Code Sandwich: Um Jogo de Cartas para Apoio ao Ensino e Aprendizagem de Lógica de Programação e Algoritmo

Code Sandwich: A Card Game to Support the Teaching and Learning of Programming Logic and Algorithms

Luciano Rosin da Luz[1], Franciele Beal[2], Alinne Cristinne Corrêa Souza[2], Marlon Marcon[2], Francisco Carlos Monteiro Souza[2]

[1]Coordenação de Engenharia de Software - COENS-DV
Universidade Tecnológica Federal do Paraná - Campus Dois Vizinhos (UTFPR)
Estr. p/ Boa Esperança, s/n - KM 04 - Zona Rural, Dois Vizinhos - PR - Brasil

[2]Programa de Pós-Graduação em Informática (PPGI-CP/DV)
Universidade Tecnológica Federal do Paraná (UTFPR)
Multicampi Cornélio Procópio e Dois Vizinhos – PR – Brazil

lucianorosin@alunos.utfpr.edu.br, fbeal@utfpr.edu.br

{alinnesouza, marlonmarcon,franciscosouza}@utfpr.edu.br

Abstract. Introduction:*Traditional teaching strategies struggle to engage students in abstract subjects such as Programming Logic and Algorithms.* ***Objective:*** *This paper presents Code Sandwich, a card game that uses the metaphor of assembling a sandwich to teach Programming Logic and Algorithms in basic education.* ***Methodology or Steps:*** *The game was validation through computational simulations, qualitative Pedagogical Coherence Analysis, and the Deep Knowledge Tracing technique.* ***Results:*** *The game is a promising pedagogical tool to overcome abstraction barriers in Computational Thinking, as it achieved full coverage of the 66 key concepts.*
Keywords *Computational Thinking, Unplugged Computing, Game-Based Learning, Basic Education*

Resumo. Introdução: *Estratégias didáticas tradicionais apresentam dificuldades para engajar estudantes em disciplinas abstratas como Lógica de Programação e Algoritmos.* ***Objetivo:*** *Este artigo apresenta o Code Sandwich, um jogo de cartas que usa a metáfora da montagem de um sanduíche para ensinar Lógica de Programação e Algoritmos na educação básica.* ***Metodologia ou Etapas:*** *A validação do jogo baseou-se em simulações computacionais, Análise de Coerência Pedagógica qualitativa e a técnica de Deep Knowledge Tracing.* ***Resultados:*** *O jogo é uma ferramenta pedagógica promissora para superar barreiras de abstração no Pensamento Computacional uma vez que alcançou a cobertura integral dos 66 conceitos-chave.*
Palavras-Chave *Pensamento computacional, Computação Desplugada, Aprendizagem Baseada em Jogos, Educação Básica.*

1. Introdução

A educação contemporânea passa por uma transformação significativa impulsionada pela revolução digital. Estratégias didáticas tradicionais frequentemente apresentam

dificuldades para engajar estudantes da era digital, os chamados "nativos digitais" [Prensky 2001]. O ensino de disciplinas abstratas, como Lógica de Programação e Algoritmos, apresenta desafios significativos na educação contemporânea. Métodos expositivos tradicionais frequentemente falham em engajar os estudantes, resultando em dificuldades de compreensão, desmotivação e elevadas taxas de evasão [Prensky 2001, Giraffa e Costa 2013]. Nesse cenário, a Aprendizagem Baseada em Jogos (Game-Based Learning – GBL) destaca-se como uma abordagem pedagógica promissora, pois combina elementos lúdicos com o desenvolvimento cognitivo, promovendo a experimentação e o *feedback* imediato [Gee 2003, Kapp 2012].

A tradução de conceitos abstratos em experiências concretas é essencial para o desenvolvimento do Pensamento Computacional (PC) [Wing 2006], habilidade central no ensino de algoritmos e prevista em diretrizes educacionais nacionais [Brasil. Ministério da Educação 2018, Brasil. Ministério da Educação 2022] e estaduais [Paraná. Secretaria de Estado da Educação 2024]. Embora ferramentas digitais possuam alto potencial de engajamento, sua adoção esbarra em limitações de infraestrutura nas escolas. Por outro lado, as alternativas analógicas existentes costumam abordar a programação de forma fragmentada. Observa-se, assim, uma carência de recursos que integrem, de maneira tangível, a estrutura fundamental de um programa: a relação entre dados (variáveis) e lógica (comandos).

Para preencher essa lacuna, este artigo apresenta o *Code Sandwich*, um jogo de cartas analógico desenvolvido para apoiar o ensino de lógica de programação na educação básica. Utilizando a metáfora da montagem de um sanduíche, o jogo materializa conceitos abstratos — variáveis (Ingredientes) e estruturas de controle (Ações) — em uma experiência de construção algorítmica. A principal contribuição deste trabalho é integrar dados e lógica em uma mecânica lúdica que prioriza a compreensão concreta do processo algorítmico, superando abordagens baseadas em memorização.

Este trabalho está organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados, contextualizando o *Code Sandwich* no cenário de ferramentas lúdicas para o ensino de programação. A Seção 3 descreve o jogo em detalhes, incluindo seus componentes, mecânicas e objetivos pedagógicos. A Seção 4 apresenta a metodologia de validação por simulações computacionais com agentes de inteligência artificial. A Seção 5 discute os resultados obtidos nas simulações. Por fim, a Seção 6 apresenta as considerações finais e perspectivas futuras.

2. Trabalhos Relacionados

Esta proposta fundamenta-se em uma revisão narrativa sobre o ensino de programação através de abordagens lúdicas, categorizadas em: (i) jogos digitais, (ii) jogos analógicos (desplugados) e (iii) metodologias gamificadas.

No âmbito digital, [Macena et al. 2022] apresentam o *Hello Food*, um jogo 2D que usa a metáfora de restaurante para ensinar lógica e estruturas de repetição. Embora eficaz no engajamento, sua aplicação é limitada pela necessidade de infraestrutura tecnológica. Quanto aos jogos desplugados, destacam-se abordagens voltadas à colaboração e à abstração de dados. [Soares et al. 2025] propõem o Codando Aventuras, usando cartas de RPG para estimular a resolução de problemas de lógica de forma colaborativa. O Baralho das Variáveis [Kahwage et al. 2013] permite aos estudantes manipular conceitos

como variáveis inteiras, reais e caracteres, contribuindo para a fixação desses conteúdos. Similarmente, [Galvim et al. 2024] utilizam o BaTips, inspirado no jogo UNO, para tornar o ensino de tipos de dados acessível no Ensino Fundamental.

A análise desses trabalhos revela que a maioria foca em desafios específicos ou conceitos isolados. O *Code Sandwich* diferencia-se ao preencher a lacuna na representação explícita da construção algorítmica por meio de uma metáfora culinária, o jogo torna tangível a relação entre dados (Ingredientes) e lógica (Ações), permitindo que os alunos construam "Receitas"que simulam a estrutura real de um algoritmo

3. O Jogo Code Sandwich

O *Code Sandwich* é um jogo de cartas desplugado desenvolvido para apoiar o ensino de lógica de programação e algoritmos. A proposta utiliza a metáfora da montagem de um sanduíche para representar a construção de um algoritmo, permitindo que conceitos abstratos da computação sejam explorados por meio de uma dinâmica lúdica e tangível. O manual completo do jogo, com regras detalhadas, cartas e exemplos, está disponível em: `https://codesandwich.netlify.app`.

3.1. Componentes do Jogo

O *Code Sandwich* é composto por 203 cartas organizadas em quatro categorias principais, que representam a relação entre dados, lógica e objetivos de um programa. O *design* das cartas foi desenvolvido para alinhar mecânica, nome e significado pedagógico aos conceitos de computação, utilizando uma abordagem humorística para tornar esses conceitos mais acessíveis. As ilustrações utilizadas nas cartas foram prototipadas com auxílio da ferramenta de IA generativa *Manus*.

As **Cartas de Ingrediente** (125 unidades) representam a camada de dados do jogo. A taxonomia divide-se em Pães, Proteínas, Vegetais e Condimentos, que correspondem, respectivamente, a *Strings*, *Integers*, *Booleans* e *Arrays*. Além da função lógica, as cartas (Figura 1a) exploram o lúdico através de artes e piadas internas da área de tecnologia. As **Cartas de Ação** (43 unidades) modelam as estruturas de controle e operações lógicas. Estas cartas definem o fluxo do algoritmo por meio de mecânicas de sequência, condição e repetição. A camada lúdica integra conceitos como *debugging* e tratamento de exceções à dinâmica de jogo, permitindo que os jogadores manipulem a "execução"das receitas e interfiram no progresso dos adversários, tornando tangível o comportamento lógico do código (Figura 1b).

(a) Cartas de Ingrediente

(b) Cartas de Ação

Figura 1. Figura (a) representa exemplos de Cartas de Ingrediente e sua conexão pedagógica: Pão de Centeio (Erro 404), Tofu (Indefinido), Rúcula (Assertiva), Ketchup (Mancha no .CSS). Figura (b) representa exemplos de Cartas de Ação que representam conceitos de programação.

As **Cartas de Receita Final** (25 cartas) representam o objetivo do jogo. Cada carta especifica uma sequência de ingredientes que deve ser montada pelo jogador, simulando a construção de um algoritmo. Por fim, as **Cartas Coringa** (10 cartas) funcionam como recursos especiais associados a práticas de desenvolvimento de *software*, como pseudocódigo, bibliotecas, refatoração e *deploy*, permitindo alterar ou otimizar estratégias durante a partida.

3.2. Mecânica do Jogo

A dinâmica do *Code Sandwich* é por turnos e simula a construção de algoritmos. Os jogadores combinam ingredientes e usam cartas de ação para completar a Receita Final.

3.2.1. Preparação do Jogo (*Setup*)

A preparação do jogo simula a inicialização de um ambiente de desenvolvimento. Os baralhos de "Receita Final"(25 cartas) e "Principal"(178 cartas) são embaralhados separadamente. Cada jogador recebe cinco cartas do baralho principal (mão inicial) e uma carta de Receita Final, que define seu objetivo. Em seguida, define-se o jogador inicial e o jogo prossegue em sentido horário. Cada jogador organiza sua Área de Resultado Final, onde os ingredientes serão posicionados, além de uma pilha de descarte comum.

3.2.2. Estrutura do Turno

O jogo ocorre em turnos, e cada turno é dividido em três fases sequenciais: (1) Compra, (2) Ação, e (3) Verificação. Na **Fase 1 – Compra**, o jogador compra 1 carta do baralho principal e a adiciona à sua mão. Se o baralho principal estiver vazio, a pilha de descarte é embaralhada para formar um novo baralho. A **Fase 2 – Ação** é a fase principal do turno. O jogador deve escolher e executar exatamente uma das três ações:

- *Opção A - Colocar um Ingrediente:* o jogador coloca uma Carta de Ingrediente de sua mão na Área de Resultado Final, seguindo a ordem sequencial da esquerda para a direita. Após colocada, a carta não pode ser movida, exceto por efeitos de cartas de Ação.
- *Opção B - Jogar uma Carta de Ação:* o jogador revela uma Carta de Ação (ou Coringa), aplica seu efeito e a descarta.
- *Opção C - Passar a vez:* o jogador opta por não realizar nenhuma ação no turno.

Por fim, na **Fase 3 – Verificação**, ao final da Fase de Ação, duas verificações ocorrem: (i) *Limite de Mão:* o jogador verifica se possui mais de cinco cartas na mão (descartando o excesso); e (ii) *Verificação de Vitória:* o jogador compara sua Área de Resultado Final com sua carta de Receita Final. Se os requisitos forem atendidos, ele vence.

3.2.3. Condições de Vitória

Para vencer, o jogador deve ser o primeiro a completar corretamente sua Receita Final. No contexto do jogo, esse processo é interpretado como a "compilação"de um algoritmo: a

Receita Final representa a especificação do algoritmo, enquanto a Área de Resultado Final corresponde à sua implementação construída passo a passo pelo jogador. A verificação ocorre ao final de cada turno e segue três critérios:

1. **Ordem correta:** os ingredientes devem aparecer exatamente na sequência especificada pela Receita Final.
2. **Quantidade e instâncias corretas:** a composição deve corresponder exatamente aos ingredientes indicados na receita. Quando um ingrediente aparece mais de uma vez (por exemplo, "Pão de Forma" no início e no fim), o jogador deve utilizar múltiplas cópias da carta correspondente. Cada carta posicionada na Área de Resultado Final representa uma instância distinta na sequência, reforçando o conceito de múltiplas ocorrências de um mesmo dado em um algoritmo.
3. **Ausência de erros (*bugs*):** a Área de Resultado Final não pode conter cartas de *BUG*. Esses erros podem ser introduzidos por oponentes e devem ser removidos pelo jogador utilizando cartas de *debug*.

Além disso, a Área de Resultado Final possui um limite máximo de dez ingredientes. Essa restrição funciona como uma analogia à limitação de memória em sistemas computacionais, incentivando os jogadores a construir soluções precisas e eficientes. A relação entre a Receita Final (o algoritmo especificado) e a sequência de ingredientes construída pelo jogador (o programa compilado) pode ser visualizada na Figura 2.

Figura 2. A Receita Final atua como a especificação do algoritmo, enquanto a Área de Resultado Final materializa sua implementação.

4. Metodologia de Validação do *Code Sandwich*

Esta seção apresenta os resultados da validação computacional do jogo *Code Sandwich*. A validação empregou uma metodologia variada, incluindo: (i) Validação computacional - simulações com agentes de Inteligência Artificial (IA); (ii) Análise por Coerência Pedagógica (ACP) - análise qualitativa por IA generativa; e (iii) implementação de um modelo de *Deep Knowledge Tracing* (DKT) - técnica de modelagem do conhecimento do aluno baseada em redes neurais [Piech et al. 2015], com inspiração na abordagem GameDKT [Hooshyar et al. 2022] (*Game Deep Knowledge Tracing*). O objetivo foi obter evidências sobre o balanceamento do jogo e seu potencial como ferramenta de apoio ao ensino de lógica de programação e algoritmos.

4.1. Validação Computacional

Os *Knowledge Components* (KCs) são os blocos fundamentais de conhecimento ou habilidades que um aprendiz deve dominar em um domínio específico. No contexto do *Code Sandwich*, os KCs foram definidos para mapear os conceitos de lógica de

programação e algoritmos que o jogo se propõe a ensinar. Cada ação ou decisão tomada pelo agente (e, por extensão, pelo jogador) no jogo é associada a um ou mais KCs. O funcionamento dos KCs é central para a modelagem dos Agentes Pedagógicos e para o treinamento do modelo DKT:

- **Maestria:** Cada agente possui um valor de maestria para cada KC, que é uma probabilidade (entre 0 e 1) de o agente aplicar corretamente o conceito associado àquele KC.
- **Associação Ação-KC:** Quando um agente decide realizar uma ação, o sistema identifica os KCs necessários para a execução ótima dessa ação.
- **Probabilidade de Erro:** A probabilidade de o agente cometer um erro (ou seja, escolher uma ação subótima) é inversamente proporcional à sua maestria nos KCs relevantes para a ação ótima. Assim, um agente com baixa maestria em um KC importante para uma jogada tem maior chance de errar, simulando a dificuldade de um aluno em aplicar um conceito que ainda não domina.

A definição dos KCs é importante para a aplicação de técnicas de *Knowledge Tracing*, pois permite rastrear a evolução do conhecimento do aprendiz (simulado) ao longo das interações com o jogo. Para avaliar o *Code Sandwich* de forma sistemática antes de testes com usuários finais, foi desenvolvido um ambiente de simulação e análise computacional. A lógica central do jogo foi implementada em Python, compatível com o *framework* RLCard[1], permitindo a execução automatizada de partidas. Dois tipos distintos de agentes de IA foram desenvolvidos para interagir com o ambiente:

O **Agente Heurístico** foi implementado com um conjunto fixo de regras, por exemplo, jogar o próximo ingrediente da receita, utilizar cartas de *Debug* para remover *bugs*, priorizar cartas de *Ação* que compram mais cartas quando a mão está pequena. Este agente serviu como *baseline* para testes de estresse, validação da funcionalidade central e análise do balanceamento mecânico do jogo por meio da execução de 1.000 simulações completas. Sua natureza determinística (dadas as mesmas opções) permite avaliar a consistência das regras e a viabilidade das diferentes estratégias de vitória (*Receitas Finais*).

Os **Agentes Pedagógicos** são mais complexos, pois buscam representar diferentes perfis de aprendizes, conforme discutido em abordagens de tutores inteligentes [Corbett e Anderson 1995]. Para o jogo, realizou-se a **modelagem de perfis**, com a criação de personas (*Novice* e *Intermediate*) que representam diferentes níveis de conhecimento prévio. A maestria inicial em cada KC foi atribuída aleatoriamente dentro de faixas específicas para cada perfil (e.g., 0.1–0.4 para *Novice* e 0.4–0.7 para *Intermediate*). Quanto à **simulação de erro** ao agir, o agente identifica uma ação ótima com base em heurísticas e, em seguida, calcula a probabilidade de erro considerando sua maestria nos KCs relevantes e parâmetros da persona, como taxa de erro base e aderência à estratégia. Quando ocorre erro, o agente executa uma **ação subótima**, simulando dificuldades típicas de aprendizes em processo de aquisição de conhecimento. Por fim, esses agentes foram utilizados em **500 simulações**, gerando dados para análise de eficácia pedagógica simulada e para o treinamento do modelo DKT.

As simulações com ambos os tipos de agentes geraram *logs* detalhados em formato CSV, registrando cada ação, o estado do jogo e o resultado final. Estes *logs* foram

[1]RLCard: disponível em https://rlcard.org/

processados por *scripts* Python dedicados a calcular métricas quantitativas de validação mecânica e eficácia pedagógica simulada.

4.2. Análise de Coerência Pedagógica (ACP)

Adicionalmente, uma Análise de Coerência Pedagógica (ACP) qualitativa foi realizada. Utilizando uma IA generativa (Google Gemini), avaliou-se o alinhamento da tríade (Metáfora, Mecânica, Conceito) em uma amostra de cartas do jogo, bem como a progressão de dificuldade implícita na sequência ordenada das 25 Receitas Finais (calculada pela média da dificuldade dos KCs associados).

4.3. *Deep Knowledge Tracing (DKT)*

Finalmente, explorou-se a aplicação da técnica DKT que utiliza redes neurais para modelar a evolução do conhecimento do aluno [Piech et al. 2015]. Foi implementada uma Rede Neural Convolucional (CNN), seguindo a arquitetura GameDKT proposta por [Hooshyar et al. 2022] como eficaz para dados de jogos educacionais. O modelo foi treinado com os *logs* sequenciais gerados pelos Agentes Pedagógicos, com o objetivo de prever a probabilidade de sucesso (p.ex., realizar uma ação ótima) do agente na interação seguinte, com base em seu histórico recente de ações.

4.4. Uso de ferramentas e tecnologias de IA Generativa

O presente trabalho usou ferramentas de IA generativa como suporte operacional. A revisão gramatical foi realizada com o *Grammarly*; o *Google Gemini 2.5* foi usado na geração de imagens e ativos visuais do jogo; e o *Manus* na codificação da página *web* e na execução automatizada das análises qualitativas. Ressalta-se que todos os resultados foram revisados pelos autores, que assumem total responsabilidade pelo conteúdo final.

5. Análise e Discussão dos Resultados

5.1. Resultados da Validação Mecânica e Balanceamento

A análise das 1.000 simulações com o Agente Heurístico validou a estabilidade e o equilíbrio do núcleo mecânico do jogo. A **duração das partidas** mostrou-se adequada para um jogo de cartas. A distribuição da duração apresentou média de 33,5 turnos e mediana de 31. A concentração da maioria das partidas abaixo de 50 turnos sugere um tempo de jogo razoável, evitando partidas excessivamente longas ou curtas. A alta taxa de conclusão observada indica que as regras do jogo não conduzem com frequência a estados de *deadlock* nem a empates por limite de turnos, garantindo uma experiência de jogo resolutiva na maioria dos casos.

O **balanceamento dos objetivos** (as 25 Cartas de Receita Final) foi um ponto central da análise. A Figura 3 ilustra a frequência com que cada uma das 25 receitas foi a condição de vitória nas 1.000 simulações. A distribuição demonstra um ótimo equilíbrio: não há receitas muito fáceis ou impossíveis de completar sob a estratégia do agente heurístico. A receita mais frequentemente vitoriosa (Sanduíche de *Multithreading*) ocorreu em cerca de 5% das partidas, enquanto a menos frequente (Sanduíche de *Container* Portátil) ocorreu em cerca de 2,8%. Esta variação é considerada adequada, pois indica que todas as receitas são viáveis, mas podem apresentar diferentes níveis de desafio ou de dependência de cartas específicas, o que incentiva a adaptação estratégica e aumenta a rejogabilidade. A ausência de uma estratégia dominante sugere que o *design* das receitas e a distribuição das cartas de *Ingrediente* e *Ação* estão bem calibrados.

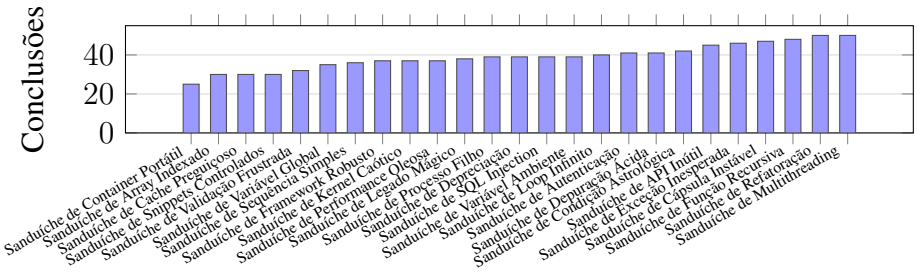

Figura 3. Frequência de Conclusão de Receitas em 1.000 simulações conduzidas por um Agente Heurístico.

5.2. Resultados da Validação Pedagógica

A validação pedagógica investigou a coerência do *design* educacional e a capacidade do jogo de simular e, potencialmente, facilitar o aprendizado. Na **Análise por Coerência Pedagógica (ACP)** as conexões entre a temática culinária e os conceitos de programação foram consideradas, em geral, claras, fortes e criativas. Destacou-se o potencial das metáforas (e.g., Milho representando o *Kernel* com seus Processos/grãos; Carne Desfiada ilustrando os desafios de concorrência em *Multithreading*; API de Sabor simulando uma requisição específica a uma biblioteca) para tornar conceitos abstratos mais tangíveis e fáceis de serem memorizados por estudantes iniciantes. A análise da progressão de dificuldade implícita nas 25 Receitas Finais (baseada na dificuldade média dos KCs associados) também indicou uma curva de aprendizado suave e logicamente estruturada, começando com conceitos fundamentais e avançando gradualmente para tópicos mais complexos, sem saltos abruptos que pudessem frustrar o aprendiz. Verificou-se ainda que 100% dos 66 KCs definidos para o domínio do jogo estão presentes em pelo menos uma carta, garantindo a cobertura do conteúdo pretendido.

Na **Análise de Eficácia Pedagógica (Simulada)** foram examinados os *logs* de 500 simulações com Agentes Pedagógicos (*Novice* e *Intermediate*), gerando evidências quantitativas do potencial do jogo em discriminar níveis de habilidade e em manter o equilíbrio da jogabilidade entre diferentes perfis de jogadores simulados. Conforme visualizado na Figura 4, existe diferenciação de performance entre as personas. O agente *Intermediate* obteve taxas de sucesso (percentual de turnos em que a ação escolhida foi considerada correta/útil) e otimalidade (percentual de ações que coincidiam com a ação ótima definida pela heurística interna) significativamente maiores ($\sim$91%) em comparação com o agente *Novice* ($\sim$74%). Esta diferença confirma que as regras do jogo e as possibilidades estratégicas permitem que agentes com diferentes níveis de conhecimento simulado (p.ex., diferentes probabilidades de escolher a ação ótima e evitar erros) apresentem desempenhos distintos e consistentes com suas definições. Isso sugere que o jogo tem potencial para refletir o nível de compreensão de um jogador humano.

Em relação à duração média dos jogos, ela foi quase idêntica para ambas as personas ($\sim$82-83 turnos), indicando que os erros mais frequentes do agente *Novice* impactam sua eficiência em atingir o objetivo, mas não necessariamente prolongam a partida de forma significativa. Análises adicionais mostraram que essa diferença de performance entre *Novice* e *Intermediate* se mantém na análise individual da maioria dos KCs, e que a taxa de sucesso por Receita Final também permanece equilibrada quando se

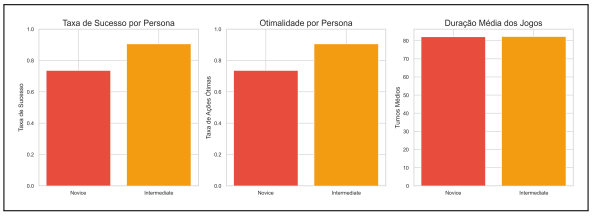

Figura 4. Comparação de Performance (Taxa de Sucesso, Otimalidade, Duração Média) entre as Personas *Novice* e *Intermediate*.

considera a performance combinada das duas personas.

5.3. Resultados do *Deep Knowledge Tracing* (DKT)

Seguindo a abordagem de DKT, que visa modelar o estado de conhecimento latente do aprendiz [Corbett e Anderson 1995], foi treinada uma Rede Neural Convolucional (CNN), inspirada no modelo GameDKT [Hooshyar et al. 2022], para prever a probabilidade de sucesso (ação ótima) da próxima interação do agente. O treinamento do modelo, contudo, não obteve sucesso em aprender padrões preditivos significativos. Na Figura 5 pode ser visualizada a Curva ROC (*Receiver Operating Characteristic*) resultante da avaliação do modelo. A Área Sob a Curva (AUC) foi de 0.504, um valor que indica um desempenho não superior ao acaso (linha pontilhada). Análises da matriz de confusão confirmaram que o modelo desenvolveu um forte viés para a classe majoritária (Sucesso), falhando em discriminar corretamente as instâncias da classe minoritária (Falha).

O desbalanceamento de classes nos dados gerados pelas simulações pedagógicas pode ser a causa. A alta taxa de sucesso e otimalidade dos agentes (especialmente o *Intermediate*) resulta em um conjunto de treinamento onde a classe Sucesso é predominante. Modelos de DKT, como outras redes neurais, podem ter dificuldade em aprender a partir de dados desbalanceados sem a aplicação de técnicas específicas [Hooshyar et al. 2022, Piech et al. 2015]. Esta dificuldade reflete um desafio metodológico na modelagem de IA aplicada aos dados simulados – dados estes que, por sua vez, indicam que o jogo permite aos agentes agirem corretamente na maior parte do tempo – e não necessariamente uma falha intrínseca ao *design* lúdico ou pedagógico do *Code Sandwich*. A literatura sugere abordagens como reamostragem (p.ex., SMOTE) ou uso de pesos de classe para mitigar este problema em trabalhos futuros.

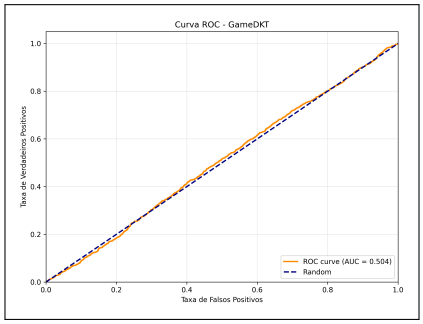

Figura 5. Curva ROC do Modelo GameDKT (CNN) treinado nos logs pedagógicos (AUC = 0.504).

5.4. Discussão dos Resultados

A validação computacional forneceu evidências de que o *Code Sandwich* é um jogo com mecânica adequada, balanceado e pedagogicamente coerente. As simulações mostraram que o jogo funciona corretamente sob diferentes estratégias de agentes, possui uma duração adequada e não favorece objetivos específicos de forma desproporcional (Figura 3). A análise de coerência pedagógica, auxiliada por IA, validou o *design* conceitual das cartas, confirmando que as metáforas empregadas são criativas e têm potencial para facilitar a compreensão de conceitos abstratos de programação. O código-fonte da simulação e os dados brutos utilizados na validação estão disponíveis no repositório do projeto.

A análise de eficácia simulada, utilizando agentes pedagógicos com diferentes níveis de conhecimento, demonstrou a capacidade do jogo em diferenciar performance com base na habilidade simulada (Figura 4). Isso sugere que a progressão no jogo pode refletir o desenvolvimento da compreensão do jogador. A dificuldade em treinar um modelo preditivo de DKT (Figura 5) não invalida os resultados positivos do jogo em si. Pelo contrário, ela pode ser um indicativo de um desafio metodológico que é o desbalanceamento de dados decorrente da alta taxa de sucesso dos agentes simulados, que pode ser comum em contextos educacionais onde o objetivo é que o aprendiz tenha sucesso [Hooshyar et al. 2022]. Este resultado, embora negativo para o modelo de IA, mostra que o *design* do jogo permitiu aos jogadores (simulados) encontrar e executar ações eficazes na maior parte do tempo.

6. Conclusão

Este trabalho apresentou o *Code Sandwich*, um jogo de cartas analógico que utiliza a metáfora da culinária para tornar o ensino de algoritmos e lógica de programação mais tangível e acessível. A proposta foca na integração entre dados (Ingredientes) e lógica (Ações), oferecendo uma alternativa desplugada para mitigar dificuldades de abstração na Educação Básica. O *design* do jogo foi fundamentado na necessidade de recursos que facilitem a compreensão da estrutura fundamental de um programa sem a dependência de infraestrutura computacional.

A validação mecânica e pedagógica, realizada via simulações de IA, demonstrou que o jogo é funcional, estável e equilibrado, com duração de partidas consistente e cobertura integral dos 66 conceitos-chave (*Knowledge Components*) pretendidos. As análises indicaram uma progressão de dificuldade adequada e confirmaram que as metáforas empregadas possuem alto potencial didático. Além disso, a comparação entre diferentes perfis de agentes mostrou que o jogo é capaz de diferenciar níveis de performance, sinalizando sua eficácia em refletir o progresso do jogador.

Apesar dos resultados promissores, a validação exclusiva por agentes computacionais representa uma limitação, pois não captura nuances da experiência humana de aprendizado. Como trabalhos futuros, planejam-se testes com estudantes para coletar *feedbacks* sobre engajamento e clareza, além de estudos experimentais controlados para quantificar o impacto real no desenvolvimento do Pensamento Computacional.

Referências

Brasil. Ministério da Educação (2018). Base Nacional Comum Curricular (BNCC). Documento normativo. Disponível em: http://basenacionalcomum.mec.gov.br/. Acessado em: 4 de julho de 2026.

Brasil. Ministério da Educação (2022). Normas sobre Computação na Educação Básica – Complemento à Base Nacional Comum Curricular. Parecer CNE/CEB nº 2/2022. Disponível em: https://www.gov.br/mec/pt-br/cne/base-nacional-comum-curricular-bncc. Acessado em: 4 de julho de 2026.

Corbett, A. T. e Anderson, J. R. (1995). Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge. *User Modeling and User-Adapted Interaction*, 4(4):253–278.

Galvim, H. d. S., Trindade, G. M., Cordovil, M. W. M., Freitas, E. G. A., Cortez, T. d. S., e Trindade, D. R. d. S. (2024). Sequência didática para educação em computação: Uma análise do jogo de cartas para o ensino de tipos de dados primitivos. In *Anais do XXXV Simpósio Brasileiro de Informática na Educação (SBIE)*, pages 332555–332564, Evento Online. SBC.

Gee, J. P. (2003). *What Video Games Have to Teach Us About Learning and Literacy*. PalgraveMacmillan.

Giraffa, L. M. M. e Costa, M. M. d. (2013). Evasão na disciplina de algoritmo e programação: um estudo a partir dos fatores intervenientes na perspectiva do aluno. In *Congresos CLABES*, pages 1–10.

Hooshyar, D., Huang, Y.-M., e Yang, Y. (2022). GameDKT: Deep knowledge tracing in educational games. *Expert Systems With Applications*, 196:116670.

Kahwage, C., França, E. L. d., Nunes, R. C., Carvalho, R., e Souza, D. T. (2013). Jogo baralho das variáveis. *Revista Brasileira de Informática na Educação (RBIE)*, 21(2):450–459.

Kapp, K. M. (2012). *The Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education*. John Wiley & Sons.

Macena, J., Pires, F., e Melo, R. (2022). Hello food: uma jornada de aprendizagem lúdica em algoritmos, programação e pensamento computacional. In *Anais do XXXIII Simpósio Brasileiro de Informática na Educação (SBIE)*, pages 561–570, Evento Online. SBC.

Paraná. Secretaria de Estado da Educação (2024). Ementa 2025 - Pensamento Computacional - 1ª Série. Documento curricular.

Piech, C., Spencer, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L., e Sohl-Dickstein, J. (2015). Deep Knowledge Tracing. *arXiv preprint arXiv:1506.05908*.

Prensky, M. (2001). *Digital Natives, Digital Immigrants*. McGraw-Hill.

Soares, R. X., Cruz, R. d. J., e Araujo, L. G. d. J. (2025). Codando aventuras: Rpg de mesa como metodologia ativa para o ensino de programação no ensino fundamental. In *Anais do XXIV Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames)*, Salvador/BA. Trilha: Educação.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3):33–35.