UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE GRADUAÇÃO
DEPARTAMENTO DE ENGENHARIAS E TECNOLOGIA
CURSO BACHARELADO EM INTERDISCIPLINAR EM TECNOLOGIA DA INFORMAÇÃO

ALEXANDER ALVES DE ANDRADE

UTILIZAÇÃO DE UM SISTEMA DE PREVISÃO DE PEÇAS PARA O JOGO TETRIS

PAU DOS FERROS
2026

ALEXANDER ALVES DE ANDRADE

UTILIZAÇÃO DE UM SISTEMA DE PREVISÃO DE PEÇAS PARA O JOGO TETRIS

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de bacharel Interdisciplinar em Tecnologia da Informação.

Orientador: Kennedy Reurison Lopes, Prof. Dr.

PAU DOS FERROS

2026

A553u Andrade, Alexander Alves de.
Utilização de um sistema de previsão de peças para o jogo tetris / Alexander Alves de Andrade. - 2026.
42 f. : il.

Orientador: Kennedy Reurison Lopes.
Monografia (graduação) - Universidade Federal Rural do Semi-árido, Curso de Tecnologia da Informação, 2026.

1. Heurística. 2. Otimização. 3. Lookahead. 4. Tetris. I. Lopes, Kennedy Reurison, orient. II. Título.

Ficha catalográfica elaborada por sistema gerador automático em conformidade com AACR2 e os dados fornecidos pelo) autor(a).
Biblioteca Campus Pau dos Ferros
Bibliotecária: Erlanda Maria Lopes da Silva
CRB: 15/966

O serviço de Geração Automática de Ficha Catalográfica para Trabalhos de Conclusão de Curso (TCC´s) foi desenvolvido pelo Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (USP) e gentilmente cedido para o Sistema de Bibliotecas da Universidade Federal Rural do Semi-Árido (SISBI-UFERSA), sendo customizado pela Superintendência de Tecnologia da Informação e Comunicação (SUTIC) sob orientação dos bibliotecários da instituição para ser adaptado às necessidades dos alunos dos Cursos de Graduação e Programas de Pós-Graduação da Universidade.

ALEXANDER ALVES DE ANDRADE

UTILIZAÇÃO DE UM SISTEMA DE PREVISÃO DE PEÇAS PARA O JOGO TETRIS

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de bacharel Interdisciplinar em Tecnologia da Informação.

Defendida em: 25/06/2026

BANCA EXAMINADORA

Kennedy Reurison Lopes, Prof. Dr. (UFERSA)
Presidente

Rebeka Oliveira Domingues, Prof. (UFRPE)
Membro Examinador

Bruno Borges da Silva, Prof. (UFERSA)
Membro Examinador

AGRADECIMENTOS

Agradeço, primeiramente, a Deus, por ter me concedido força e perseverança ao longo desta jornada, que foi marcada por desafios, aprendizados e superações.

Agradeço também à minha família, pelo apoio constante, pelo incentivo e por estar presente durante todo o caminho que escolhi.

Ao professor e orientador Kennedy Reurison Lopes, expresso minha gratidão pela paciência, dedicação, orientação e contribuições durante o desenvolvimento deste trabalho.

EPÍGRAFE

“O maior perigo da Inteligência Artificial é que as pessoas concluam muito cedo que a entendem.”

(Eliezer Yudkowsky)

RESUMO

Este projeto apresenta o desenvolvimento e avaliação de um agente inteligente para o jogo tetris, com foco na otimização em decisões em tempo real. o tetris foi classificado por pesquisas como um problema NP-completo, ou seja, torna inviável construir um algoritmo capaz de tomar sempre a melhor decisão, todas as vezes, em longos períodos de tempo. Logo, a solução proposta foi combinar heurísticas com técnicas de previsão para simular cenários e chegar o mais próximo de decisões satisfatória, prevendo não apenas onde a peça atual encaixa melhor, mas verificando a próxima peça também. Os resultados demonstraram aumentos bastante significativos, onde a pontuação média sem o lookahead ficou três vezes menor do que utilizando o algoritmo, reduzindo bastante a altura de colunas, buracos criados, evitando um cenário rápido de fim de jogo. Concluindo que a antecipação de estados futuros, é indispensável em problemas de otimização em tempo real.

Palavras-chave: otimização; tetris; heurísticas; lookahead

ABSTRACT

This project presents the development and evaluation of an intelligent agent for the Tetris game, focusing on optimization in real-time decisions. Tetris has been classified by research as an NP-complete problem, meaning it is impossible to build an algorithm capable of always making the best decision, every time, over long periods of time. Therefore, the proposed solution was to combine heuristics with forecasting techniques to simulate scenarios and get as close as possible to satisfactory decisions, predicting not only where the current piece fits best, but also checking the next piece. The results showed quite significant increases, where the average score without lookahead was three times lower than using the algorithm, significantly reducing the height of columns and holes created, avoiding a quick end-of-game scenario.
In conclusion, anticipating future states is indispensable in real-time optimization problems

Keywords: optimization; tetris; heuristics; lookahead

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Ao longo das últimas décadas, os jogos digitais passaram a desempenhar um papel relevante não apenas como forma de entretenimento, mas também como ambientes de experimentação para algoritmos de inteligência artificial. Por possuírem regras bem definidas, objetivos claros e estados mensuráveis, esses jogos permitem testar estratégias de tomada de decisão, busca e otimização em cenários controlados. Nesse contexto, jogos clássicos oferecem uma base adequada para a análise de agentes computacionais, uma vez que combinam mecânicas simples com desafios progressivamente complexos.

Entre esses jogos, destaca-se o Tetris, desenvolvido em 1984 por Alexey Pajitnov. O jogo tornou-se um dos títulos mais conhecidos da história dos videogames e passou a ser estudado em diferentes áreas, como ciência da computação, inteligência artificial e cognição. Sua relevância como objeto de estudo está relacionada ao fato de apresentar regras simples, mas exigir decisões sucessivas que influenciam diretamente o desempenho da partida.

No Tetris, o jogador deve posicionar peças de diferentes formatos em um tabuleiro, buscando completar linhas horizontais. Quando uma linha é preenchida, ela é removida, liberando espaço para novas peças. Caso os blocos acumulados atinjam o topo do tabuleiro, a partida é encerrada. Assim, o desempenho do jogador depende da capacidade de organizar o espaço disponível, evitar a formação de buracos, controlar a altura das colunas e manter o tabuleiro em uma configuração favorável para jogadas futuras.

Essa dinâmica torna o Tetris um problema relevante para estudos de otimização e tomada de decisão em tempo real. Cada peça pode ser posicionada em diferentes rotações e colunas, gerando múltiplas possibilidades de jogada. Além disso, cada decisão altera o estado do tabuleiro e interfere nas alternativas disponíveis posteriormente. Dessa forma, uma jogada aparentemente adequada no momento atual pode comprometer o desempenho nas etapas seguintes da partida.

Diante dessa complexidade, torna-se inviável avaliar todas as sequências possíveis de jogadas em partidas longas. Por esse motivo, estratégias heurísticas são frequentemente utilizadas para orientar a tomada de decisão, permitindo que o agente escolha jogadas satisfatórias em tempo viável. Nesse contexto, técnicas como a avaliação de estados, a simulação de jogadas e o *lookahead* tornam-se relevantes, pois possibilitam ao algoritmo considerar não apenas o efeito imediato de uma peça, mas também consequências futuras.

Este trabalho desenvolve uma versão digital do jogo Tetris e implementa um agente computacional baseado em função heurística. O objetivo é analisar o desempenho do agente em

duas configurações: uma sem o uso de *lookahead*, considerando apenas a peça atual, e outra com *lookahead*, considerando também a próxima peça disponibilizada pelo jogo. Com isso, busca-se verificar se a antecipação de estados futuros contribui para melhorar a pontuação, reduzir buracos, controlar a altura do tabuleiro e prolongar o tempo até o *Game Over*.

A questão central que orienta este trabalho é: de que forma o uso de heurísticas e da técnica de *lookahead* pode contribuir para otimizar a tomada de decisão de um agente computacional no jogo Tetris? A partir dessa questão, o estudo busca contribuir para a compreensão de estratégias de decisão aplicadas a jogos digitais e a problemas de otimização em tempo real.

Embora o estudo seja desenvolvido no contexto de um jogo digital, a lógica de tomada de decisão utilizada no Tetris apresenta relação com problemas reais de planejamento e automação. Em ambientes industriais, por exemplo, máquinas automatizadas precisam escolher posições, encaixes e sequências de montagem de forma eficiente, evitando configurações que comprometam etapas futuras. Situações semelhantes podem ser observadas em linhas de montagem de veículos, sistemas automatizados de armazenamento, braços robóticos e equipamentos voltados à construção automatizada. Nesses casos, assim como no Tetris, a decisão mais adequada não depende apenas do resultado imediato, mas também dos efeitos que essa escolha pode gerar nas próximas etapas do processo.

1.1 Objetivo Geral

Desenvolver uma versão digital do jogo Tetris com um agente computacional baseado em função heurística, avaliando o impacto do uso da técnica de *lookahead* na tomada de decisão e no desempenho do jogo.

1.2 Objetivos Específicos

1. Recriar a mecânica fundamental do jogo Tetris em linguagem de programação adequada.
2. Implementar uma função heurística para avaliar diferentes estados do tabuleiro.
3. Desenvolver um agente capaz de analisar possibilidades de rotação e posicionamento das peças.
4. Comparar o desempenho do agente sem *lookahead* e com *lookahead*.
5. Avaliar os resultados obtidos a partir de métricas como pontuação, linhas eliminadas, tempo até o *Game Over*, altura do tabuleiro e quantidade de buracos.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 NP-completo

O Tetris pode ser compreendido como um problema de elevada complexidade computacional, pois cada jogada modifica o estado do tabuleiro e influencia diretamente as possibilidades futuras. Embora a mecânica do jogo seja simples, baseada no deslocamento e na rotação de peças, a quantidade de combinações possíveis cresce rapidamente à medida que novas peças são inseridas. Assim, para encontrar uma sequência ideal de jogadas, seria necessário avaliar diferentes posições, rotações e estados futuros do tabuleiro, o que torna a busca exaustiva inviável em partidas longas.

Na teoria da computação, problemas classificados como NP-completos são aqueles para os quais não se conhece um algoritmo eficiente capaz de encontrar sempre a melhor solução em tempo viável, especialmente quando o tamanho da entrada aumenta (Hearn; Demaine, 2005). No caso do Tetris, estudos demonstram que determinar uma estratégia ótima para determinadas sequências de peças é um problema NP-completo, o que significa que a análise de todas as possibilidades pode exigir um esforço computacional muito elevado. Dessa forma, a obtenção de uma solução exata torna-se impraticável em cenários dinâmicos e de longa duração. Essa característica justifica o uso de estratégias aproximadas, como heurísticas e técnicas de antecipação de estados futuros. Em vez de buscar a solução perfeita para toda a partida, o agente avalia estados possíveis do tabuleiro com base em critérios definidos, como altura das colunas, número de buracos, linhas eliminadas e irregularidade da superfície. Com isso, é possível selecionar jogadas satisfatórias em tempo viável, ainda que não haja garantia de que sejam sempre as melhores possíveis. No presente trabalho, essa limitação computacional fundamenta a adoção de uma função heurística combinada ao lookahead, permitindo ao agente considerar não apenas a peça atual, mas também os impactos prováveis da próxima peça.

2.2 Jogos digitais como ambiente de teste para inteligência artificial

Os jogos digitais têm sido amplamente utilizados como ambientes de experimentação para algoritmos de inteligência artificial, uma vez que apresentam regras bem definidas, estados mensuráveis e objetivos claros. Nesse contexto, o Tetris se destaca por combinar uma mecânica simples com elevada complexidade estratégica, exigindo decisões sucessivas em tempo real.

Essa condição permite que o jogo seja utilizado como modelo para investigar problemas de busca, previsão e otimização, especialmente quando o agente precisa selecionar a melhor ação possível a partir de um conjunto limitado de informações disponíveis.

Além disso, os jogos digitais possibilitam observar o comportamento de agentes inteligentes em ambientes controlados, nos quais é possível repetir testes, comparar estratégias e medir resultados por meio de indicadores objetivos. Essa possibilidade é relevante para estudos de inteligência artificial, pois permite verificar se determinada técnica melhora ou não o desempenho do agente. No caso do Tetris, métricas como pontuação, número de linhas eliminadas, tempo até o fim da partida e estabilidade do tabuleiro permitem analisar quantitativamente a eficiência das estratégias adotadas.

Embora o estudo esteja inserido no contexto de um jogo digital, a lógica envolvida na tomada de decisão apresenta relação com problemas reais de planejamento e automação. Em diferentes sistemas automatizados, como linhas de montagem, braços robóticos e processos de organização de peças ou materiais, é necessário avaliar alternativas, prever consequências e escolher ações que não comprometam etapas futuras. Assim, o Tetris funciona como um ambiente simplificado para observar estratégias de decisão aplicáveis a problemas mais amplos de otimização em tempo real.

2.3 Técnicas de otimização

Em problemas de elevada complexidade computacional, como os classificados como NP-completos, torna-se necessário recorrer a técnicas de otimização capazes de encontrar soluções satisfatórias em tempo viável. Entre essas estratégias, destacam-se os algoritmos de aproximação, que buscam soluções próximas da ótima quando a solução exata é computacionalmente inviável, e os algoritmos gulosos, que selecionam a alternativa mais promissora em cada etapa do processo decisório.

No entanto, em jogos de longo prazo, como o Tetris, decisões baseadas apenas no melhor resultado imediato podem gerar consequências negativas nas jogadas seguintes. Uma peça posicionada de forma aparentemente vantajosa pode criar buracos, aumentar a irregularidade do tabuleiro ou comprometer encaixes futuros. Por esse motivo, a otimização no Tetris não deve considerar apenas a pontuação instantânea, mas também a preservação de estados favoráveis para a continuidade da partida.

Métodos como minimax e poda também são relevantes no campo da otimização e da

tomada de decisão em jogos. O minimax é comumente utilizado em jogos de dois jogadores, como xadrez, damas e jogo da velha, pois alterna entre a maximização dos ganhos de um jogador e a minimização dos ganhos do oponente (Herrmann, 1998). A poda, por sua vez, reduz o espaço de busca ao eliminar ramos que não precisam ser explorados, diminuindo o custo computacional da análise.

No caso deste trabalho, foram consideradas estratégias heurísticas inspiradas em estudos sobre Tetris e metodologias de busca (Lundgaard; McKee, 1998; Burke; Kendall, 2014). A proposta consiste em minimizar fatores negativos, como altura excessiva, formação de buracos e irregularidade entre colunas, ao mesmo tempo em que se busca favorecer a eliminação de linhas. Dessa forma, o algoritmo seleciona o estado considerado mais promissor de acordo com os critérios definidos na função de avaliação.

É fundamental definir corretamente os critérios heurísticos, pois eles influenciam diretamente o desempenho do agente. Ajustando adequadamente os pesos e prioridades, como manter a altura baixa, evitar buracos e favorecer a remoção de linhas, é possível tornar o comportamento do agente mais consistente e eficiente durante a partida.

2.4 Função heurística e avaliação de estados

No Tetris, a tomada de decisão do agente está sujeita ao problema da explosão combinatória. A cada nova peça, o algoritmo precisa considerar diferentes rotações e posições horizontais possíveis. Além disso, quando se considera a próxima peça por meio do *lookahead*, o número de combinações cresce ainda mais, pois cada jogada atual passa a gerar novos estados futuros a serem avaliados. Dessa forma, torna-se inviável analisar todas as sequências possíveis de movimentos ao longo de uma partida completa. Esse cenário justifica o uso de uma função heurística, capaz de estimar a qualidade de cada estado do tabuleiro e orientar a escolha da jogada mais promissora em tempo computacional viável.

Em problemas nos quais a busca exaustiva é inviável, a função heurística assume papel fundamental na orientação das decisões do agente. No caso do Tetris, cada possível posicionamento de uma peça gera um novo estado do tabuleiro, que precisa ser avaliado segundo critérios previamente definidos. Assim, a função heurística atribui um valor de qualidade ou custo para cada jogada, permitindo que o algoritmo escolha a alternativa mais promissora sem precisar analisar todas as sequências futuras possíveis.

Entre os critérios mais utilizados para esse tipo de avaliação estão a altura máxima das

colunas, a quantidade de buracos formados, a irregularidade entre colunas adjacentes e o número de linhas eliminadas. Esses elementos representam diretamente a estabilidade do tabuleiro, pois estados com muitos espaços vazios ou colunas muito altas tendem a reduzir as possibilidades de encaixe nas próximas jogadas. Dessa forma, a função heurística não busca apenas aumentar a pontuação imediata, mas também preservar condições favoráveis para a continuidade da partida.

A função heurística utilizada neste trabalho é expressa da seguinte forma:

$$f(s) = w_h \cdot H(s) + w_b \cdot B(s) + w_c \cdot C(s) - w_l \cdot L(s)$$

Em que (f(s)) representa o custo total do estado avaliado; (H(s)) corresponde à altura máxima do tabuleiro após o movimento; (B(s)) representa a quantidade de buracos formados; (C(s)) corresponde à rugosidade do tabuleiro, isto é, à diferença entre as alturas de colunas adjacentes; e (L(s)) representa o número de linhas eliminadas. Os termos (w h), (w b), (w c) e (w l) correspondem aos pesos atribuídos a cada critério.

Nessa função, os fatores altura, buracos e rugosidade aumentam o custo da jogada, pois indicam estados menos favoráveis para a continuidade da partida. Por outro lado, a eliminação de linhas reduz o custo, uma vez que representa um resultado positivo para o agente, liberando espaço no tabuleiro e contribuindo para o aumento da pontuação. Dessa forma, quanto menor for o valor de (f(s)), melhor será considerado o estado resultante da jogada.

A atribuição dos pesos define a prioridade do agente durante o processo de escolha. A eliminação de linhas recebeu o maior peso, por estar diretamente relacionada ao objetivo principal do jogo. A quantidade de buracos também recebeu peso elevado, pois buracos comprometem jogadas futuras e dificultam a limpeza de linhas. Já a altura e a rugosidade receberam pesos menores, mas ainda relevantes, pois indicam riscos progressivos relacionados ao acúmulo de peças e à dificuldade de encaixe.

2.5 Tomada de decisão em tempo real

A tomada de decisão em tempo real ocorre quando um sistema precisa selecionar uma ação adequada dentro de um intervalo reduzido de tempo. Em jogos como o Tetris, essa exigência é ainda mais evidente, pois a peça em movimento continua descendo enquanto o agente avalia as possibilidades disponíveis. Assim, a qualidade da decisão não depende apenas da precisão do algoritmo, mas também da sua capacidade de responder rapidamente às mudanças do ambiente.

No contexto do Tetris, cada nova peça exige que o agente avalie possibilidades de rotação, deslocamento e encaixe antes que o jogo avance para uma condição desfavorável. Essa característica torna inviável a análise completa de todas as sequências futuras, pois o tempo disponível para decisão é limitado. Dessa forma, o agente precisa equilibrar a qualidade da escolha com o custo computacional necessário para avaliá-la.

Nesse sentido, estratégias baseadas em heurísticas tornam-se adequadas porque reduzem o espaço de busca e permitem encontrar soluções satisfatórias em menor tempo. Embora não garantam a solução ótima, essas estratégias possibilitam que o agente atue de forma eficiente em cenários dinâmicos, nos quais esperar por uma análise completa de todas as possibilidades poderia tornar a decisão inviável.

A utilização de técnicas de previsão contribuem para esse processo ao permitir que o agente antecipe parte das consequências futuras sem expandir todo o espaço de possibilidades. Assim, a decisão passa a considerar não apenas o estado imediato do tabuleiro, mas também o impacto provável da próxima peça. Essa antecipação melhora a qualidade das escolhas sem eliminar a necessidade de respostas rápidas, mantendo o equilíbrio entre desempenho e viabilidade computacional.

2.6 Lookahead

Uma das melhorias propostas neste trabalho é a incorporação da previsão da próxima peça, o que transforma a decisão em um problema de busca em horizonte dupla, levando em consideração a peça atual juntamente com a próxima peça (lookahead)(Zhang *et al.*, 2019), o que é bastante relevante no tetris, pois, o próprio jogo fornece qual será a próxima peça, melhorando bastante os resultados, evitando situações que pareçam ser boas a curto prazo e gerem cenários ruim mais pra frente. Para aplicar essa previsão, o agente deve simular não apenas todas as possibilidades de encaixe e rotação da peça atual, mas também todas as combinações com a próxima peça. Dessa forma, a escolha final não se baseia apenas no estado resultante, mas no impacto conjunto de duas ações consecutivas.O objetivo é identificar o estado do tabuleiro que maximiza o potencial de estabilidade futura, minimiza custos heurísticos acumulados e prolonga a sobrevivência no jogo.

Um exemplo muito prático, é imaginar um quebra-cabeça normal e antes de colocar qualquer peça na mesa, ele precisa copiar todo o tabuleiro e testar essa peça em todas as posições, avaliando onde essa peça deve encaixar perfeitamente, é exatamente isso que o lookahead faz,

em vez de avaliar apenas o impacto imediato da peça atual, o algoritmo simula as possíveis jogadas da próxima peça prevista, buscando a combinação de movimentos que resulte no menor custo, com as métricas que foram definidas, após jogadas seguidas.

Logo, para isso, temos que utilizar algoritmos de deepcopy, que permitem duplicar a estrutura interna do estado atual, preservando referências e valores, testando virtualmente onde a peça pode encaixar, sem alterar o tabuleiro original e após o cálculo de pesos, decide a jogada mais aproximada a uma ótima decisão, onde adia ainda mais o fim de jogo.

2.7 Espaço de estados e simulação de jogadas

O espaço de estados corresponde ao conjunto de configurações possíveis que um sistema pode assumir durante sua execução. No Tetris, cada estado é representado pela disposição das peças no tabuleiro, pela peça atual, pela próxima peça e pelas possibilidades de rotação e deslocamento. A cada nova jogada, o número de estados possíveis aumenta, tornando inviável a análise completa de todas as sequências futuras.

No contexto do jogo, cada combinação entre rotação e posição horizontal de uma peça gera um estado diferente do tabuleiro. Assim, uma mesma peça pode produzir resultados variados, dependendo do local em que é posicionada. Algumas jogadas podem reduzir a altura das colunas e eliminar linhas, enquanto outras podem formar buracos, aumentar a irregularidade da superfície ou dificultar encaixes futuros. Por isso, a avaliação dos estados é essencial para orientar a escolha da jogada mais adequada.

Por esse motivo, a simulação de jogadas passa a ser uma estratégia importante. Ao gerar estados temporários do tabuleiro, o agente consegue prever as consequências de diferentes posicionamentos antes de alterar o jogo real. Essa simulação permite comparar alternativas e selecionar aquela que apresenta menor custo segundo a função heurística adotada. Assim, o processo de decisão deixa de ser puramente imediato e passa a considerar os efeitos prováveis de cada movimento.

Neste trabalho, a simulação dos estados é realizada por meio de cópias temporárias do tabuleiro, permitindo que o agente teste diferentes posições e rotações sem modificar a partida em andamento. Essa lógica também possibilita a aplicação do *lookahead*, pois o algoritmo consegue avaliar não apenas o estado gerado pela peça atual, mas também os possíveis estados resultantes da próxima peça. Dessa forma, a análise do espaço de estados contribui diretamente para uma tomada de decisão mais segura e eficiente.

2.8 Deepcopy

Os autores (Grogono; Sakkinen, 1999) argumentam que a simples atribuição ou cópia rasa (shallow copy) não mantém a integridade de objetos compostos, nesse tipo de cópia, estruturas internas continuam sendo compartilhadas entre o objeto original e sua réplica, de modo que qualquer alteração feita durante uma simulação refletirá também no tabuleiro real, podendo causar até o fim de jogo e o agente perderia a capacidade de prever consequências, esse problema fundamenta a necessidade do uso de cópias profundas (deepcopy) em estruturas complexas. Para cada jogada, um tabuleiro temporário é gerado,fazendo cópias da peça e do tabuleiro para manter o estado isolado, então, o algoritmo testa todas as rotações das peças e todas as posições, escolhendo assim a jogada com o menor peso, seguindo os critérios definidos, dessa forma, a união entre a heurística e o deepcopy possiblita uma tomada e decisão robusta e execute a jogada mais vantajosa possível. Especialmente quando múltiplos estados precisam ser avaliados de independentemente. Dessa forma, os autores reforçam que preservar a autonomia entre o tabuleiro simulado e o tabuleiro real é essencial para que o agente mantenha sua capacidade de prever consequências

De acordo com (Cook, 1992), a cópia profunda envolve a replica recursiva de todas as estruturas internas de um objeto, garantindo a total independência entre a cópia e o original. Na linguagem Python, esse mecanismo é realizado pela função deepcopy, que é bastante usada em algoritmos que necessitam de simulações em ambientes separados, como inteligência artificial, jogos e busca heurística, ou seja, a cópia profunda é extremamente necessária para o uso do lookahead, no geral, é o que permite o agente testar, comparar, prever e decidir.

O deepcopy atua como o mecanismo que possibilita a existência do lookahead, ao passo que o lookahead é o responsável por antecipar erros e garantir um desempenho superior no ambiente altamente dinâmico e imprevisível do Tetris.

3 TRABALHOS RELACIONADOS

Este capítulo apresenta trabalhos que contribuem para a compreensão da complexidade computacional do Tetris e para a definição das estratégias adotadas no desenvolvimento do agente proposto. Foram considerados estudos relacionados à dificuldade do jogo como problema de otimização, ao uso de heurísticas, à antecipação de estados futuros e às metodologias de busca aplicadas à tomada de decisão.

3.1 Tetris é difícil, mesmo para se aproximar

(Demaine; Hohenberger; Liben-Nowell, 2003) demonstram a elevada complexidade computacional do Tetris, indicando que a obtenção de soluções ótimas para determinadas formulações do jogo exige a análise de um grande número de possibilidades. Essa característica reforça a necessidade de estratégias aproximadas, uma vez que a busca exaustiva por todas as sequências possíveis de peças e movimentos torna-se inviável em partidas longas.

O estudo contribui diretamente para a fundamentação deste trabalho, pois evidencia que o Tetris não deve ser tratado apenas como um jogo de regras simples, mas como um problema combinatório complexo. Cada peça posicionada altera o estado do tabuleiro e influencia as jogadas seguintes, fazendo com que a decisão ideal dependa não apenas do movimento atual, mas também das consequências futuras.

O presente trabalho aproxima-se dessa discussão ao propor uma solução baseada em heurísticas, *lookahead* e simulação de estados. Em vez de buscar uma solução ótima para toda a sequência de jogadas, o agente desenvolvido avalia diferentes posições e rotações possíveis, atribuindo um custo heurístico a cada estado resultante. Dessa forma, busca-se uma solução satisfatória em tempo viável, compatível com as limitações de um ambiente dinâmico como o Tetris.

A principal diferença em relação ao estudo de Demaine, Hohenberger e Liben-Nowell (Demaine; Hohenberger; Liben-Nowell, 2003) está no enfoque adotado. Enquanto os autores analisam a complexidade teórica do Tetris, este trabalho concentra-se na implementação prática de um agente capaz de tomar decisões automáticas durante a execução do jogo. Assim, a contribuição do estudo teórico é utilizada como justificativa para a adoção de métodos aproximados.

3.2 Computador com IA joga Tetris

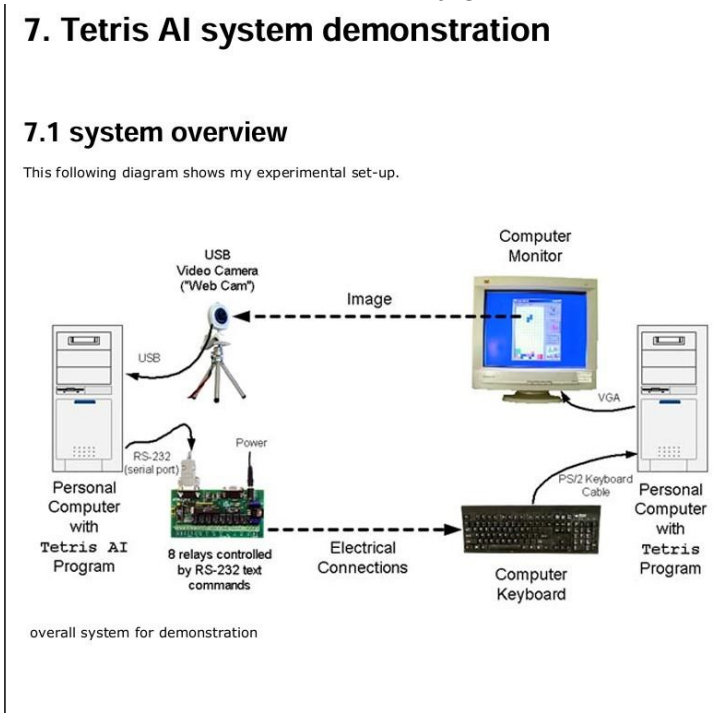
O trabalho de (Fahey; Dellacherie, 2003) apresenta uma das abordagens clássicas de inteligência artificial aplicada ao Tetris. O estudo utiliza métricas relacionadas ao estado do tabuleiro, como linhas removidas, buracos criados e diferenças de altura, para orientar a tomada de decisão do agente. Esses critérios se aproximam dos utilizados no presente trabalho, especialmente no que se refere à avaliação da estabilidade do tabuleiro.

Um dos pontos mais relevantes desse estudo é a importância atribuída ao conhecimento da próxima peça. (Fahey; Dellacherie, 2003) indicam que conhecer antecipadamente a peça seguinte pode melhorar significativamente o desempenho de um algoritmo jogador de Tetris. Essa ideia se relaciona diretamente ao uso do *lookahead*, pois permite que o agente não avalie apenas o encaixe imediato da peça atual, mas também os efeitos prováveis da próxima jogada.

A principal diferença entre o trabalho de (Fahey; Dellacherie, 2003) e a metodologia adotada nesta pesquisa está nos recursos computacionais utilizados. Enquanto no estudo clássico o agente dependia da reconstrução do tabuleiro a partir de imagens capturadas da tela, neste trabalho o estado do jogo é manipulado diretamente por meio de estruturas internas de dados. Isso permite realizar simulações de forma mais direta, utilizando cópias temporárias do tabuleiro para testar diferentes possibilidades de encaixe antes da execução da jogada definitiva.

Dessa forma, o trabalho (Fahey; Dellacherie, 2003) contribui para este estudo ao demonstrar a importância de métricas heurísticas e da previsão da próxima peça. O presente trabalho retoma essa lógica em um contexto de implementação mais simples e direto, utilizando a simulação computacional dos estados do tabuleiro como base para a tomada de decisão.

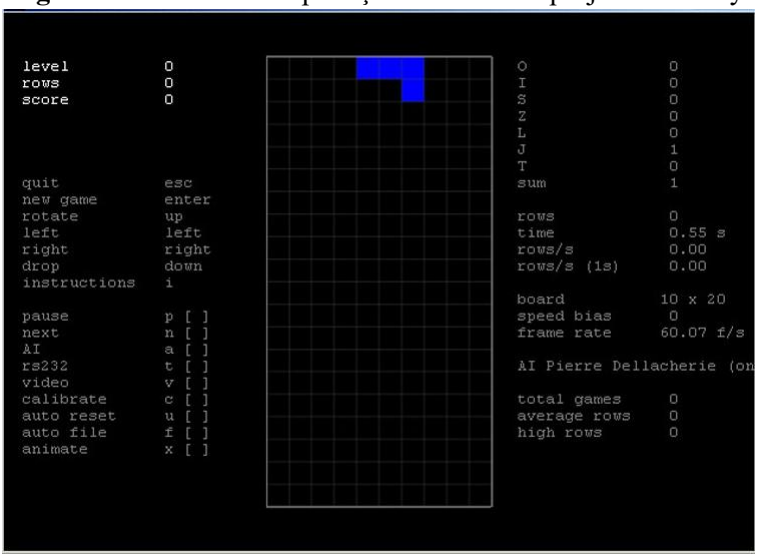
Figura 1 – Sistema de captura utilizado para identificar o estado do jogo



Fonte: Adaptado de fahey (2008).

A Figura 1 ilustra o sistema utilizado por (Fahey, 2008) para capturar o estado do jogo. Devido às limitações tecnológicas da época, a leitura do tabuleiro dependia de uma câmera externa, tornando o processo mais sensível a fatores como iluminação, qualidade da imagem e precisão da captura.

Figura 2 – Interface da aplicação utilizada no projeto de Fahey



Fonte: Adaptado de fahey (2008).

A Figura 2 apresenta a interface da aplicação utilizada no estudo de (Fahey, 2008). A comparação com a abordagem adotada neste trabalho evidencia a diferença entre um sistema baseado em captura visual externa e uma implementação que manipula diretamente o estado lógico do tabuleiro.

3.3 Uma estratégia de antecipação para o planejamento de buscas heurísticas

(Vidal, 2004) discute o uso de *lookahead* no contexto do planejamento por busca heurística. Embora o estudo não seja especificamente voltado ao Tetris, sua contribuição é relevante por demonstrar como a antecipação de estados futuros pode melhorar a tomada de decisão em problemas complexos. A técnica permite avaliar não apenas o estado atual, mas também possíveis desdobramentos, reduzindo decisões imediatistas que podem gerar resultados desfavoráveis posteriormente.

Essa lógica se relaciona diretamente ao presente trabalho, pois o agente com *lookahead* avalia a peça atual em conjunto com a próxima peça disponibilizada pelo jogo. Dessa forma, a escolha da jogada passa a considerar o impacto provável de duas ações consecutivas, o que contribui para reduzir buracos, controlar a altura do tabuleiro e manter uma superfície mais regular.

O estudo de (Vidal, 2004) também contribui ao discutir a relação entre heurística e expansão de estados. Em estratégias de busca, estados considerados mais promissores são priorizados de acordo com uma função de avaliação. No Tetris, lógica semelhante é aplicada quando o agente simula diferentes encaixes e seleciona aquele que apresenta menor custo heurístico.

Assim, embora o trabalho de (Vidal, 2004) esteja inserido em outro contexto de planejamento, ele fornece uma base conceitual importante para justificar o uso do *lookahead* como técnica de antecipação em ambientes dinâmicos e altamente combinatórios.

3.4 Metodologias de busca: otimização e técnicas de apoio à decisão

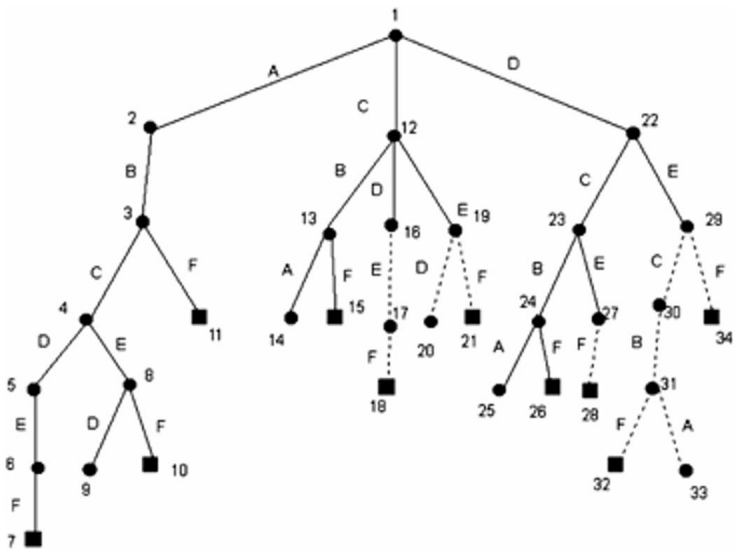
(Burke; Kendall, 2014) apresentam metodologias de busca aplicadas a problemas de otimização e apoio à decisão. Entre as estratégias discutidas, destacam-se técnicas como programação dinâmica e *branch-and-bound*, que buscam reduzir o espaço de busca por meio da decomposição do problema em subproblemas e da eliminação de alternativas menos promissoras.

Essas estratégias são relevantes para compreender a lógica geral da otimização em problemas complexos. A programação dinâmica evita a repetição de cálculos ao armazenar resultados intermediários, enquanto o *branch-and-bound* reduz o espaço de busca ao descartar caminhos que não podem produzir soluções melhores. No entanto, em jogos como o Tetris, a aleatoriedade das peças, a profundidade do espaço de estados e a necessidade de decisões em tempo real tornam difícil a aplicação direta de métodos exaustivos.

Segundo (Burke; Kendall, 2014), a combinação de heurísticas distintas pode contribuir para uma busca mais equilibrada e menos dependente de uma única métrica. Essa ideia é incorporada no presente trabalho por meio de uma função de avaliação composta por diferentes critérios, como altura do tabuleiro, quantidade de buracos, rugosidade entre colunas e linhas eliminadas. Dessa forma, o agente não considera apenas a pontuação imediata, mas também a estabilidade futura do tabuleiro.

A contribuição desse trabalho para a presente pesquisa está, portanto, na fundamentação das estratégias de busca e avaliação de estados. Embora o objetivo aqui não seja implementar diretamente programação dinâmica ou *branch-and-bound*, esses métodos ajudam a contextualizar a importância de reduzir o espaço de busca e orientar a tomada de decisão por meio de critérios bem definidos.

Figura 3 – Exemplo de árvore de busca associada ao método *branch-and-bound*.



Fonte: Burke & Kendall (2014).

A Figura 3 representa uma árvore de busca associada ao método *branch-and-bound*. A imagem evidencia a divisão do problema em alternativas menores e a eliminação de caminhos

que não apresentam potencial para alcançar uma solução mais adequada.

De modo geral, os trabalhos analisados demonstram que o Tetris constitui um problema de elevada complexidade computacional, exigindo estratégias aproximadas para a tomada de decisão. Os estudos sobre complexidade justificam a inviabilidade da busca exaustiva, enquanto os trabalhos sobre heurísticas, *lookahead* e metodologias de busca contribuem para a definição da abordagem adotada nesta pesquisa. Assim, o presente trabalho se apoia nessas contribuições para desenvolver e comparar agentes heurísticos com e sem antecipação da próxima peça.

4 METODOLOGIA

A pesquisa foi desenvolvida a partir de uma abordagem aplicada, experimental e quantitativa, tendo como finalidade construir e avaliar um agente computacional capaz de tomar decisões automáticas no jogo Tetris. O caráter aplicado se justifica pelo desenvolvimento prático de uma versão digital do jogo e pela implementação de algoritmos voltados à escolha automática das jogadas. A abordagem experimental ocorre pela realização de testes comparativos entre duas estratégias de decisão, enquanto a dimensão quantitativa está relacionada à análise de métricas como pontuação final, número de linhas eliminadas, tempo até o *Game Over*, altura média do tabuleiro e quantidade de buracos formados.

Inicialmente, foi realizada uma revisão da literatura sobre jogos digitais, inteligência artificial aplicada a jogos, heurísticas, técnicas de otimização, *lookahead* e simulação de estados. Essa etapa teve como objetivo estabelecer a base teórica necessária para a definição da estratégia adotada, além de auxiliar na escolha dos critérios utilizados para avaliar o desempenho do agente no contexto do Tetris.

Após a revisão teórica, foi desenvolvida uma versão base do jogo Tetris, contendo os principais elementos de sua mecânica, como geração das peças, movimentação lateral, rotação, queda automática, detecção de colisões, eliminação de linhas completas, atualização da pontuação e condição de fim de jogo. O desenvolvimento foi realizado em linguagem Python, por sua facilidade de manipulação de estruturas de dados e por sua utilização recorrente em projetos de inteligência artificial e simulação computacional.

Com o jogo base implementado, foram desenvolvidas duas versões do agente. A primeira corresponde a um agente heurístico sem *lookahead*, que avalia apenas os efeitos imediatos da peça atual sobre o tabuleiro. A segunda corresponde a um agente heurístico com *lookahead*, que considera, além da peça atual, a próxima peça disponibilizada pelo jogo. Dessa forma, tornou-se possível comparar o desempenho de uma estratégia baseada na decisão imediata com outra baseada na antecipação de estados futuros.

O funcionamento do agente foi organizado a partir de três etapas principais. A primeira consiste na geração das ações possíveis, em que o algoritmo identifica as rotações e posições horizontais válidas para a peça em movimento. A segunda etapa corresponde à simulação dos estados resultantes, na qual cada possibilidade de jogada é testada em uma cópia temporária do tabuleiro, sem alterar o estado real do jogo. Por fim, na terceira etapa, cada estado simulado é avaliado por meio de uma função heurística, responsável por atribuir um valor de custo à jogada.

O jogo é composto por sete peças distintas, chamadas tetraminós, que podem ser posicionadas em diferentes rotações e colunas, gerando múltiplas possibilidades de jogada a cada rodada.

Na implementação desenvolvida, cada tetraminó foi representado por meio de matrizes, nas quais as células ocupadas indicam a forma da peça em determinada orientação. Essa estrutura permite que o agente percorra as rotações disponíveis de cada peça e teste seus possíveis posicionamentos no tabuleiro. Assim, a representação das peças está diretamente relacionada à geração das ações possíveis avaliadas pela função heurística.

Figura 4 – Demonstrando como as peças foram adicionadas

Fonte: Autoria própria (2026).

A existência de diferentes rotações para cada tetraminó amplia o espaço de busca do agente computacional. Para cada peça recebida, o algoritmo precisa considerar suas orientações possíveis e os locais válidos de posicionamento no tabuleiro. Cada combinação entre rotação e coluna gera um novo estado do jogo, que pode ser avaliado pela função heurística. Assim, a representação matricial das peças é fundamental para permitir a simulação das jogadas e a escolha da ação mais adequada de acordo com critérios como linhas completadas, altura do tabuleiro, buracos formados e irregularidade da superfície.

A função heurística adotada foi definida com base em critérios recorrentes em agentes aplicados ao Tetris, considerando características diretamente relacionadas à estabilidade do tabu-

Tabela 1 – Tetraminós utilizados no jogo Tetris

Peça	Rotações no código	Característica no jogo
I	2	Peça reta, útil para completar linhas longas e reduzir espaços verticais.
O	1	Peça quadrada, estável e simples de encaixar, pois não muda com rotação.
T	4	Peça versátil, capaz de preencher espaços centrais e encaixes irregulares.
S	2	Peça em zigue-zague, útil em encaixes específicos, mas pode gerar buracos se mal posicionada.
Z	2	Semelhante à peça S, porém invertida, exigindo cuidado na escolha da coluna.
J	4	Peça com formato semelhante a um "L" invertido, útil para preencher laterais e desníveis.
L	4	Peça em formato de "L", complementar à peça J, usada para encaixes laterais e cavidades.

Fonte: Autoria própria (2026).

leiro e à continuidade da partida. Foram avaliados fatores como altura do tabuleiro, quantidade de buracos, rugosidade entre colunas adjacentes e número de linhas eliminadas. A função pode ser expressa da seguinte forma:

$$f(s) = w_h \cdot H(s) + w_b \cdot B(s) + w_c \cdot C(s) - w_l \cdot L(s)$$

Em que:

- $f(s)$ representa o custo total do estado avaliado;
- $H(s)$ corresponde à altura máxima do tabuleiro após o movimento;
- $B(s)$ representa a quantidade de buracos formados;
- $C(s)$ corresponde à rugosidade do tabuleiro, isto é, à diferença entre alturas de colunas adjacentes;
- $L(s)$ representa o número de linhas eliminadas;
- w_h, w_b, w_c e w_l representam os pesos atribuídos a cada critério.

Nessa função, os fatores altura, buracos e rugosidade aumentam o custo da jogada, pois indicam estados menos favoráveis para a continuidade da partida. Por outro lado, a eliminação de linhas reduz o custo, uma vez que representa um resultado positivo para o agente. Assim, quanto menor for o valor de $f(s)$, melhor será considerado o estado resultante da jogada.

Os pesos utilizados nos testes foram definidos da seguinte forma:

- $w_l = 1000.0$, atribuído à eliminação de linhas;

- w_h = 4.0, atribuído à altura do tabuleiro;
- w_b = 100.0, atribuído à quantidade de buracos;
- w_c = 4.0, atribuído à rugosidade entre colunas.

A definição da estratégia do agente foi fundamentada nos trabalhos de (Ackerman, 2018) e (Demaine; Hohenberger; Liben-Nowell, 2003). Esses estudos contribuiram para justificar a utilização de métodos heurísticos, uma vez que a busca por uma solução ótima no Tetris apresenta elevada complexidade computacional. Já o trabalho de (Ackerman, 2018) contribui para a adoção de critérios de avaliação relacionados à estabilidade do tabuleiro.

A definição desses pesos buscou priorizar a eliminação de linhas e penalizar a formação de buracos, uma vez que esses espaços vazios dificultam encaixes posteriores e reduzem a estabilidade do tabuleiro. A altura agregada e a rugosidade também foram consideradas, pois o acúmulo de blocos e a diferença excessiva entre colunas aumentam o risco de *Game Over*, especialmente em estágios mais avançados da partida. Dessa forma, a função heurística busca equilibrar ganho imediato, estabilidade do tabuleiro e preservação de possibilidades para jogadas futuras.

No agente sem *lookahead*, o algoritmo avalia todas as posições e rotações possíveis da peça atual, calcula o custo do estado resultante e seleciona a jogada com menor valor heurístico. Já no agente com *lookahead*, o processo é ampliado: para cada possibilidade da peça atual, o algoritmo gera um estado temporário do tabuleiro e, a partir dele, simula as possibilidades de encaixe da próxima peça. Em seguida, o custo dos estados resultantes é avaliado, permitindo escolher a jogada atual que apresenta melhor consequência provável na jogada seguinte.

Todas as execuções foram realizadas de forma automática, sem intervenção do usuário durante as partidas. As duas versões do agente foram submetidas às mesmas regras do jogo, mantendo-se a mesma estrutura do tabuleiro, os mesmos critérios de pontuação e as mesmas condições de encerramento da partida.

Para avaliar o desempenho dos agentes, foram realizadas 20 execuções independentes para cada versão, totalizando 40 testes. Foram escolhidos 20 testes, pois, assim evitaremos o fator "sorte"e conseguiremos observar essa diferença de média, minima e máxima. Ao final de cada partida, foram registrados os resultados obtidos em relação à pontuação final, ao número de linhas eliminadas, ao tempo até o *Game Over*, à altura máxima média do tabuleiro e à quantidade de buracos formados. Esses critérios foram escolhidos por representarem diretamente a eficiência do agente, a estabilidade do tabuleiro e sua capacidade de prolongar a partida.

Por fim, os dados obtidos foram organizados e comparados entre as duas versões do agente. Essa comparação permitiu avaliar o impacto do uso do *lookahead* na tomada de decisão, verificando se a antecipação da próxima peça contribuiu para aumentar a pontuação, eliminar mais linhas, reduzir a quantidade de buracos, manter o tabuleiro mais estável e prolongar o tempo de jogo.

5 RESULTADOS

Tabela 2 – Resultados obtidos pelo agente heurístico otimizado sem *lookahead*

Métrica	Média	Mínimo	Máximo
Pontuação final	8.260	6.420	29.420
Linhas eliminadas	215	174	289
Tempo até o *Game Over* (s)	455	372	593
Altura máxima média	12	10	15
Número de buracos	14	9	22

Fonte: Autoria própria (2026).

este agente não utiliza lookahead de próxima peça, o código avalia somente as consequências imediatas da peça atual, onde os pontos fortes e fragilizades foram :

- O agente é rápido e responsivo.
- Evita buracos com boa eficiência.
- Eliminação de linhas é bem priorizada.
- O agente não consegue prever situações de risco por não prever a próxima peça.
- A diferença de altura varia muito, consequentemente facilitando ao fim de jogo.

Tabela 3 – Resultados obtidos pelo agente heurístico com *lookahead*

Métrica	Média	Mínimo	Máximo
Pontuação final	26.120	11.920	58.860
Linhas eliminadas	283	250	390
Tempo até o *Game Over* (s)	750	600	950
Altura máxima média	10	8	13
Número de buracos	9	6	16

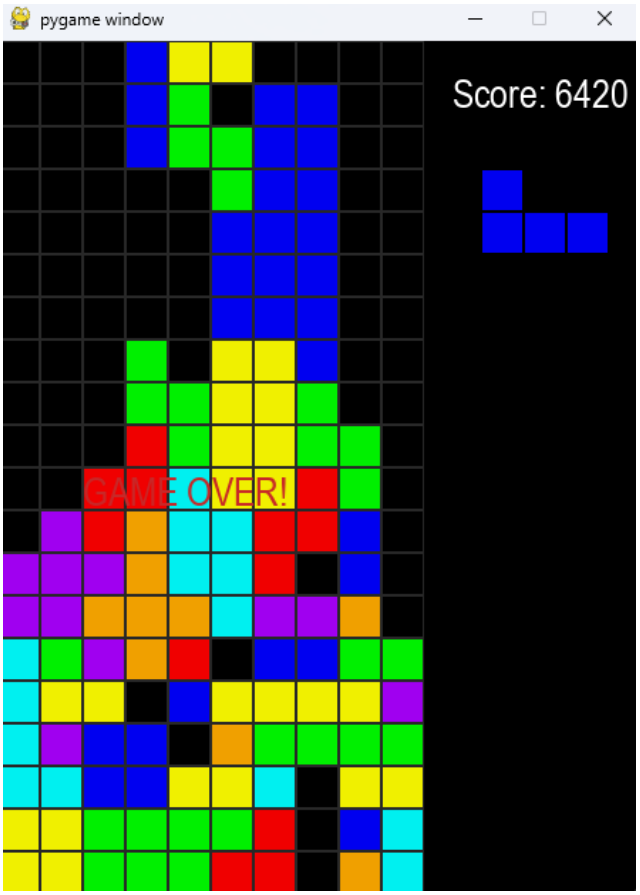
Fonte: Autoria própria (2026).

Com lookahead, o agente consegue planejar jogadas futuras, aumentando a pontuação final média em mais de três vezes em comparação ao agente sem lookahead. O número de linhas eliminadas e o tempo até o fim de jogo também aumentaram bastante, enquanto a altura máxima média e o número de buracos diminuíram. A heuristica com o lookahead mostra ganhos em todas os pontos avaliados, comprovando que a antecipação de jogadas futuras é uma estratégia bastante eficaz para agentes em tetris,

O algoritmo sem o lookahead se mostra fragilizada quanto mais se avança no jogo, a avaliação somente da peça imediata, causa muitos problemas a longo prazo, ao contrário desse

cenário, utilizando o lookahead, ele prevê com mais precisão, uma jogada que não prejudique tanto o tabuleiro, aumentando o tempo até o fim de jogo e maximizando pontos.

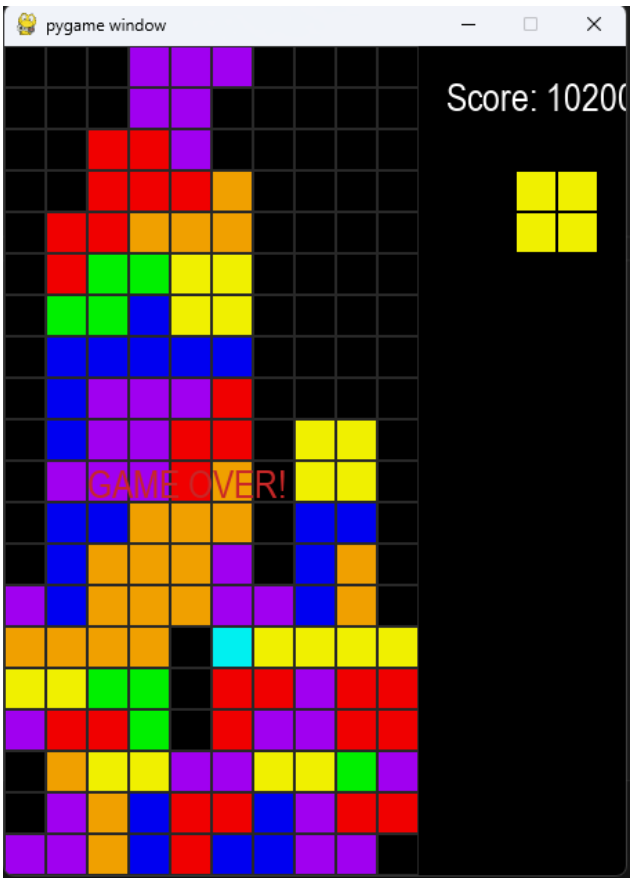
Figura 5 – Menor resultado obtido pelo agente sem *lookahead*.



Fonte: Autoria própria (2026).

Na Figura 5, observa-se uma configuração irregular do tabuleiro, com presença de buracos e desníveis acentuados entre as colunas. Esse comportamento evidencia a limitação da estratégia sem *lookahead*, que avalia apenas o impacto imediato da peça atual.

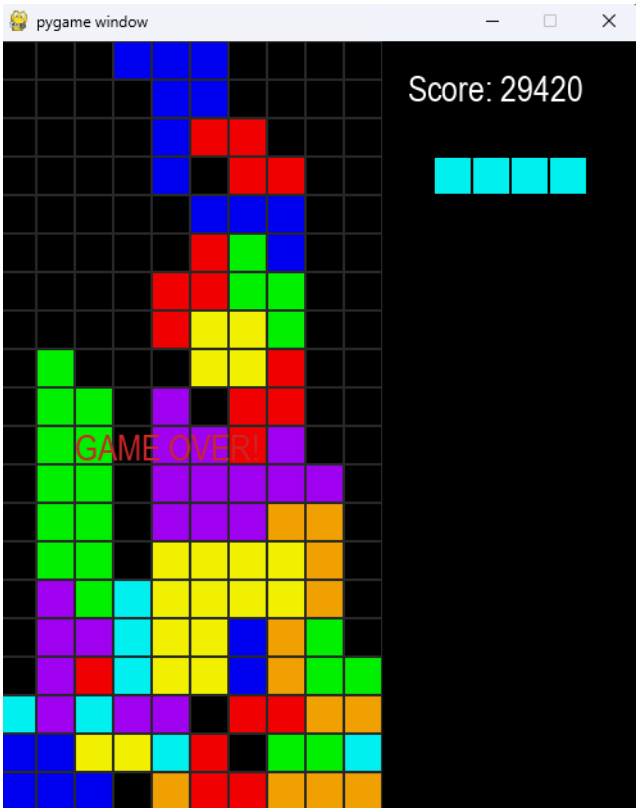
Figura 6 – Resultado médio obtido pelo agente sem *lookahead*.



Fonte: Autoria própria (2026).

A Figura 6 representa uma situação intermediária do agente sem *lookahead*, na qual ainda são perceptíveis buracos e irregularidades na superfície do tabuleiro. Esse resultado indica a dificuldade do agente em manter uma configuração estável ao longo da partida quando considera apenas a peça atual.

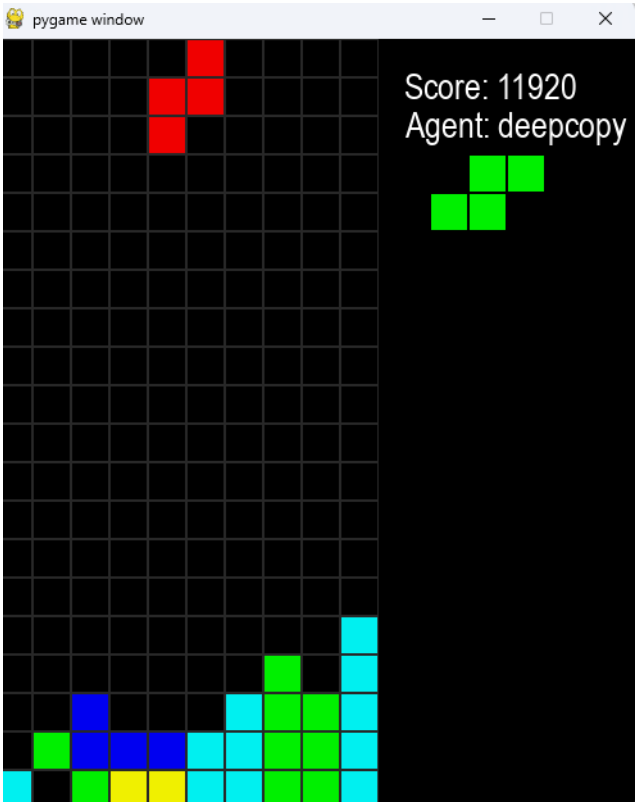
Figura 7 – Melhor resultado obtido pelo agente sem *lookahead*.



Fonte: Autoria própria (2026).

Na Figura 7, apesar de corresponder ao melhor desempenho dessa configuração, ainda é possível observar certa desorganização no tabuleiro. Esse resultado indica que o agente sem *lookahead* pode alcançar boas pontuações, mas tende a depender mais da sequência de peças geradas durante a partida.

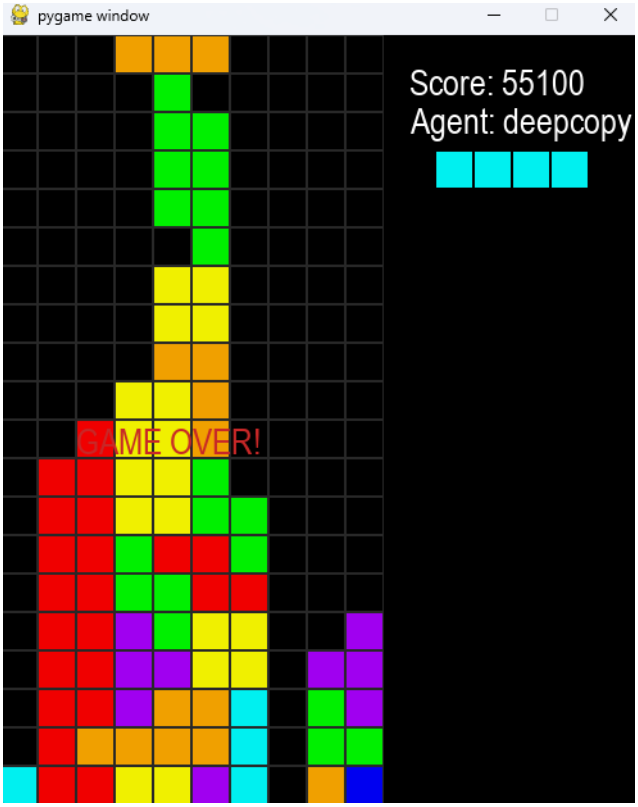
Figura 8 – Resultado obtido pelo agente com *lookahead*.



Fonte: Autoria própria (2026).

Com o uso do *lookahead*, conforme apresentado na Figura 8, observa-se maior controle da altura das colunas e menor formação de buracos. A antecipação da próxima peça contribui para decisões mais estáveis e reduz a ocorrência de jogadas prejudiciais nas etapas seguintes.

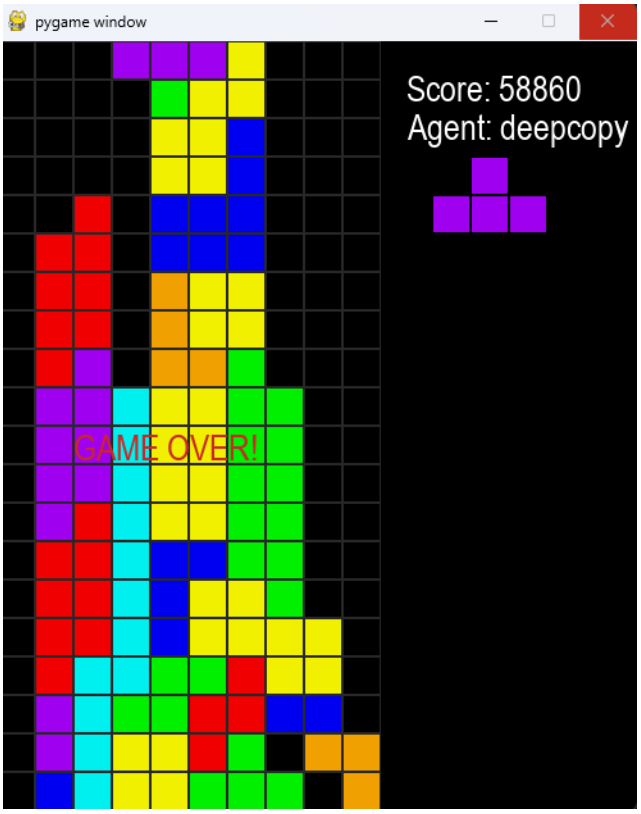
Figura 9 – Segundo melhor resultado obtido pelo agente com *lookahead*.



Fonte: Autoria própria (2026).

A Figura 9 demonstra um desempenho elevado do agente com *lookahead*, mantendo o tabuleiro relativamente organizado mesmo em uma etapa avançada da partida. Esse comportamento reforça a eficiência da previsão da próxima peça na preservação da estabilidade do jogo.

Figura 10 – Melhor resultado obtido pelo agente com *lookahead*.



Fonte: Autoria própria (2026).

O melhor resultado do agente com *lookahead*, apresentado na Figura 10, evidencia maior capacidade de planejamento, com melhor controle da altura do tabuleiro e redução de configurações desfavoráveis. Esse desempenho confirma a contribuição da antecipação de estados futuros para prolongar a partida e aumentar a pontuação.

6 CONCLUSÕES E TRABALHOS FUTUROS

O presente trabalho teve como objetivo desenvolver e avaliar um agente heurístico para o jogo Tetris, comparando seu desempenho em duas configurações: sem a utilização da técnica de *lookahead* e com a utilização dessa estratégia. A proposta partiu da compreensão do Tetris como um problema de tomada de decisão em tempo real, no qual cada jogada influencia diretamente os estados futuros do tabuleiro e, consequentemente, a continuidade da partida.

A partir dos experimentos realizados, foi possível verificar que o uso do *lookahead* contribuiu significativamente para a melhoria do desempenho do agente. Ao considerar não apenas a peça atual, mas também a próxima peça disponibilizada pelo jogo, o algoritmo passou a tomar decisões mais seguras, reduzindo a formação de buracos, controlando melhor a altura das colunas e mantendo o tabuleiro em uma configuração mais estável. Como consequência, observou-se aumento na pontuação média, no número de linhas eliminadas e no tempo até o *Game Over*.

Os resultados demonstraram que a avaliação apenas da peça atual, embora seja capaz de produzir respostas rápidas, apresenta limitações em partidas mais longas, pois pode gerar decisões adequadas no curto prazo, mas prejudiciais para os estados seguintes. Por outro lado, a combinação entre função heurística, simulação de estados por meio de *deepcopy* e antecipação da próxima peça permitiu ao agente avaliar melhor as consequências de suas jogadas e selecionar alternativas mais favoráveis para a continuidade da partida.

Dessa forma, os objetivos propostos foram alcançados, uma vez que foi possível recriar a mecânica fundamental do Tetris, implementar uma função heurística de avaliação, desenvolver agentes com diferentes estratégias de decisão e comparar seus desempenhos por meio de métricas quantitativas. A análise dos resultados evidenciou que o *lookahead* é uma estratégia eficiente para melhorar a tomada de decisão em ambientes dinâmicos e combinatórios, especialmente quando associado a critérios de avaliação do estado do tabuleiro.

Apesar dos resultados positivos, o trabalho apresenta algumas limitações. A principal delas está relacionada ao horizonte de previsão adotado, restrito à peça atual e à próxima peça. Embora essa estratégia tenha apresentado ganhos relevantes, ela ainda não considera sequências mais longas de peças, o que poderia ampliar a capacidade de planejamento do agente. Além disso, os pesos da função heurística foram definidos previamente, podendo ser ajustados de forma mais sistemática em estudos futuros.

Como trabalhos futuros, recomenda-se investigar estratégias mais sofisticadas de busca

e otimização, como o *beam search*, que mantém apenas os estados mais promissores a cada nível de busca, reduzindo o custo computacional em comparação à expansão completa das possibilidades. Também podem ser explorados algoritmos genéticos, aprendizado por reforço e métodos automáticos de ajuste de pesos da função heurística, possibilitando comparações mais amplas entre diferentes abordagens.

Embora o presente trabalho tenha sido aplicado ao contexto do Tetris, a lógica utilizada pelo agente pode ser relacionada a problemas reais de planejamento e automação. A simulação de estados, a avaliação de custos e a antecipação de consequências são estratégias presentes em diferentes sistemas automatizados, como linhas de montagem, braços robóticos, sistemas de armazenamento e processos de construção automatizada. Nesses cenários, assim como no jogo, uma decisão aparentemente eficiente no curto prazo pode gerar dificuldades em etapas futuras. Dessa forma, o estudo contribui conceitualmente para a compreensão de técnicas de tomada de decisão em ambientes dinâmicos, nos quais é necessário escolher ações satisfatórias em tempo reduzido.

Por fim, este trabalho demonstra que a combinação entre heurísticas bem definidas, simulação de estados e técnicas de antecipação pode contribuir para o desenvolvimento de agentes inteligentes mais eficientes. Embora aplicado ao contexto do Tetris, o estudo também dialoga com problemas mais amplos de planejamento e controle em ambientes dinâmicos, nos quais decisões rápidas e aproximadas são necessárias para alcançar bons resultados.

REFERÊNCIAS

ACKERMAN, D. **Tetris: The Games People Play**. New York: Simon and Schuster, 2018.

BURKE, E. K.; KENDALL, G. **Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques**. 2nd. ed. [S.l.]: Springer, 2014.

COOK, W. R. **Deep Copying in Object-Oriented Systems**. Austin, Texas, 1992.

DEMAINE, E. D.; HOHENBERGER, S.; LIBEN-NOWELL, D. Tetris is hard, even to approximate. In: **Proceedings of the 9th Annual International Conference on Computing and Combinatorics (COCOON)**. Berlin, Heidelberg: Springer, 2003. p. 351–363.

FAHEY, C. **Tetris AI: computer plays Tetris**. 2008. Disponível em: http://colinfahey.com/tetris/tetris.html. Acesso em: 16 jun. 2026.

FAHEY, C.; Dellacherie. **Tetris AI Project**. 2003. http://colinfahey.com/tetris/tetris.html. Acesso em: 25 set. 2025.

GROGONO, P.; SAKKINEN, M. Copying and comparing: Problems and solutions. In: **Proceedings of the ECOOP Workshop on Copying, Cloning and Assignment**. Lisbon, Portugal: ECOOP, 1999.

HEARN, R. A.; DEMAINE, E. D. Games, puzzles, and computation. **Theoretical Computer Science**, v. 343, 2005. Disponível em: https://doi.org/10.1016/j.tcs.2005.06.008.

HERRMANN, J. W. A genetic algorithm for minimax optimization problems. In: **Proceedings of the IEEE Congress on Evolutionary Computation (CEC)**. Anchorage, Alaska: IEEE, 1998.

LUNDGAARD, N.; MCKEE, B. Reinforcement learning and neural networks for tetris. In: **Proceedings of the ICML-98 Workshop**. [S.l.: s.n.], 1998.

VIDAL, V. A lookahead strategy for heuristic search planning. In: **Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS 2004)**. Whistler, Canada: AAAI Press, 2004. p. 150–159.

ZHANG, M. R. *et al.* Lookahead optimizer: k steps forward, 1 step back. In: CURRAN ASSOCIATES, INC. **Advances in Neural Information Processing Systems (NeurIPS)**. 2019. Disponível em: https://github.com/michaelrzhang/lookahead.