UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

GABRIEL BAUER DE OLIVEIRA

Geração de Variações de Jogos de Tabuleiro Via Técnicas de Inteligência Artificial

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Engenharia da Computação

Orientador: Prof. Dr. Anderson Rocha Tavares

Porto Alegre
2023

“Man cannot remake himself without suffering,
for he is both the marble and the sculptor.“
— Alexis Carrel

AGRADECIMENTOS

Para Ágata: o sol sempre haverá de brilhar acima das nuvens de chuva.

Para minha família: a quem sempre me incentivou ao estudo e ao pensamento crítico, além de todo afeto, meus mais sinceros agradecimentos. Sonho que se sonha só é só um sonho, mas sonho que se sonha junto é realidade.

Para meus amigos: essa jornada só foi possível graças a cada um de vocês. Amigo é coisa pra se guardar do lado esquerdo do peito.

Para meus professores: muito obrigado por todos os ensinamentos proporcionados, sejam eles teóricos ou extra-classe. Em especial, um muito obrigado ao meu orientador, Anderson Tavares, que além de um grande mentor e docente, é um ser humano extra-classe.

RESUMO

Evolutionary Game Design (EGD) é uma abordagem de criação de jogos de tabuleiro de maneira automatizada. Sua concepção inicial tem origem no Sistema *Ludi*, um Sistema de Jogos que possuia, entre outras funcionalidades, uma Linguagem de Descrição de Jogos (GDL) própria, um *General Game Player* capaz de realizar a automatização de partidas de jogos descritos nessa linguagem, e um variado conjunto de métricas utilizadas para avaliar um jogo sob diferentes aspectos. Com base nessas características, utilizando o sistema *Ludi*, foi possível realizar a criação de novos jogos de tabuleiro de maneira automatizada utilizando técnicas de Programação Genética.

O presente trabalho visa implementar um Algoritmo de Busca Local utilizando o sistema *Ludii*, sucessor do sistema original *Ludi*, para realizar a procura de variantes de jogos que apresentem características mais interessantes que os originais com relação às métricas adotadas. Para isso, partindo de um jogo base, é realizada a alteração em um parâmetro inteiro ou *booleano* existente na sua definição. A variação obtida é avaliada de acordo com as métricas de análise implementadas no sistema através de partidas automatizadas entre agentes executando o Algoritmo de Busca de Monte Carlo em Árvores.

A partir da abordagem utilizada neste trabalho, foi possível obter variantes dos jogos utilizados como base, mas que mantinham uma similaridade com as mecânicas dos jogos originais.

Palavras-chave: Jogos de Tabuleiro. Busca Local. Geração Procedural de Conteúdo. Design Evolutivo de Jogos.

Generation of Board Games Variations Via Artificial Intelligence Techniques

ABSTRACT

Evolutionary Game Design is an approach to create board games in an automated way. Its initial conception originates from the Ludi System, a Game System that had, among other functionalities, its own Game Description Language (GDL), a General Game Player capable of performing the automation of games described in this language, and a varied set of metrics used to evaluate a game under different aspects. Based on these characteristics, using the Ludi system, it was possible to create new board games in an automated way using Genetic Programming techniques.

The present work aims to implement a Local Search Algorithm using the Ludii system, successor of the original Ludi system, to search for game variants that present more interesting characteristics than the original ones in relation to the adopted metrics. To this, starting from a base game, an integer or boolean parameter in its definition is changed. The variation obtained is evaluated according to the analysis metrics implemented in the system through random matches between agents executing the Monte Carlo Tree Search Algorithm.

From the approach used in this work, it was possible to obtain variants of the games used as a base, but which maintained a similarity with the mechanics of the original games.

Keywords: Board Games. Local Search. Procedural Content Generation. Evolutionary Game Design.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

EGD	*Evolutionary Game Design* - Design Evolutivo de Jogos
GGP	*General Game Playing* - Jogador Geral de Jogos
GDL	*Game Description Language* - Linguagem de Descrição de Jogos
MCTS	*Monte Carlo Tree Search* - Busca de Monte Carlo em Árvores
UCB1	*Upper Confidence Bound* - Limite Superior de Confiança
UCT	*Upper Confidence bound for Trees* - Limite Superior de Confiança em Árvores
PCG	*Procedural Content Generation* - Geração Procedural de Conteúdo

SUMÁRIO

1 INTRODUÇÃO

Jogos de tabuleiro são utilizados como forma de passatempo e de desenvolvimento de habilidades (interação social, raciocínio lógico, concentração, entre outras) desde os primórdios do agrupamento dos seres humanos em conglomerados sociais. Alguns dos jogos mais difundidos em diferentes regiões são aqueles que retratam vivências do cotidiano, como o Bagha Guti[1], de origem da região de Bihar, Índia, em que um tigre tem como objetivo capturar todas as cabras dispostas em um tabuleiro. Outro exemplo famoso é o da família de jogos da mancala, com provável origem na região do Egito, em que o jogo representa plantações e colheitas, bem como o ato da semeadura, atividade cotidiana dos habitantes da região em tempos mais antigos.

Apesar dos jogadores serem capazes de determinar se um jogo de tabuleiro os diverte, definir a maneira como elementos que formam esse jogo interagem, sejam essas interações através dos seus componentes físicos, das condições de vitória, ou de situações que possam ocorrer durante o desenrolar de partidas, é uma tarefa não-trivial. Isso se deve principalmente ao fato de que pequenas mudanças nas relações existentes entre esses recursos podem apresentar grande impacto na jogabilidade, fazendo com que uma pequena variação nas suas regras muitas vezes gere jogos que não possuem o mesmo interesse por parte dos seus jogadores.

Realizar a criação de regras de jogos de tabuleiro de maneira automatizada vem se mostrando uma tarefa desafiadora, pois o nível de abstração escolhido para realizar a representação dos elementos indica a capacidade de recombinar e agrupar tais elementos a fim de obter novos jogos. Além disso, analisar apenas os componentes que formam um jogo pode não representar as sensações que o mesmo gera, sendo necessário avaliar esses jogos através de embates que produzem informações quantitativas sobre o andamento das partidas.

Em sua tese de doutorado, (BROWNE, 2011c) criou o sistema *Ludi* com a finalidade de realizar a criação automatizada de jogos de tabuleiro através de uma abordagem denominada *Evolutionary Game Design* (EGD), baseada em Programação Genética. A partir dessa abordagem, foi possível obter a criação de novos jogos, sendo o mais famoso deles batizado como Yavalath (apresentado no Capítulo 2). Com base nesse primeiro trabalho, após alguns anos foi concebido o sistema *Ludii*, sucessor do sistema *Ludi*, mas que não tinha como objetivo principal a busca por novos jogos de tabuleiro com base em jogos

[1] Bagha Guti. Disponível em https://ludii.games/details.php?keyword=Bagha%20Guti.

pré-existentes.

Tendo como base o novo sistema *Ludii*, o presente trabalho visa utilizar componentes do sistema, bem como realizar alterações necessárias nesses componentes, para implementar uma técnica distinta de geração automática de variações de jogos de tabuleiro utilizando um Algoritmo de Busca Local. A partir da alteração de um parâmetro inteiro ou *booleano* presente na descrição de um jogo, novas variações são avaliadas em relação a determinadas métricas pré-estabelecidas com o objetivo de buscar por variações de jogos que apresentem melhores *scores* características baseadas nas métricas adotadas.

Com isso, o presente trabalho se estrutura da seguinte maneira:

- No Capítulo 2 são apresentados os conceitos utilizados no trabalho, como o sistema *Ludii*, a Linguagem de Descrição de Jogos implementada no seu sistema, o Algoritmo de Busca de Monte Carlo em Árvores, e o conceito de Busca Local.
- No Capítulo 3 é feita uma revisão bibliográfica acerca dos conteúdos de trabalhos relacionados à área.
- No Capítulo 4 é apresentada a metodologia do trabalho: a utilização do sistema *Ludii* para a implementação de um Algoritmo de Busca Local para realizar a busca de variações de jogos de tabuleiro implementados na linguagem de descrição de jogos utilizada no sistema.
- No Capítulo 5 são apresentados e analisados os resultados da implementação do Algoritmo de Busca Local.
- Por fim, no Capítulo 6 é apresentada a conclusão do trabalho, onde as escolhas realizadas durante a implementação da metodologia são comentadas. Também são abordadas possíveis maneiras de realizar a continuação da pesquisa.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta Seção são explorados os conceitos e fundamentos essenciais para a compreensão deste projeto de pesquisa.

2.1 Sistema *Ludi*

Ludi foi um *General Game Player* (GGP) desenvolvido por (BROWNE, 2011c) em sua tese de doutorado. O objetivo original do sistema *Ludi* era jogar, mensurar e gerar novos jogos com relação às características dos jogos definidos em seu escopo.

No sistema *Ludi*, a representação de jogos era limitada a jogos combinatoriais, que possuem as seguintes características:

- Finitos: O jogo termina em um número finito de jogadas;
- Discretos: As jogadas são alternadas e os tabuleiros tem posições discretas bem definidas;
- Determinísticos: Não há elementos de aleatoriedade (como o lançamento de dados);
- Informação Perfeita: Toda a informação necessária para os jogadores realizarem as jogadas está contida no tabuleiro.

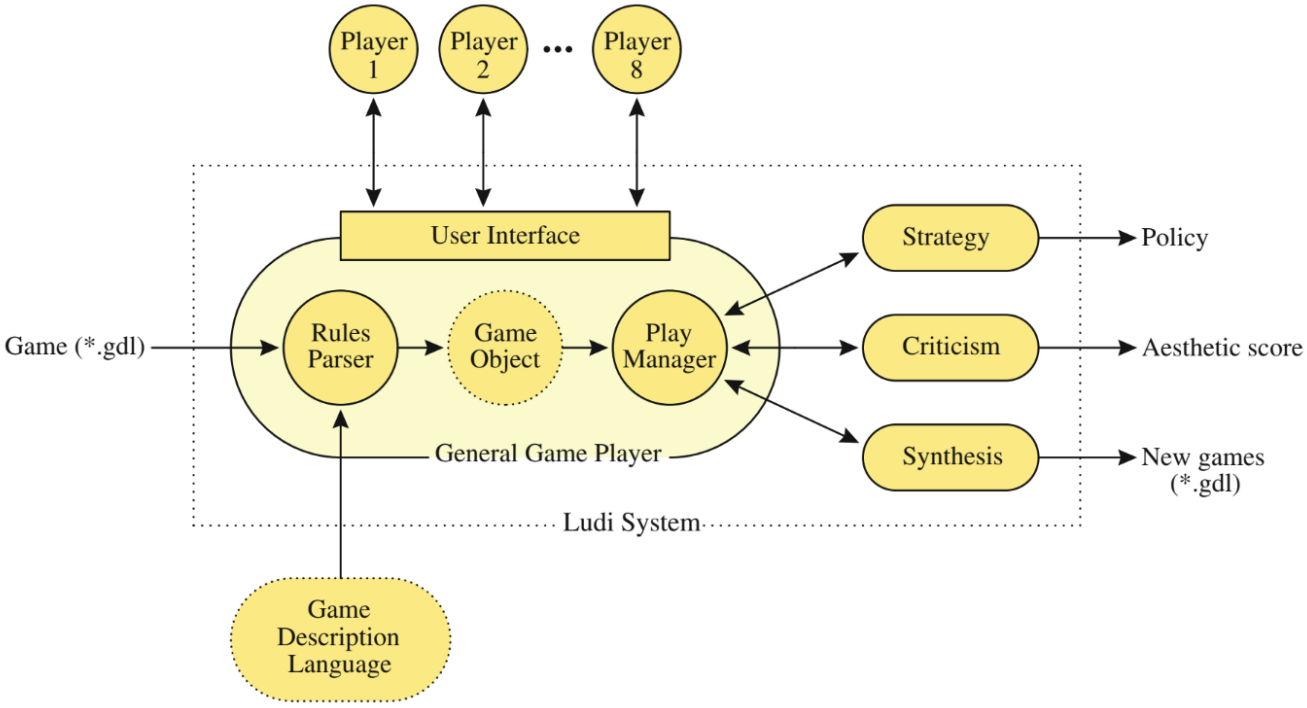
2.2 Sistema *Ludii*

2.2.1 Características Gerais do Sistema *Ludii*

A principal linguagem utilizada para o desenvolvimento do sistema *Ludii*, sucessor do sistema *Ludi*, é Java. Os principais módulos que compõem o sistema do *Ludii* (cuja visão geral se encontra na Figura 2.1) são:

- Jogador Geral de Jogos: Realiza a interpretação dos jogos e coordena as partidas;
- Módulo de Estratégia: Informa os jogadores (quando os mesmos são automatizados) sobre o planejamento de movimentos durante partidas;
- Módulo de Crítica: Mede a qualidade de um jogo descrito na GDL do sistema; e
- Módulo de Síntese: Gera novos jogos de maneira aleatória.

Figura 2.1 – Visão geral do sistema *Ludii*.



Fonte: (BROWNE, 2011a).

2.2.2 Módulo de Crítica

O Módulo de Crítica implementado no sistema *Ludii* é o componente responsável por analisar um jogo definido na sua GDL com relação a diversos aspectos. Avaliar um jogo, indicando se o mesmo possui características que o tornam atrativo para quem o joga, muitas vezes é uma tarefa subjetiva. Para (THOMPSON, 2000), algumas das características básicas que tornam um jogo atrativo são:

- Duração: Jogos devem ser interessantes durante toda a sua duração;
- Clareza: As mecânicas de um jogo não devem ser confusas;
- Drama: Deve haver pelo menos a esperança da recuperação quando em posições de desvantagem;
- Decisão: Jogos devem terminar rapidamente assim que um jogador tem a vitória assegurada.

Em (BROWNE, 2011d), incorporando o conceito de medida estética concebido por (BIRKHOFF, 2013) e aprimorado por (STINY; GIPS, 1978), o Módulo de Crítica do Ludi é responsável pela interpretação de um jogo através de medidas estéticas obtidas a

partir de partidas automatizadas, e onde, ao final da avaliação, um jogo é caracterizado por um valor estético.

Na sua categorização, (STINY; GIPS, 1978) realizam a distinção entre modo construtivo e modo evocativo para o entendimento de um objeto. No modo construtivo, um objeto é entendido através de suas regras e elementos que o compõem. No modo evocativo, um objeto é entendido através das associações, ideias e emoções que o mesmo provê. Com base nessas duas maneiras de realizar a análise de um objeto, o Módulo de Crítica do Ludi incorpora o modo evocativo na sua forma de realizar a avaliação de um jogo, pois essa forma permite encontrar métricas que representam situações que ocorrem durante as partidas.

Ainda em (BROWNE, 2011d), são definidos 57 critérios estéticos a serem analisados em um jogo, sendo esse critérios divididos em 3 principais categorias:

- Intrínsecos: 16 critérios relacionados a características físicas de um jogo, como formato do tabuleiro, formato das peças, objetivo do jogo (captura, cercamento, alinhamento, etc). Representam a presença ou a falta de determinada característica em um jogo;
- Viabilidade: 11 critérios relacionados a resultados de partidas, como quantidade de vitórias de cada um dos jogadores, quantidade de empates, relação entre o número de vitórias entre os jogadores, número de partidas que não atingem um resultado em uma determinada quantidade de movimentos preestabelecido, entre outras;
- Qualidade: 30 critérios relacionados a situações ocorridas dentro de partidas, como a mudança da liderança da partida durante turnos de uma partida, número de movimentos realizados por um jogador a partir de um possível desfecho da partida determinado, entre outros.

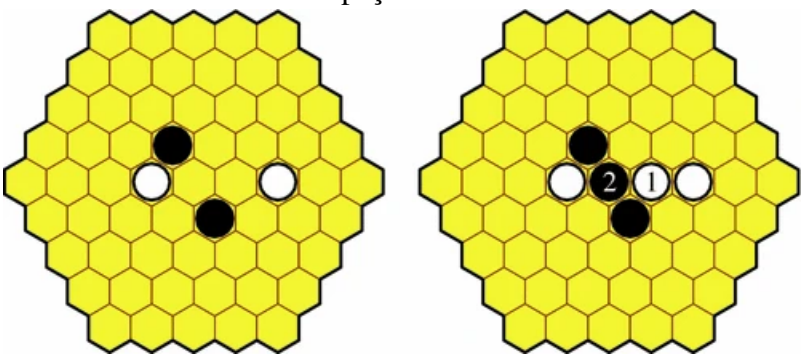
Ao final da análise do jogo, a cada um dos critérios avaliados é atribuido uma pontuação entre -1 e 1, onde esse valor indica o predomínio estimado desse critério no jogo. Em (BROWNE, 2011d), o critério estético final A_s de um jogo é obtido através da Equação 2.1, onde C é o número critérios a serem utilizados, A_i representa o valor estimado de cada um dos critérios estéticos aplicados na análise do jogo, w_i representa um peso associado a cada critério (definidos empiricamente através de políticas utilizadas pelos agentes nas partidas automatizadas), e w_0 representa um termo de viés (utilizado

para o aumento da correlação geral da medida estética):

$$A_s = (\sum_{i=1}^{C} A_i w_i) + w_0 \ . \tag{2.1}$$

A utilização dos critérios estéticos definidos nessa metodologia de trabalho possibilitou a descoberta de jogos com mecânicas interessantes, sendo o mais famoso deles batizado como Yavalath (representado na Figura 2.2), onde os jogadores devem formar uma linha com 4 peças em sequência, mas não podem formar linhas com 3 peças (o que resulta na derrota automática).

Figura 2.2 – Posição de jogadores durante partida do Yavalath. Na figura da direita, as brancas realizaram uma jogada na posição 1, forçando as pretas a jogarem na posição 2 para evitar a derrota. Entretanto, ao colocar uma peça preta na posição 2, as pretas sofrem uma derrota automática ao alinharem 3 peças.



Fonte: (BROWNE, 2011e).

2.3 GDL - Game Description Language

Os elementos que formam a linguagem, denominados *ludemes*, possuem uma hierarquia de composição na descrição de um jogo, o que possibilita a representação direta dessa descrição através de uma árvore, onde cada nó da árvore representa um *ludeme* utilizado na descrição do jogo. Cada *ludeme* implementado na GDL do sistema possui uma correspondência direta a uma classe implementada no sistema. De acordo com (PIETTE; BROWNE; SOEMERS, 2023), as definições das funções das classes que implementam a interface *Ludeme* são:

- *Game*: *Ludeme* raiz de qualquer árvore de *ludemes*;

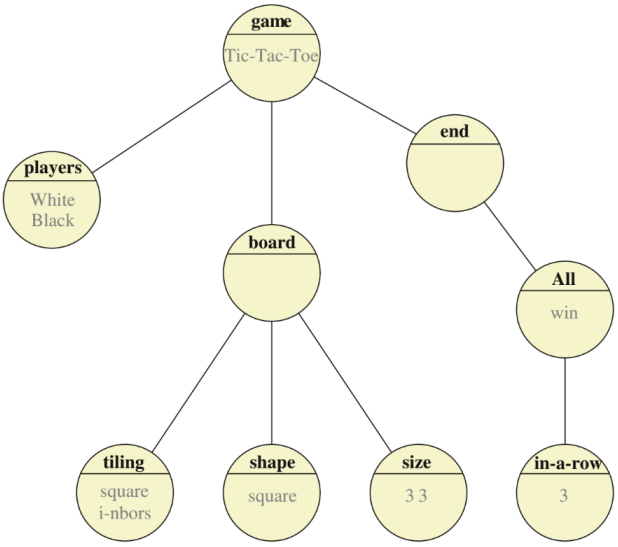
- *Players* e *Mode*: Membros diretos do *ludeme* `game`;
- *Equipment*: Qualquer *ludeme* que faça parte do jogo, como tabuleiro, peças, dados, cartas, etc;
- *Rules*: *Ludemes* que representam as regras de início, meio e fim do jogo;
- *Game*: *Ludemes* raiz de qualquer árvore de *ludemes*;
- *Functions*: Os *ludemes* representando funções utilizadas na descrição de um jogo, como funções booleanas, funções de grafos, funções inteiras, funções de direção e funções de região;
- *Types*: Inclui *ludemes* de enumerações, como direção de bússola (N, S, W, E, ...), de tipos de atribuições, como (*Mover*, *Next*, P1, ...), ou de tipos de região, como (*Vertex*, *Edge*, *Cell*);
- *Auxiliary*: Inclui *ludemes* como pares usados no *ludeme* `Map` para realizar a associação de dois elementos ou a estrutura de *ludemes from/between/to* muito utilizada no *ludeme* do tipo `moves`.

A representação do popular Jogo da Velha utilizando a GDL do sistema *Ludii* e a respectiva árvore gerada a partir dessa descrição (Figura 2.3) são mostradas a seguir:

```
(game Tic-Tac-Toe
    (players White Black)
    (board
        (tiling square i-nbors)
        (size 3 3)
    )
    (end (All win (in-a-row 3)))
)
```

Fonte: The Ludi System. Fonte: (BROWNE, 2011c).

Figura 2.3 – Árvore de ludemes para o Jogo da Velha.



Fonte: (BROWNE, 2011c).

2.3.1 Representação de Jogos em Memória

Cada jogo descrito em um arquivo *.lud* carregado no sistema *Ludii* passa por um processo de compilação para que o jogo possa ser utilizado pelo sistema. Ao final, é gerado um objeto do tipo `description` contendo atributos utilizados pelo sistema. Alguns dos principais atributos contidos nesse objeto são:

- `tokenForest`: objeto da classe `TokenForrest` que implementa uma floresta de *tokens* (cada floresta é uma lista de *tokens*) formando a descrição expandida de um jogo. A classe `Token`, por sua vez, contém a informação de construção da árvore de *ludemes* do jogo, onde um token pode ser de um dos tipos `Class`, `Array` ou `Terminal`;
- `callTree`: árvore de objetos do tipo `Call` contendo cada item da árvore de *ludemes* instanciado para os respectivos objetos que os implementam no sistema *Ludii*.

2.3.2 Parâmetros Numéricos e Booleanos

No sistema *Ludii*, objetos do tipo `Terminal` da classe `Call` instanciados na árvore `callTree` possuem um símbolo associado, e esse símbolo permite realizar a distinção de elementos implementados na gramática da GDL do sistema. O símbolo associado

a cada objeto permite realizar a categorização desse símbolo em um jogo carregado em memória.

2.4 UCT

O algoritmo UCT foi inicialmente proposto por (KOCSIS; SZEPESVÁRI, 2006) como uma variação do algoritmo MCTS. A estrutura do algoritmo se baseia na construção de uma árvore de estados, onde cada nodo N corresponde a um estado do jogo, e onde os filhos $L_1, L_2, ..., L_n \in expansao(N)$ representam possíveis ações executadas em N, levando a um novo estado.

Nesse algoritmo, cada nó possui a ele associado dois valores. são eles:

- $partidas\_jogadas$: Valor que indica quantas partidas foram jogadas até o final passando por esse nó; e
- $partidas\_ganhas$: Valor que indica quantas partidas passando por esse nó resultaram em um estado terminal de vitória.

Esses valores são importantes pois são utilizados para obter os critéros de limites de confiança (que serão abordados na sequência).

O algoritmo UCT se divide em 4 fases de execução. A definição de cada uma das 4 fases é:

- Seleção: um nó folha F da árvore que não esteja completamente expandido é selecionado para que a partir dele seja realizada uma execução completa de uma partida;
- Expansão: a partir da escolha de um nó folha F, contanto que F não seja um estado terminal da partida (vitória/derrota/empate), o nó é expandido. Cada filho de F representa uma jogada válida na partida a partir do estado de F. Na sequência, um nó $L \in expansao(F)$ é escolhido de maneira aleatória como ponto de partida da execução;
- Simulação: após a seleção do nó L, uma partida é executada de forma completa até que seja atingido um estado terminal; e
- Retropropagação: após a execução de uma partida até um estado terminal, cada nó do caminho entre o nó R raiz da árvore e L (incluso) tem os valores de partidas e vitórias alterados com base no resultado obtido na execução da partida.

No algoritmo UCT, pelo fato de apenas uma sequência de nós entre a raiz e um nó

folha ser visitada a cada execução do algoritmo, é necessário realizar um balanço entre a utilização de nós que possuem uma boa razão $\frac{partidas\_ganhas}{partidas\_jogadas}$ e nós que possuam uma razão menor. Esse balanço visa equilibrar a utilização de nós que produzem bons resultados em partidas (utilização essa comumente denominada como *exploration*) e a utilização de nós pouco frequentemente utilizados (utilização essa denominada como *exploitation*).

Para realizar esse balanço é utilizada a equação 2.2 denominada UCB(*Upper Confidence Bound*) de um nó S'. Nessa equação, S' é o nó a ter a sua utilidade calculada, enquanto S é o seu nó pai:

$$UCB(S, S') = \frac{S'.partidas\_ganhas}{S'.partidas\_jogadas} + 2C\sqrt{\frac{\ln{(S.partidas\_jogadas)}}{S'.partidas\_jogadas}} \tag{2.2}$$

Na Equação 2.2, o primeiro termo é denominado *exploitation term*, e esse termo é utilizado para que nodos com uma maior razão $\frac{partidas\_ganhas}{partidas\_jogadas}$ sejam mais utilizados em partidas. Já o segundo termo é denominado *exploration term*. Esse termo é utilizado para que os filhos de um nó pai (cujo valor de $S.partidas\_jogadas$ é o mesmo para todos os filhos, mas onde o valor de $S'.partidas\_jogadas$ é diferente) que tenham sido pouco explorados em partidas, à medida que mais partidas sejam executadas (fazendo com que o numerador da fração aumente), tenham suas chances de serem escolhidos em outras execuções aumentadas.

O fator C é uma constante incorporada para encorajar (valores maiores de C) ou desencorajar (valores menores de C) a busca por novas soluções durante a seleção dos nós.

2.5 Resumo

Nesse Capítulo foram abordados os conceitos fundamentais que servirão como base para a execução do presente trabalho. Foi abordado de maneira sucinta o sistema *Ludi* e seu sucessor, *Ludii*. Deste, alguns de seus módulos foram apresentados, módulos que serão utilizados para a implementação dessa pesquisa. Também foi apresentada a GDL implementada no sistema.

3 TRABALHOS RELACIONADOS

Diversos autores abordam a Geração Procedural de Conteúdo com relação à criação sistemática de elementos presentes em jogos. Majoritariamente, as categorias de jogos abordadas por esses autores são jogos de plataforma digitais ou similares, em que as técnicas de PCG aplicadas visam a obtenção de novos mapas e/ou elementos gráficos, mas onde as mecânicas ou objetivos dos jogos são mantidas.

3.1 Geração de Conteúdo em Jogos Digitais

Em (KELLY; MCCABE, 2006), os autores discutem sobre a utilização de Fractais, Sistemas de Lindenmayer, Ladrilhamento, entre outras técnicas, para realizar a composição de diferentes *designs* de cidades, analisando a verossimilhança com a realidade. Em (SMELIK et al., 2009), os autores avaliam a utilização de técnicas similares para a criação de diferentes tipos de terreno, como montanhas, lagos, rios, vegetações, e inclusive ambientes urbanos. Entretanto, as técnicas abordam aspectos sobre diferentes maneiras de realizar a composição de um mesmo conjunto de elementos, não alterando a mecânica da relação entre os mesmos no objeto final.

3.2 Geração de Mecânicas em Jogos Digitais

Em (TOGELIUS; SCHMIDHUBER, 2008), os autores abordam a criação de jogos de arcade em um *grid* através da descrição de um conjunto com 8 elementos contidos no seu espaço de regras (tempo máximo de partida, *score* máximo, número de oponentes vermelhos, azuis, verdes, e movimentação de oponentes vermelhos, azuis, verdes) e de duas tabelas contendo regras de colisão entre o agente e os oponentes. A evolução das regras de um jogo, cujo conjunto de valores que definem suas regras é inicializado aleatoriamente, ocorre a partir do treinamento de agentes automatizados para a realização de *playtests* (portanto, a avaliação dos jogos gerados é realizada com base no modo evocativo de (STINY; GIPS, 1978)).

Ainda em (TOGELIUS; SCHMIDHUBER, 2008), a função de avaliação tem como base o agente automatizado que apresentou melhor aprendizado no mapa, onde é calculado o *score* médio desse agente ao longo de 5 partidas. A cada geração, é feita a

cópia de um jogo, e são realizadas mutações sobre essa cópia, calculando o *score* médio do jogo original e da nova versão. Caso o *score* da cópia seja maior, o original é substituído, e o processo evolutivo tem sequência. Alguns dos jogos criados, apesar de serem jogáveis, apresentaram má combinação de regras, onde o processo evolutivo não foi capaz de corrigir a não funcionalidade total do objeto final.

Em (COOK; COLTON, 2011), de forma similiar a (TOGELIUS; SCHMIDHUBER, 2008), os autores propõem a criação de um sistema, denominado ANGELINA, para a geração automatizada de jogos de arcade com um agente, oponentes, um grid, obstáculos e relações entre o agente e os obstáculos. Os três principais componentes que representam um jogo no sistema são um mapa, um *layout* (com as mesmas dimensões do mapa) do jogador e dos oponentes, e um conjunto de regras que descreve as interações entre o jogador e os oponentes. Nesse sistema, a metodologia utilizada para a evolução dos jogos, denominados objetos, cria um processo evolutivo para cada uma das três componentes principais do sistema, onde cada processo possui uma função de *fitness* interna e uma função de *fitness* externa, com relação ao objeto em si.

Ainda em (COOK; COLTON, 2011), a partir da metodologia utilizada no design evolutivo, os autores concluem que essa metodologia se provou interessante pelo fato dos processos responsáveis pela evolução dos componentes avaliarem o resultado de cada geração com relação a funções de avaliação internas e externas, o que possibilitou a geração de objetos cuja jogabilidade se mostrou interessante. Entretanto, como o espaço de busca do design se limitou a um similiar utilizado por (TOGELIUS; SCHMIDHUBER, 2008), os jogos gerados estão limitados a esse mesmo espaço de busca, sendo necessária a ampliação dos elementos que o sistema tem conhecimento para a geração de objetos mais diversificados.

Em (NELSON; MATEAS, 2007), os autores propõem o *design* de um jogo como uma atividade de resolução de problemas, dividindo esse espaço em mecânicas abstratas do jogo, representação concreta do jogo, conteúdo temático, e mapeamento de controles. Através dessa divisão, a capacidade de geração de novos jogos é definida pelo conhecimento do sistema sobre o domínio de cada uma dessas divisões. Nesse trabalho, os autores focam no domínio de conteúdo temático, realizando a geração de jogos do estilo *WarioWare* (minijogos envolvendo completar tarefas com relação ao ambiente especificado, como desviar de obstáculos, acertar alvos, etc) através de descrições textuais, analisando a jogabilidade geral do jogo e quão próximo da descrição original o jogo se encontra.

Ainda em (NELSON; MATEAS, 2007), os autores abstraem três estilos de jogo no

domínio de conhecimento do sistema, sendo jogos de evitar obstáculos, jogos de aquisição de elementos, e jogos onde uma tarefa deve ser completada em um determinado tempo. Com esse trabalho, os autores observam que a capacidade de criatividade do sistema com relação ao objetivo do jogo é diretamente relacionada com os estilos de jogos conhecidos *a priori*, limitando sob certos aspectos os resultados obtidos.

3.3 Geração de Jogos de Tabuleiro

No livro (BROWNE, 2011a), é apresentado o sistema Ludii, que implementa uma Linguagem de Descrição de Jogos própria, e é formado por módulos responsáveis pela instanciação de jogos descritos na sua linguagem (Módulo Jogador Geral de Jogos), pelo planejamento de jogadas durante partidas (Módulo de Planejamento), pela atribuição de uma medida estética com base na descrição de um jogo (Módulo de Crítica), e pela geração de novos jogos (Módulo de Síntese).

Em (BROWNE, 2011b), o autor apresenta um *pipeline* que utiliza a Linguagem de Descrição de Jogos implementada para utilização no sistema Ludi, bem como dos módulos presentes no sistema. O *pipeline* proposto tem como objetivo a aplicação de Algoritmos Evolutivos no desenvolvimento de melhores indivíduos (no caso, descrições de jogos) com base em uma população inicial de indivíduos. Nesse *pipeline*, através de técnicas como a recombinação gênica e a mutação, novos indivíduos são gerados. Esses novos indivíduos passam, inicialmente, por uma série de checagens com base em fatores construtivos (e.g. a análise de *ludemes* que geram um jogo impossível de ser instanciado pelo sistema). Após um indivíduo ter sua estrutura avaliada, uma série curta de *playtests* é realizada, onde a política utilizada pelos agentes é escolhida entre as políticas do indivíduo e dos indivíduos cujos genes foram recombinados para sua criação. Após essa série de partidas, são mantidos no *pipeline* os indivíduos que não apresentarem tempo entre jogadas maiores que 15 segundos, descartando possíveis jogos lentos do processo evolutivo.

Ao final do *pipeline*, após checagens acerca de similaridade com outros jogos (através da comparação das árvores de *ludemes* geradas a partir das suas descrições), os jogos restantes passam por uma série maior de *playtests*, onde, ao final das partidas, o Módulo de Crítica atribui uma medida estética para o jogo (a escolha das políticas utilizadas no Módulo Jogador Geral de Jogos influencia na atribuição dos pesos w_i na Equação 2.1). Com base nas medidas estéticas de um jogo relacionadas à sua viabilidade

(como duração, completude, balanço e vantagem de início), jogos com requisitos mínimos são escolhidos para integrar a população de indivíduos utilizada para a geração de novas gerações, repetindo o processo de evolução dos indivíduos.

3.4 Resumo

Neste Capítulo, foram vistas abordagens relacionadas tanto a PCG quanto à geração de mecânicas de jogos, onde comumente a criação de mecânicas envolve jogos digitais de plataforma. Destacando o trabalho realizado por (BROWNE, 2011b) para a geração de jogos de tabuleiro, a abordagem a ser seguida no presente trabalho se diferencia do seu trabalho com relação à abordagem a ser seguida para a geração de novos jogos, conforme será visto nos Capítulos 4 e 5.

4 PROPOSTA

No Capítulo 2 foi comentado de maneira geral o processo original de EGD obtido com o sistema *Ludi*, resultando na criação de um jogo comercialmente bem sucedido utilizando a abordagem de Programação Genética. Após a sua concepção, o sistema *Ludi* não teve uma continuação, originando mais tarde o sistema *Ludii*.

Com base nos conceitos apresentados anteriormente, a proposta do seguinte trabalho é realizar a implementação do Algoritmo de Busca Local (Algoritmo 1) para realizar a geração de variações de jogos de tabuleiros implementados na GDL do sistema *Ludii*. Pelo fato da motivação da criação do sistema *Ludii* ser diferente da sua versão original, a abordagem a ser utilizada para obter tais variações será baseada em um Algoritmo de Busca Local.

4.1 Busca Local

A abordagem de Busca Local é uma das ideias mais intuitivas com relação à necessidade de realizar a busca de melhores soluções para um problema em um espaço de soluções. Nessa abordagem, cada solução $S_1, S_2, ..., S_n$ de um problema é representada como um elemento em um espaço de estados cujas dimensões são os parâmetros que formam cada solução. Além disso, é utilizada uma função $C(S)$ de avaliação que realiza o cálculo da aptidão de um indivíduo com relação ao problema.

Com base nessa ideia, uma solução inicial S é escolhida no espaço de soluções, e o valor de $C(S)$ é calculado. A cada iteração do algoritmo uma nova solução vizinha S' à melhor solução até o momento é escolhida. Essa solução vizinha é obtida ao realizar uma alteração em um ou mais parâmetros existentes na melhor solução.

O Algoritmo de Busca Local (Algoritmo 1 1), proposto como abordagem para o presente trabalho, é uma variação do Algoritmo Hill-Climbing. Em sua versão original, o Algoritmo Hill-Climbing executa a procura de soluções vizinhas S' a um determinado estado S até que não sejam encontradas soluções com um valor de $C(S')$ superior que o de $C(S)$. Para problemas cujo cálculo da função C é resultado da simples aplicação dos valores dos parâmetros da solução S, a procura por soluções vizinhas com melhores valores de C é uma tarefa mais simples que a necessária para o problema de avaliação de variações de jogos, pelo fato da avaliação de um jogo ser realizada através de *playtests*, tornando a função de avaliação computacionalmente custosa. Como um jogo possui um

número prévio de parâmetros desconhecido, e a cada parâmetro pode ser atribuido um valor em uma faixa também previamente desconhecida, o número de possíveis vizinhos a uma solução S cresce de uma maneira que se torna inviável a avaliação de todos os possíveis candidatos à procura de uma melhor solução vizinha S'.

Algoritmo 1: Busca Local

```
Input: initial_solution, iteration_range
Output: best_solution
best_solution← initial_solution
best_score ← evaluate(initial_solution)
for i in range(1, iteration_range) do
    current_solution ← random_neighbor(best_solution)
    current_score ← evaluate(current_solution)
    if current_score > best_score then
        best_solution ← current_solution
        best_score ← current_score
    end
end
return best_solution
```

4.2 Alteração de Parâmetros

Para cada descrição presente durante a iteração do algoritmo de Busca Local, uma lista com todos os seus parâmetros numéricos e *booleanos* é extraída. A partir dessa lista, um único parâmetro é sorteado aleatoriamente e tem seu valor alterado.

Para que seja mantida uma coerência com o valor presente anteriormente, caso o valor V sorteado seja do tipo numérico, o novo valor V' estará na faixa $[0.2V, 1.8V]$. A faixa de valores escolhida para atribuição do novo valor do parâmetro foi utilizada para que o novo tenha uma coerência semântica com o valor anterior. Assim, as variações obtidas possuem valores similares ao jogo anterior, o que permite que o algoritmo de Busca Local realize uma procura gradativa no espaço de estados. Caso o valor sorteado seja do tipo *booleano*, o novo valor será o inverso do valor anterior.

4.3 Método de Avaliação

Para realizar a avaliação de uma variante no Algoritmo de Busca Local implementado, o método de avaliação utilizado consiste na execução de partidas do jogo. Para a

execução das partidas, os agentes automatizados utilizados são baseados no Algoritmo UCT.

O Algoritmo UCT (descrito na Seção 2.4) é um algoritmo de GGP baseado em estatísticas coletadas através de simulações aleatórias de um jogo a partir de seus estados, e que independe de uma heurística específica conhecida sobre o jogo a ser simulado. Pelo fato de cada partida ser executada através de agentes que tomam decisões baseadas em simulações a cada turno, o tempo necessário por partida é relativamente elevado.

Ao final da execução das partidas de uma avaliação, cada variante é avaliada com base nos critérios explicados na Seção 4.4.

4.4 Critérios de Avaliação

Os critérios utilizados para a avaliação de cada um dos jogos são listados a seguir, bem como o seu significado (de acordo com sua definição em (BROWNE, 2011a)):

- *Duration*: medida da média de jogadas necessárias para completar uma partida. É medida através do desvio de jogadas necessárias para completar todas as partidas com relação a um número médio de jogadas desejadas, e indica jogos que não acabam muito cedo ou não se estendem muito;
- *Lead Change*: indica a tendência da liderança da partida ser alterada durante um partida. É calculada somando o número de vezes que os gráficos de liderança da partida se cruzam;
- *Completion*: indica a tendência de partidas acabarem com um vencedor em um número razoável de movimentos. É obtido através da razão entre partidas completadas sobre o total de partidas jogadas;
- *Drama (average)*: indica a tendência de jogadores se recuperarem de posições aparentemente perdidas. É calculado como a divisão da soma da diferença da liderança em cada posição do eventual ganhador da partida sobre o total de movimentos realizados em desvantagem;
- *Decisiveness*: indica se um jogador obtém uma vitória de maneira rápida após atingir um determinado *threshold* de decisibilidade no jogo. É obtido através da média entre todas as partidas da razão entre o número de movimentos da partida após atingir o *threshold* sobre o total de movimentos da partida;
- *Advantage P1*: estima uma tendência natural do primeiro jogador ser o vencedor da

partida. É obtido através da razão absoluta entre o total de partidas vencidas pelo primeiro jogador sobre o esperado de 50% das partidas;

- *Balance*: indica se os jogadores das duas peças possuem chances iguais de vencerem o jogo. É obtido através da diferença absoluta de vitórias dos dois jogadores sobre o total de vitórias;
- *Drawishness*: razão entre o total de partidas que acabam sem um vencedor (empate ou mais de um vencedor) sobre o total de partidas disputadas. Quanto menor melhor, indicando jogos que acabam sem um vencedor;
- *Timeouts*: indica a tendência de um jogo acabar sem um vencedor em um número limite de movimentos, indicando jogos que podem ter objetivos muito difíceis de serem alcançados.
- *Decisiveness Moves*: indica o número de movimentos realizados após um jogador ultrapassar um determinado *threshold* de decisibilidade no jogo.

5 AVALIAÇÃO EXPERIMENTAL

O presente Capítulo apresenta e explica todos os experimentos realizados durante o desenvolvimento deste trabalho. Ao fim, é feita a análise de resultados dos experimentos.

O trabalho será realizado utilizando os módulos do sistema *Ludii*, incorporando suas funcionalidades para realizar a implementação própria do Algoritmo de Busca Local, bem como realizar a manipulação de estruturas relacionadas à descrição de cada jogo compilado no sistema e representado em memória.

O Algoritmo de Busca Local (Algoritmo 1) utilizado foi implementado pelo autor do trabalho e incorporado ao sistema *Ludii*. A escolha dos parâmetros de execução do Algoritmo são aborados na Subseção 5.2.

Cada descrição de um jogo deve ser compilada no sistema utilizando o módulo de compilação para que possa ser gerada a sua descrição correspondente em memória. A partir da representaçao em memória, parâmetros presentes na descrição que possuam correspondência de tipo inteiro ou *booleano* serão adicionados a uma lista de possíveis parâmetros a serem alterados para realizar a geração de vizinhos no espaço de estados.

5.1 Inicialização da Busca Local

O algoritmo inicia a execução partindo de uma definição de um jogo descrito na GDL do sistema *Ludii*.

Os jogos escolhidos para o experimento foram jogos que apresentam características de jogabilidade voltadas diretamente às quantidades de elementos disponíveis pelo fato da implementação dos mecanismos de alteração dos parâmetros dos jogos ser voltada a parâmetros inteiros e *booleanos*.

Para parâmetros inteiros, alguns dos tipos de parâmetros nas definições que poderiam ter seu valores alterados são quantidade inicial de peças, posição inicial de peças, número de linhas ou colunas do tabuleiro, quantidade de repetições de um elemento, entre outros. Para parâmetros *booleanos*, representando mecânicas do jogo, alguns dos tipos de parâmetros que poderiam ter seus valores alterados são possibilidade de saltar sobre peças, ações a serem ou não executadas ao atingir determinado objetivo durante as partidas, entre outros.

Com base nessas informações, a lista dos jogos utilizados como base no expe-

rimento, bem como uma breve descrição de cada um, é (os jogos são apresentados na Figura 5.1):

- *1D Chess*: variação do Xadrez jogada em uma única dimensão, onde o cavalo deve saltar duas casas em linha reta. O objetivo é dar xeque-mate no rei adversário;
- *Alquerque*: jogado por 2 jogadores em um tabuleiro 5x5 com diagonais que se conectam. Cada jogador possui 12 peças. As peças se movem para vértices adjacentes. Ao pular sobre uma peça adversária a mesma é capturada. O objetivo é capturar todas as peças do adversário;
- *Al-Qirq*: jogado por 2 jogadores em um tabuleiro com 3 níveis quadrados de dimensões diferentes conectados pelos seus centros e cantos. O jogo é dividido em duas fases, uma de posicionamento das peças e outra de movimentação para vértices livres adjacentes a peças. Caso 3 peças sejam alinhadas uma peça adversária é retirada (exceto peças em linhas com 3). O objetivo é deixar o jogador adversário com 2 peças;
- *Ancient Merels*: jogado por 2 jogadores em um tabuleiro com 3 níveis quadrados de dimensões diferentes conectados pelos seus centros. O jogo é dividido em duas fases, uma de posicionamento das peças e outra de movimentação para vértices livres adjacentes a peças. Caso 3 peças sejam alinhadas uma peça adversária é retirada (exceto peças em linhas com 3). O objetivo é deixar o jogador adversário com 2 peças;
- *Bravalath*: jogado por 2 jogadores. O jogo começa com uma peça de cada cor. Os jogadores se revezam adicionando uma peça de sua cor em uma posição adjacente a pelo menos uma peça existente. O objetivo é formar uma linha de quatro (ou mais) da sua cor, e o jogador perde automaticamente ao formar uma linha de três peças da sua cor;
- Damas: jogado por 2 jogadores em um tabuleiro 8x8. As peças são colocadas nas casas escuras das 3 primeiras e 3 últimas linhas. As peças se movem nas diagonais da frente. Ao pular sobre uma peça na diagonal a peça adversária é capturada. Ao chegar na última linha oposta, a peça do jogador se torna uma Dama, que pode se mover quantas casas quiser na diagonal. O objetivo é capturar todas as peças adversárias.

Figura 5.1 – Jogos escolhidos para o experimento.

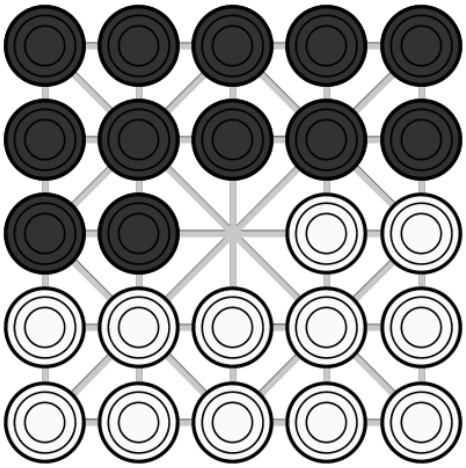
(a) *Bravalath*

Fonte: o autor.

(b) *1D Chess*

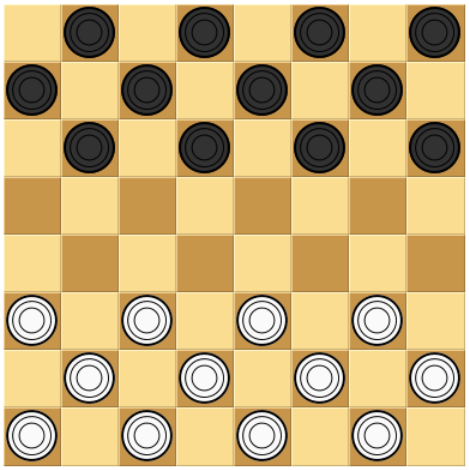
Fonte: o autor.

(a) *Alquerque*



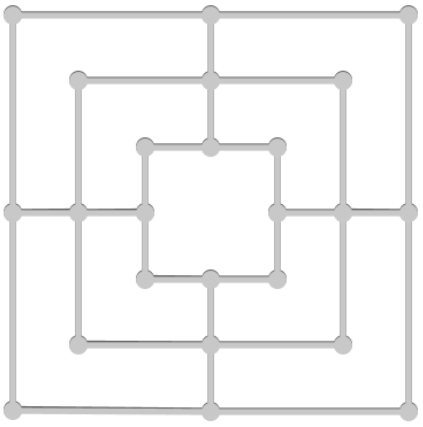
Fonte: o autor.

(b) Damas



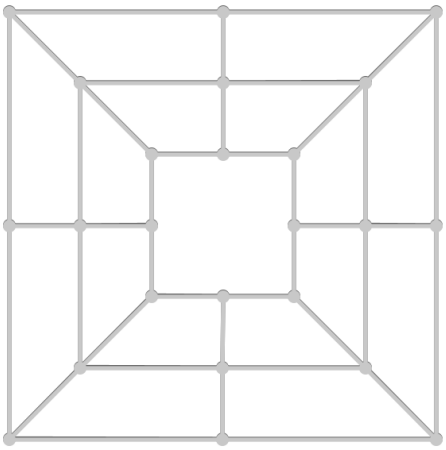
Fonte: o autor.

(c) *Ancient-Merels*



Fonte: o autor.

(d) *Al-Qirq*



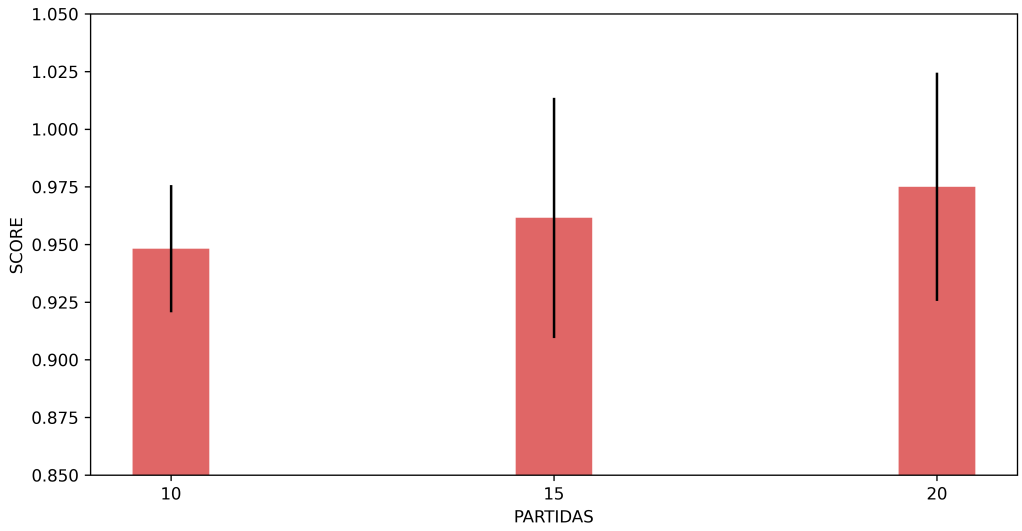
Fonte: o autor.

5.2 Definição do Número de Partidas da Avaliação

Para encontrar o parâmetro de número de partidas a serem executadas a cada avaliação do algoritmo de Busca Local (linha 7 do Algoritmo 1) para obter o *score* de uma variação de um jogo, foram executadas 10 avaliações do *score* do jogo *Alquerque*, alterando o número de partidas executadas em cada avaliação com os valores de 10, 15 e 20. Os gráficos nas Figuras 5.3 e 5.4 apresentam os resultados obtidos para os scores médios e tempos médios, respectivamente.

Com base nos resultados do gráfico da Figura 5.3, é possível observar uma tendência do *score* do jogo sofrer um crescimento. Caso o número de partidas executadas em cada iteração seguisse sofrendo crescimento, a tendência seria o *score* obtido representar com maior precisão um *score* ideal do jogo com relação às métricas adotadas, tendo em vista que mais partidas seriam executadas para realizar o cálculo.

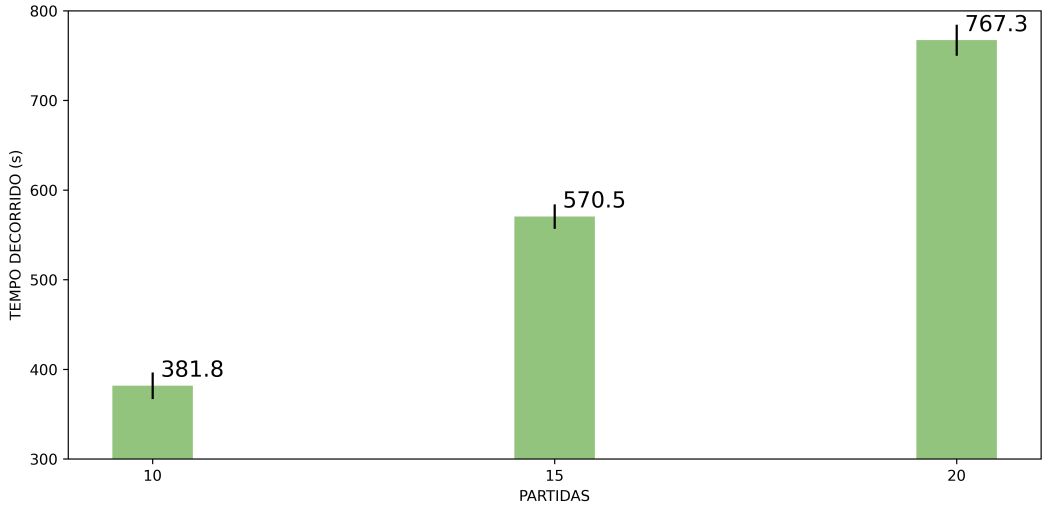
Figura 5.3 – Gráfico de partidas × scores.



Fonte: o autor.

Analisando o gráfico da Figura 5.4, o tempo necessário para a realização de cada iteração também cresce, tendo em vista o aumento do número de partidas executadas em cada avaliação. Entre 10 e 15 partidas há um aumento de 49,42%, enquanto entre 15 e 20 partidas há um aumento de 34,5%.

Figura 5.4 – Gráfico de partidas × tempos.



Fonte: o autor.

Pelo fato dos desvios padrões dos *scores* das avaliações com 15 e 20 partidas, indicados pelas barras escuras no gráfico 5.3, terem valores muito próximos, e pelo fato do tempo da execução de 15 partidas por avaliação ser 34,5% menor que o tempo da execução de 20 partidas por avaliação, para a realização dos experimentos foi escolhido uma quantidade de 15 partidas a cada iteração do algoritmo de Busca Local implementado.

5.3 Execução do Algoritmo

Com os parâmetros definidos, o algoritmo de Busca Local implementado foi executado. A execução do algoritmo para todos os jogos base com uma execução de n = 40 iterações para cada jogo teve uma duração aproximada de 5 dias.

Nas tabelas A.1, A.2, A.3, A.4, A.5 e A.6 do Apêndice A são apresentados os valores individuais das métricas obtidos nas 10 primeiras iterações de cada variação do jogo original e suas variações.

Os pesos utilizados para cada uma das métricas utilizadas foram extraídos de (BROWNE, 2011d), onde cada peso foi obtido através de um experimento realizado com humanos avaliando jogos (valores negativos dos pesos indicam uma correlação negativa entre o valor obtido na métrica e sua influência no *score* final da avaliação).

O experimento realizado nesse trabalho foi dividido em duas etapas. Na primeira

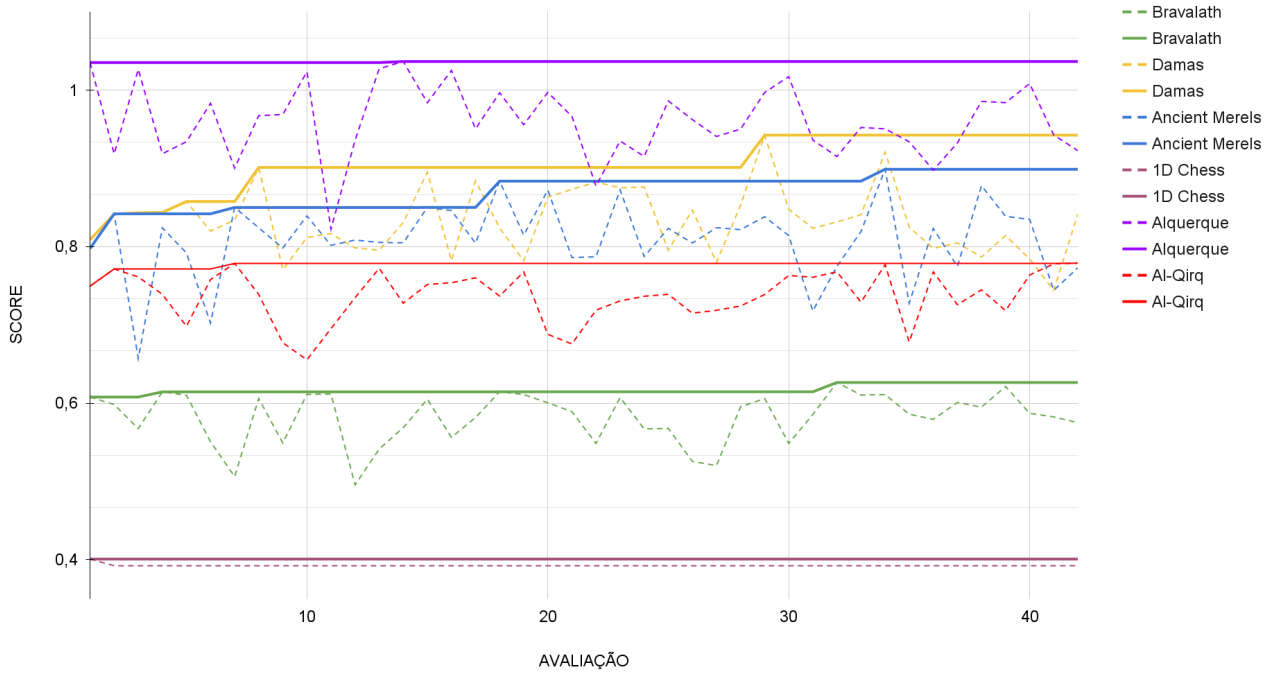
etapa, um conjunto de jogos de tabuleiro com as características definidas no sistema *Ludii* foi representado no sistema. Um conjunto de indivíduos humano foi apresentado a pares de jogos, onde cada indivíduo jogaria partidas contra um jogador não-humano e, ao final, indicaria qual dos jogos julgava mais interessante, gerando, ao final, uma lista de jogos ordenados pelo interesse humano. Na segunda etapa do experimento, foram realizadas partidas automáticas do conjunto de jogos utilizando jogadores não-humanos, onde as partidas foram analisadas com relação às métricas implementadas no sistema. Ao final, foram obtidos diferentes conjuntos de métricas e pesos, que, ao serem combinados, correlacionavam o *score* obtido na avaliação automática com a lista de interesse dos jogadores humanos.

Os pesos adotados para as métricas do Algoritmo de avaliação de cada jogo foram:

- *Duration*: -0.0907
- *Lead Change*: -0.2769
- *Completion*: 0.5941
- *Drama (average)*: 0.2167
- *Decisiveness*: 0.1311
- *Advantage P1*: 0.0394
- *Balance*: 0.1880
- *Drawishness*: 0.4634
- *Timeouts*: 0.4962
- *Decisiveness Moves*: -0.1288

Na figura 5.5 são apresentados os *scores* obtidos para as avaliações do Algoritmo de Busca Local para os jogos base para o número de 40 iterações. A linha tracejada representa o *score* de cada uma das variantes avaliadas após uma alteração em um dos seus parâmetros. A linha contínua representa o *score* da melhor solução encontrada até a determinada iteração do Algoritmo.

Figura 5.5 – Scores obtidos para diferentes jogos na Busca Local com 40 iterações. Para cada jogo há duas linhas. A linha tracejada indica o *score* atribuído à variação do jogo em cada iteração do Algoritmo de Busca Local, enquanto a linha contínua indica o melhor *score* encontrado até a determinada iteração.

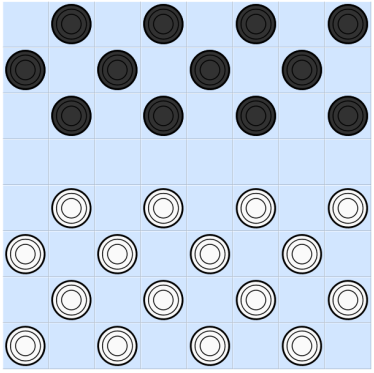


Fonte: o autor.

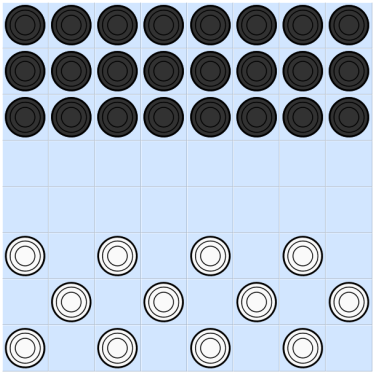
5.4 Jogos Variantes

Alguns dos jogos variantes obtidos durante o processo de busca são apresentados nas figuras a seguir.

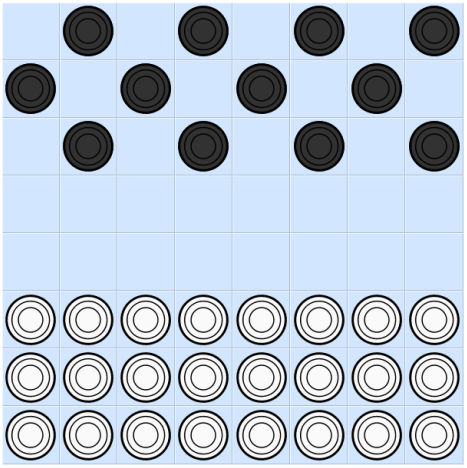
(a) Variante de damas. Nessa variação, uma linha de peças com o padrão da linha 2 foi adicionado à linha 4. Variação obtida na iteração 11 do Algoritmo de Busca Local. *Score*: 0,816718654492673.



(b) Variante de damas. Nessa variação, o segundo jogador possui peças em todas as colunas das 3 últimas linhas. Variação obtida na iteração 14 do Algoritmo de Busca Local. *Score*: 0,830790751288428.

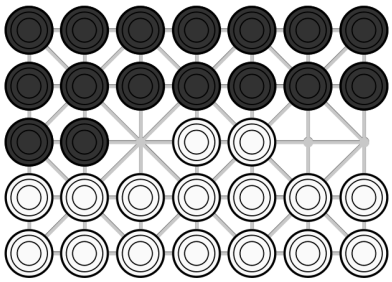


(c) Variante de damas. Nessa variação, o primeiro jogador possui peças em todas as colunas das 3 primeiras linhas. Variação obtida na iteração 30 do Algoritmo de Busca Local. *Score*: 0,847747213752694.

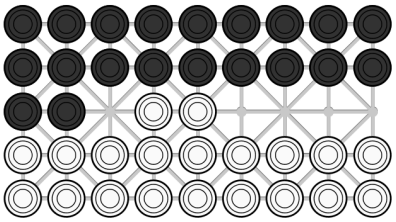


(d) Variação de *1D Chess*. Nessa variação, foram adicionadas 4 colunas à direita do tabuleiro original. Variação obtida na iteração 2 do Algoritmo de Busca Local. *Score*: 0,392190725374323.

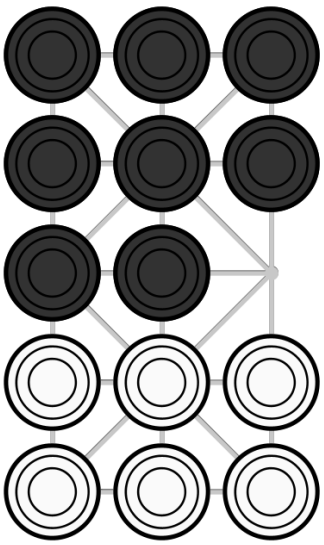
(e) Variação de *Alquerque*. Nessa variação, foram adicionadas 2 colunas à direita repetindo o padrão central. Variação obtida na iteração 22 do Algoritmo de Busca Local. *Score*: 0,877469369044145



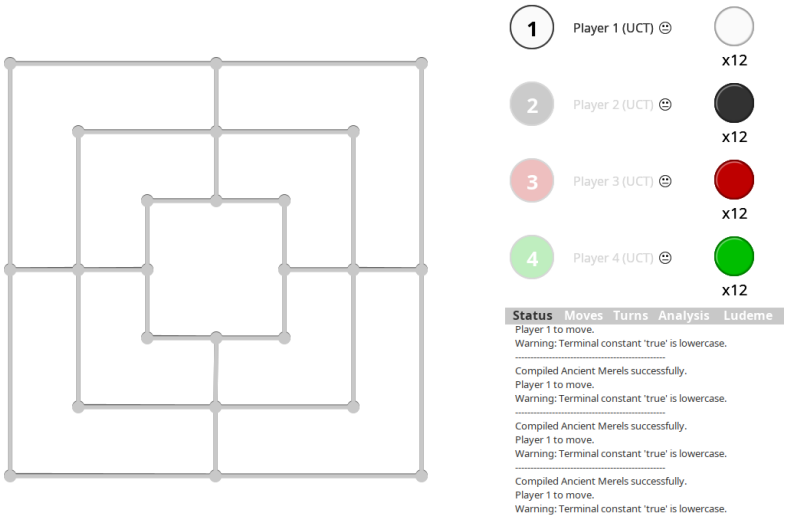
(f) Variação de *Alquerque*. Nessa variação, foram adicionadas 4 colunas à direita repetindo o padrão central. Variação obtida na iteração 12 do Algoritmo de Busca Local. *Score*: 0,936046144167318.



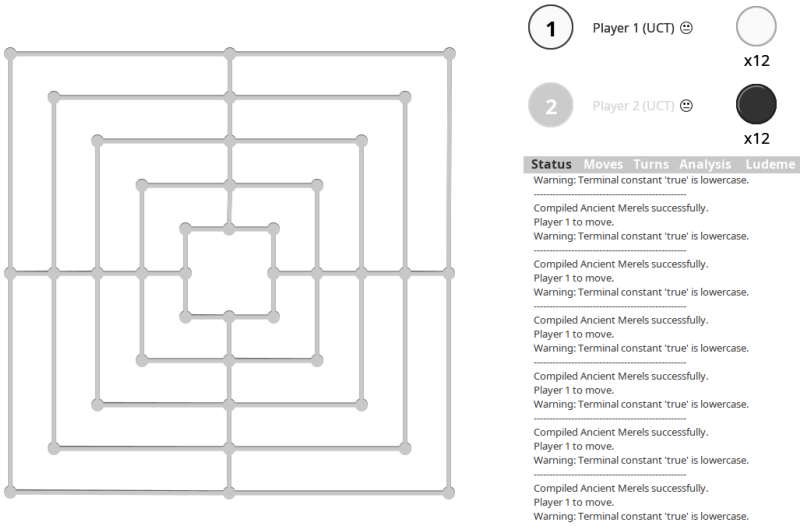
(g) Variação de *Alquerque*. Nessa variação, foram mantidas apenas as 3 primeiras colunas do jogo original. Variação obtida na iteraçao 2 do Algoritmo de Busca Local. *Score*: 0,918433997487116.



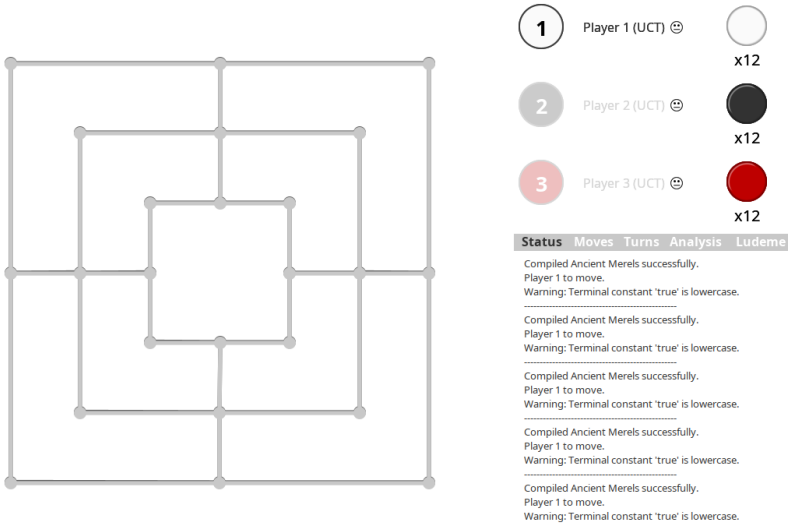
(h) Variação de *Ancient-Merels*. Nessa variação, foram adicionados 2 jogadores. Variação obtida na iteração 7 do Algoritmo de Busca Local. *Score*: 0,8500357378.



(i) Variação de *Ancient-Merels*. Nessa variação, foram adicionados 2 níveis exteriores. Variação obtida na iteração 19 do Algoritmo de Busca Local. *Score*: 0,8146126915.



(j) Variação de *Ancient-Merels*. Nessa variação, foi adicionado 1 jogador. Variação obtida na iteração 26 do Algoritmo de Busca Local. *Score*: 0,80458520052576.



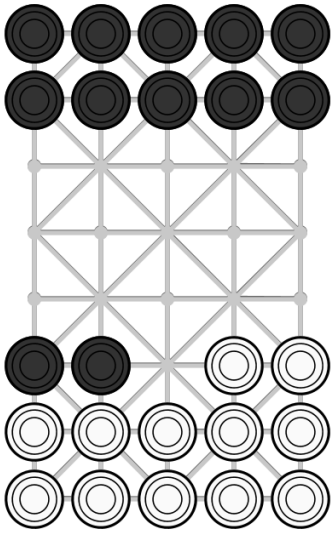
5.5 Avaliação de Resultados

A execução do Algoritmo de Busca Local (Algoritmo 1) com a alteração de parâmetros inteiros e *booleanos* presentes nas descrições dos jogos apresentou, na maioria das vezes, alterações em parâmetros numéricos das descrições, como quantidade de jogadores, dimensões do tabuleiro, número de peças, linhas de disposição de peças, e parâmetros similares.

Uma possível explicação a esse resultado se deve ao fato de que esses parâmetros aparecem em maior quantidade nas descrições. Em uma execução do Algoritmo de Busca Local com poucas iterações (para o experimento, um número $iteration\_range$ de iterações igual a 40), é dada uma menor oportunidade de variantes terem mecânicas de jogo alteradas, e nos casos em que essa alteração ocorre, a atribuição de um *score* menor a uma variante não permite que novas combinações com base na mecânica alterada emerjam.

O conjunto de jogos escolhidos como base para a execução do Algoritmo de Busca Local implementado também influenciou nos resultados obtidos. Pelo fato dos operadores de alteração de parâmetros se limitarem à alteração de apenas dois tipos (numéricos de *booleanos*), a forma como os jogos são descritos na GDL do sistema influencia nos valores em que é possível realizar a alteração. Assim, para um mesmo jogo descrito de maneira

Figura 5.7 – Abaixo, variação de Alquerque com o melhor *score*, com um valor de 1,0365, em contraste com um *score* da versão original do jogo de 1,0351. O fato de duas peças adversárias iniciarem em uma posição onde o adversário deve realizar uma captura possivelmente pode gerar situações desafiadoras nas partidas.



Fonte: o autor.

levemente distinta, a possibilidade de alteração nos seus parâmetros pode ser diferente.

Por fim, a função de avaliação do *score* da variação de um jogo, baseada em *playtests*, apresenta uma relação direta com a qualidade dos resultados obtidos. A busca por parâmetros do algoritmo UCT, apresentada na Seção 5.2, teve como objetivo realizar um balanço entre a qualidade da avaliação obtida e o tempo de execução da mesma, tendo em vista o número de iterações desejadas do Algoritmo de Busca Local. Com isso, caso sejam realizadas mais partidas em cada uma das avaliações, elevando o tempo de cada avaliação, o *score* obtido para cada variante tende a ser mais fiel a um valor específico que melhor o represente.

6 CONCLUSÃO

O presente trabalho teve como objetivo realizar a procura de variações de jogos de tabuleiro através de um algoritmo de Busca Local, cuja função de avaliação utiliza um conjunto de métricas (BROWNE, 2011d), e cujos *scores* são obtidos através de partidas automatizadas para cada variação obtida.

Com base nos resultados obtidos através dos experimentos, foi possível observar que a avaliação de um jogo com relação às métricas selecionadas através do sistema *Ludii* é uma tarefa que demanda grande esforço computacional e temporal. Para que a avaliação de um jogo seja a mais fiel possível ao que as suas regras realmente representam a quantidade de partidas que devem ser jogadas deve ser a maior possível, pois quanto maior o número de partidas jogadas, maior será o número de situações que ocorrem durante as partidas, aproximando o cálculo de cada uma das métricas a um valor específico.

Pelo fato da implementação abordada com relação à variação de elementos do jogo se limitar à alteração de valores numéricos e *booleanos*, as variações obtidas tenderam a apresentar jogos com características similares aos jogos base utilizados como objeto da busca. Entretanto, apesar da implementação realizada não ser complexa com relação à criação de novas variações, foram obtidas variantes que apresentam características interessantes quando comparadas aos jogos originais. Além disso, algumas variantes interessantes foram descobertas, apesar de terem um *score* menor atribuído durante o algoritmo de Busca Local (Algoritmo 1). Caso melhor exploradas, essas variações podem ter a capacidade de servir como base para a busca de outros jogos com características atraentes aos jogadores.

A partir da análise da abordagem e dos resultados obtidos nesse trabalho, alguns pontos que podem servir como continuação dessa linha de pesquisa incluem:

- A implementação de mecânicas de alteração de parâmetros mais complexos descritos na GDL do sistema para a execução do Algoritmo de Busca Local implementado nesse trabalho;
- A utilização de diferentes abordagens para a manipulação de parâmetros e estruturas da GDL, como de forma semelhante a (BROWNE, 2011a), o uso de Programação Genética no sistema *Ludii*;
- A validação dos *scores* dos jogos variantes gerados nesse trabalho através de partidas disputadas por participantes humanos. Assim, gerando uma correlação entre os valores obtidos através da implementação de avaliação utilizada nesse trabalho e os

valores obtidos em partidas reais, avaliar a efetividade das métricas utilizadas para a análise;

- Um estudo sobre diferentes conjuntos de métricas relevantes para categorias distintas de jogos, podendo indicar uma melhor combinação de métricas e jogos base para a execução do Algoritmo de Busca Local implementado nesse trabalho.

REFERÊNCIAS

BIRKHOFF, G. D. **Aesthetic Measure**. Cambridge, MA and London, England: Harvard University Press, 2013. ISBN 9780674734470. Disponível em: <https://doi.org/10.4159/harvard.9780674734470>.

BROWNE, C. **Evolutionary Game Design**. Springer London, 2011. (SpringerBriefs in Computer Science). ISBN 9781447121794. Disponível em: <https://books.google.com.br/books?id=tn2fE9USzZkC>.

BROWNE, C. Evolving games. In: ____. **Evolutionary Game Design**. London: Springer London, 2011. p. 37–49. ISBN 978-1-4471-2179-4. Disponível em: <https://doi.org/10.1007/978-1-4471-2179-4_5>.

BROWNE, C. The ludi system. In: ____. **Evolutionary Game Design**. London: Springer London, 2011. p. 11–21. ISBN 978-1-4471-2179-4. Disponível em: <https://doi.org/10.1007/978-1-4471-2179-4_3>.

BROWNE, C. Measuring games. In: ____. **Evolutionary Game Design**. London: Springer London, 2011. p. 23–36. ISBN 978-1-4471-2179-4. Disponível em: <https://doi.org/10.1007/978-1-4471-2179-4_4>.

BROWNE, C. Yavalath. In: ____. **Evolutionary Game Design**. London: Springer London, 2011. p. 75–85. ISBN 978-1-4471-2179-4. Disponível em: <https://doi.org/10.1007/978-1-4471-2179-4_7>.

COOK, M.; COLTON, S. Multi-faceted evolution of simple arcade games. In: **Computational Intelligence and Games (CIG'11)**. [S.l.]: 2011 IEEE Conference, 2011. p. 289–296.

KELLY, G.; MCCABE, H. A survey of procedural techniques for city generation. **The ITB Journal**, v. 7, p. 5, 2006.

KOCSIS, L.; SZEPESVÁRI, C. Bandit based monte-carlo planning. In: FÜRNKRANZ, J.; SCHEFFER, T.; SPILIOPOULOU, M. (Ed.). **Machine Learning: ECML 2006**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 282–293. ISBN 978-3-540-46056-5.

NELSON, M. J.; MATEAS, M. Towards automated game design. In: BASILI, R.; PAZIENZA, M. T. (Ed.). **AI*IA 2007: Artificial Intelligence and Human-Oriented Computing**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 626–637. ISBN 978-3-540-74782-6.

PIETTE, E.; BROWNE, C.; SOEMERS, D. J. N. J. **Ludii Game Logic Guide**. 1.3.12. ed. [S.l.], 2023.

SMELIK, R. M. et al. A survey of procedural methods for terrain modelling. In: **Proc. of the CASA Workshop on 3D Advanced Media In Gaming And Simulation (3AMIGAS**. [S.l.: s.n.], 2009.

STINY, G.; GIPS, J. **Algorithmic aesthetics**. Berkeley and Los Angeles: University Calif Press, 1978. ISBN 0-520-03467-8.

THOMPSON, J. M. **Defining the Abstract**. 2000. <http://www.thegamesjournal.com/articles/DefiningtheAbstract.shtml>. Accessed: 20-09-2022.

TOGELIUS, J.; SCHMIDHUBER, J. An experiment in automatic game design. In: **2008 IEEE Symposium On Computational Intelligence and Games**. [S.l.: s.n.], 2008. p. 111–118.

APÊNDICE A — TABELAS DAS MÉTRICAS DAS AVALIAÇÕES

Tabela A.1 – Métricas obtidas nas 10 primeiras avaliações da alteração do jogo *Bravalath*.

Métrica	**Iter. 1**	**Iter. 2**	**Iter. 3**	**Iter. 4**	**Iter. 5**	**Iter. 6**	**Iter. 7**	**Iter. 8**	**Iter. 9**	**Iter. 10**
Duration	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Lead Change	0.2228	0.2458	0.2592	0.1969	0.2421	0.2291	0.2686	0.2185	0.2206	0.2544
Completion	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Drama Average	0.1133	0.0983	0.1014	0.1105	0.1364	0.1133	0.1045	0.1327	0.1267	0.1346
Decisiveness	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
Advantage P1	0.4613	0.4667	0.4	0.5333	0.5333	0.3333	0.8	0.5333	0.7333	0.4667
Balance	0.9333	0.9333	0.8	0.9333	0.9333	0.6667	0.4	0.9333	0.5333	0.9333
Drawishness	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Timeouts	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Decisiveness Moves	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Tabela A.2 – Métricas obtidas nas 10 primeiras avaliações da alteração do jogo Damas.

Métrica	**Iter. 1**	**Iter. 2**	**Iter. 3**	**Iter. 4**	**Iter. 5**	**Iter. 6**	**Iter. 7**	**Iter. 8**	**Iter. 9**	**Iter. 10**
Duration	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Lead Change	0.1508	0.1544	0.1842	0.1712	0.1596	0.1336	0.1436	0.1585	0.1994	0.1499
Completion	0.7333	0.6	0.6	0.6667	0.5333	0.5333	0.7333	0.4	0.8	0.6
Drama Average	0.6882	0.6671	0.6095	0.6317	0.6325	0.6806	0.7033	0.6096	0.6096	0.6096
Decisiveness	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
Advantage P1	0.6	0.6	0.5333	0.5	0.4333	0.3	0.4667	0.5667	0.5667	0.5333
Balance	0.8	0.8	0.9333	1.0	0.8667	0.6	0.9333	0.8667	0.8667	0.9333
Drawishness	0.2667	0.4	0.4	0.3333	0.4667	0.4667	0.2667	0.6	0.2	0.4
Timeouts	0.2667	0.4	0.4	0.3333	0.4667	0.4667	0.2667	0.6	0.2	0.4
Decisiveness Moves	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Tabela A.3 – Métricas obtidas nas 10 primeiras avaliações da alteração do jogo *Ancient-Merels*.

Métrica	**Iter. 1**	**Iter. 2**	**Iter. 3**	**Iter. 4**	**Iter. 5**	**Iter. 6**	**Iter. 7**	**Iter. 8**	**Iter. 9**	**Iter. 10**
Duration	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Lead Change	0.0819	0.0569	0.0853	0.0472	0.0609	0.0804	0.0543	0.0592	0.0623	0.0806
Completion	0.6667	0.7333	0.8667	0.9333	0.8667	0.8	0.8667	0.6667	0.8	0.6667
Drama Average	0.6883	0.7207	0.6472	0.7346	0.7382	0.6966	0.7185	0.7138	0.7011	0.6856
Decisiveness	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
Advantage P1	0.3	0.6	0.8667	0.5	0.7	0.8667	0.6333	0.4667	0.7	0.5
Balance	0.6	0.8	0.2667	1.0	0.6	0.6	0.7333	0.9333	0.8667	0.8667
Drawishness	0.3333	0.2667	0.1333	0.0667	0.2	0.1333	0.2667	0.2	0.2	0.2
Timeouts	0.3333	0.2667	0.0667	0.0667	0.2	0.2	0.2667	0.2667	0.4	0.2
Decisiveness Moves	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Tabela A.4 – Métricas obtidas nas 10 primeiras avaliações da alteração do jogo *1D Chess*.

Métrica	Iter. 1	Iter. 2	Iter. 3	Iter. 4	Iter. 5	Iter. 6	Iter. 7	Iter. 8	Iter. 9	Iter. 10
Duration	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Lead Change	0.5367	0.5556	0.5556	0.5556	0.5556	0.5556	0.5556	0.5556	0.5556	0.5556
Completion	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Drama Average	0.2716	0.2562	0.2562	0.2562	0.2562	0.2562	0.2562	0.2562	0.2562	0.2562
Decisiveness	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
Advantage P1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Balance	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Drawishness	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Timeouts	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Decisiveness Moves	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Tabela A.5 – Métricas obtidas nas 10 primeiras avaliações da alteração do jogo *Alquerque*.

Métrica	Iter. 1	Iter. 2	Iter. 3	Iter. 4	Iter. 5	Iter. 6	Iter. 7	Iter. 8	Iter. 9	Iter. 10
Duration	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Lead Change	0.0692	0.0945	0.0741	0.0696	0.0964	0.0886	0.0906	0.0661	0.0898	0.0825
Completion	0.4	0.4	0.2667	0.3333	0.2667	0.2667	0.2667	0.2667	0.1333	0.4
Drama Average	0.7941	0.7880	0.7523	0.8214	0.7298	0.7629	0.7429	0.7942	0.7548	0.7729
Decisiveness	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
Advantage P1	0.4333	0.4333	0.4333	0.4000	0.3667	0.4333	0.5000	0.4333	0.4333	0.4667
Balance	0.8667	0.8667	0.8667	0.8000	0.7333	0.8667	1.0	0.8667	0.8667	0.9333
Drawishness	0.6	0.6	0.7333	0.6667	0.7333	0.7333	0.7333	0.8667	0.8667	0.6
Timeouts	0.6	0.6	0.7333	0.6667	0.7333	0.7333	0.7333	0.8667	0.8667	0.6
Decisiveness Moves	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Tabela A.6 – Métricas obtidas nas 10 primeiras avaliações da alteração do jogo *Al-Qirq*.

Métrica	Iter. 1	Iter. 2	Iter. 3	Iter. 4	Iter. 5	Iter. 6	Iter. 7	Iter. 8	Iter. 9	Iter. 10
Duration	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Lead Change	0.0934	0.0866	0.0937	0.1026	0.0793	0.0697	0.0847	0.0881	0.0793	0.1145
Completion	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Drama Average	0.6933	0.6822	0.6444	0.6567	0.7371	0.7081	0.6757	0.6591	0.6620	0.6033
Decisiveness	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
Advantage P1	0.6	0.5333	0.5333	0.6	0.8	0.6	0.5333	0.6	0.8	0.4
Balance	0.8	0.9333	0.9333	0.8	0.4	0.8	0.9333	0.8	0.6667	0.8
Drawishness	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Timeouts	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Decisiveness Moves	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0