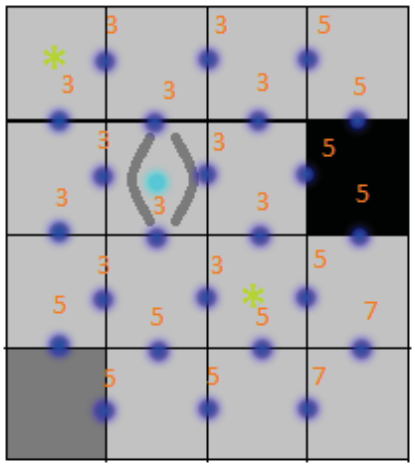


Figura 5: O teletransporte no grafo direcionado ponderado

Teorema 2.1. $SAT \leq_p BLOXORZ$.

Demonstração. Para esta demonstração, não é necessário considerar as chaves de teletransporte. Dada uma fórmula da Lógica Proposicional na CNF, com a observação de que toda fórmula da Lógica Proposicional possui uma equisatisfatível, ou seja, para a qual a resposta de SAT é a mesma, na CNF obtível em tempo polinomial em relação ao tamanho da original, pode-se construir uma fase como na Figura 7.

Na Figura 7, a região cinza mais clara representa a área de piso normal, os retângulos representam pontes fechadas, ou seja, que podem ser atravessadas, os triângulos representam pontes abertas, o quadrado cinza escuro é o bloco inicial da fase, o preto é o buraco final e os círculos indicam chaves leves. Se uma chave tem na figura a mesma cor que uma ponte, ela abre ou fecha a ponte definitivamente quando ativada. Para cada cláusula da fórmula, há uma ponte que pode ser fechada por várias chaves como os triângulos bicolores na figura, as chamaremos de engrenagens de cláusula. Para cada variável da fórmula, há uma região bifurcada como as que são mostradas entre a retangular onde se encontra o bloco inicial e a das engrenagens de cláusula, as chamaremos de engrenagens de variável. Em uma engrenagem de variável, logo à direita de cada ponte inicialmente fechada, há uma chave que quando ativada a abre definitivamente. As chaves restantes das engrenagens de variável fecham definitivamente as engrenagens daquelas cláusulas onde o literal representado por cada uma aparece. Cada engrenagem de variável possui 2 chaves desse tipo, uma representa a própria variável e a outra sua negação.

Se $\varphi = C_1 \wedge C_2 \wedge \ldots \wedge C_n$, onde $\forall 1 \leq i \leq n$, $C_i = l_1 \vee l_2 \vee \ldots \vee l_{m_i}$, onde $\forall 1 \leq j \leq m_i$, $\exists x \in VAR(\varphi)$ tal que $l_j = x$ ou $l_j = \neg x$, é uma fórmula satisfatível, existe uma valoração v para $VAR(\varphi)$ que satisfaz toda cláusula C_i de φ e portanto ao menos um literal l_j de cada C_i. Para a fase construída para φ, existe uma sequência de movimentos que permite atravessar as engrenagens de variável ativando as chaves correspondentes aos literais satisfeitos por v, pois a fase pode ser concretamente construída de forma tal que rolando sucessivamente o bloco para frente ou para trás dependendo da posição ocupada por ele e de $v(x)$ e depois o

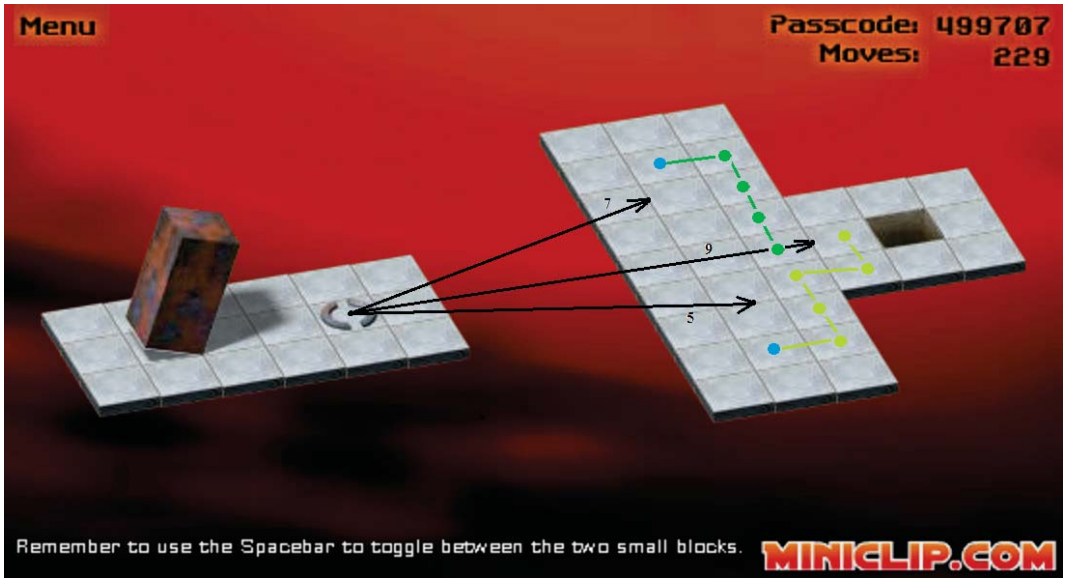


Figura 6: Gerando o grafo para uma fase

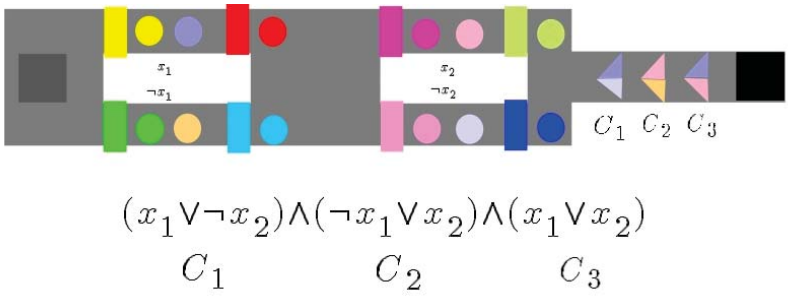


Figura 7: Esquema de redução com chaves

rolando sucessivamente para a direita, que aqui significa ir à direita na Figura 7, alcança-se a chave que abre uma ponte após atravessá-la sem que o último movimento coloque o bloco em uma posição vazia pela ativação da chave, rolando mais uma vez para a direita ativa-se a chave correspondente ao literal satisfeito, mais duas ativa-se a chave após mais uma ponte e então pode-se reiciniar o processo para a próxima engrenagem de variável, que estará à direita da que se acabou de atravessar. Como as chaves de cada literal satisfeito por v foram ativadas, após essa sequência, todas as engrenagens de cláusula foram fechadas, logo pode-se rolar sucessivamente o bloco de uma maneira que alcança a área retangular onde está o final, portanto existe uma sequência-solução para essa fase.

Por outro lado, seja $f(\varphi)$ a fase gerada pela redução em questão para a fórmula na CNF φ, se existe uma sequência-solução para $f(\varphi)$, existe uma sequência de movimentos que fecha todas as engrenagens de cláusula, logo uma que atravessa a região das engrenagens de variável ativando a chave de pelo menos um literal de cada cláusula de φ. Além disso, se $x \in VAR(\varphi)$, não é possível ativar a chave de x e a de $\neg x$ ao mesmo tempo em uma sequência de movimentos sucessivos, pois, se a chave de x foi ativada, a ponte mais próxima à esquerda dela foi atravessada, pois, devido à chave à direita da ponte posterior, não é possível alcançá-la vindo da

direita, e, como existe uma chave logo à direita da primeira ponte que a abre definitivamente, não é possível ativar a chave de x e depois retornar à esquerda para ativar a chave de $\neg x$. Portanto, φ é satisfatível.

□

Colorário: $BLOXORZ$ é NP-completo.

3 Bobbin 3D

Bobbin 3D é um jogo semelhante ao Bloxorz desenvolvido pela Maplabs para o sistema Android. Nele o bloco a ser levado entre 2 pontos do cenário é um cilindro de altura 1 e raio $\frac{1}{2}$. Apesar de trazer uma impressão ainda maior que os anteriores de tridimensionalidade, a mecânica continua sendo bidimensional. A Figura 8 exemplifica o que pode-se esperar ver no jogo:

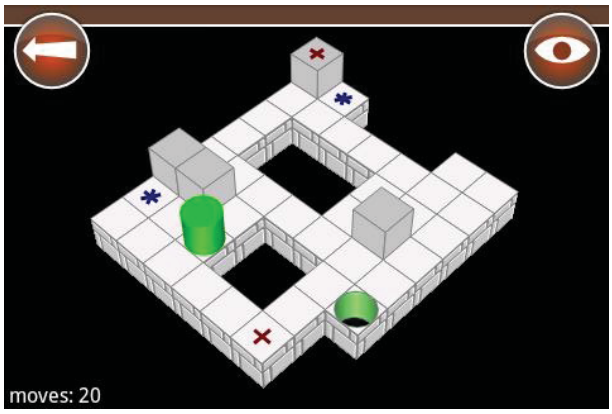


Figura 8: Uma visão geral do jogo

O objetivo é rolar o cilindro a partir da posição inicial até que ele fique em pé sobre o buraco final. O bloco pode ser rolado para frente, para trás, para esquerda ou para a direita. Com a justificativa geométrica de o bloco ser um cilindro, rolar não significa sempre um movimento restrito a uma vizinhança limitada pelas dimensões do objeto. Se o bloco está deitado e é rolado de forma a se manter deitado, o bloco resultante ocupa somente posição anterior ao primeiro obstáculo presente na linha ou coluna.

Em uma fase de Bobbin 3D, pode haver piso normal e piso elevado, que serve de obstáculo quando o bloco rola. Também pode haver uma quantidade par de chaves '+' elevadas ou abaixadas, que, quando elevadas, funcionam como o piso elevado e quando abaixadas alternam o estado de outra chave '+' pré-definida entre elevada e abaixada. Se uma chave '+' alterna o estado de outra, essa outra alterna o estado dela. Além disso, podem existir posições '*' que transformam o bloco cilíndrico em cúbico por uma quantidade determinada de movimentos. Um bloco cúbico ocupa uma única posição e pode ser rolado nas mesmas 4 direções, mas sempre no máximo até uma posição vizinha.

O problema computacional que é estudado nesta seção, corresponde à seguinte pergunta:

Dada uma fase hipotética de Bobbin 3D, ou seja, uma construção arbitrária que respeite as regras expostas acima sobre o que pode haver em uma fase do jogo, mas sem que se considere possíveis restrições práticas, existe uma sequência de jogadas que leve o jogador a passar de fase?

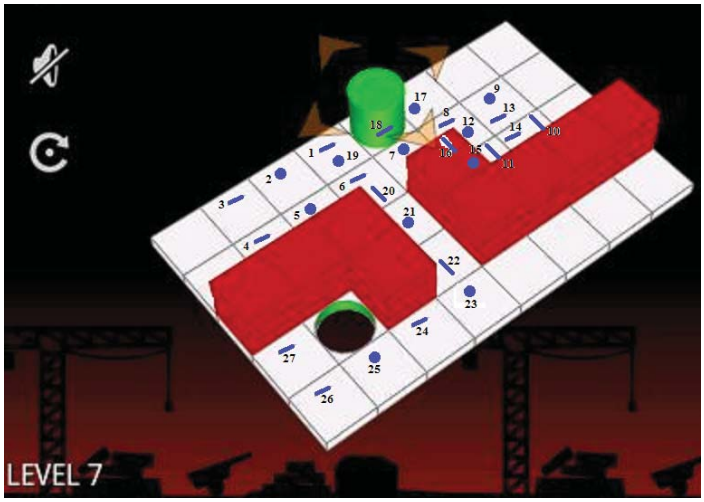


Figura 9: Passos em fase do Bobbin 3D

A Figura 9 ilustra um exemplo de solução para uma fase concreta do jogo.

Para as instâncias do problema sem a presença de chaves '+' ou posições '*', pode-se aplicar a mesma estratégia de nosso algoritmo para o $BLOXORZ^{-}$, como ilustrado na Figura 10. Nela, os quadrados cinzas claros são piso normal, o escuro piso elevado, o círculo cinza escuro demarca a posiçao inicial e o preto o final da fase. Além disso, os círculos vermelhos representam vértices que são blocos em pé, as elipses verdes os que são blocos deitados horizontalmente e a azul um deitado verticalmente. Cada elemento da fase corresponde a um vértice de cada tipo, ou seja, há 3 vértices para cada elemento e os arcos correspondem aos pares de blocos entre os quais um movimento válido é possível. Vale ressaltar que neste problema blocos deitados ocupam apenas uma posição. As linhas amarelas indicam arcos do caminho que corresponde à solução. A seta apenas um arco e as linhas comuns 2 arcos (u, v) e (v, u).

Referências

[Wikipedia, 2013] "Max Bezzel."Wikipedia, The Free Encyclopedia. Wikimedia Foundation, $\langle http://en.wikipedia.org/wiki/Max_Bezzel \rangle$ (2013).

[Aloupis et al., 2012] G. Aloupis, E. D. Demaine e A. Guo, Classic Nintendo Games are (NP)-Hard, $\langle http://arxiv.org/abs/1203.1895 \rangle$ (2012).

[Demaine et al., 2002] E. D. Demaine, S. Hohenberger e D. Liben-Nowell,Tetris is Hard, Even to Approximate, $\langle http://arxiv.org/abs/cs/0210020 \rangle$ (2002).

[Dor e Zwick, 1999] D. Dor e U. Zwick, SOKOBAN and other motion planning problems,*Computational Geometry: Theory and Applications - COMGEO, vol. 13, no. 4* (1999) 215–228.

[Dijkstra, 1959] E.W. Dijkstra, A Note on Two Problems in Connexion with Graphs, *Numerische Mathematlk 1*(1959) 269–271.

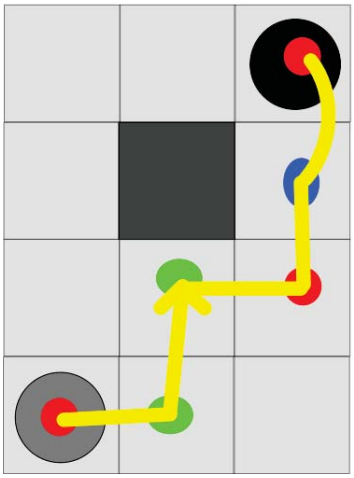


Figura 10: Resolvendo um caso simples