Aprimorando o Processo de Aprendizagem e Alocação de Agentes Inteligentes em Plataformas Multiagentes - Aplicação no Domínio do Jogo de Damas

Valquíria Aparecida Rosa Duarte

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2016

Valquíria Aparecida Rosa Duarte

Aprimorando o Processo de Aprendizagem e Alocação de Agentes Inteligentes em Plataformas Multiagentes - Aplicação no Domínio do Jogo de Damas

Tese de doutorado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Área de concentração: Inteligência Artificial

Orientador: Profa Dra Rita Maria da Silva Julia

Uberlândia
2016

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

D812a
2016

Duarte, Valquíria Aparecida Rosa, 1984-
Aprimorando o processo de aprendizagem e alocação de agentes inteligentes em plataformas multiagentes : aplicação no domínio do jogo de damas / Valquíria Aparecida Rosa Duarte. - 2016.
146 f. : il.

Orientadora: Rita Maria da Silva Julia.
Tese (doutorado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação.
Inclui bibliografia.

1. Computação - Teses. 2. Aprendizado do computador - Teses. 3. Redes neurais (Computação) - Teses. 4. Mineração de dados (Computação) - Teses. I. Julia, Rita Maria da Silva. II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU: 681.3

UNIVERSIDADE FEDERAL DE UBERLÂNDIA – UFU
FACULDADE DE COMPUTAÇÃO – FACOM
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO – PPGCO

Os abaixo assinados, por meio deste, certificam que leram e recomendam para a Faculdade de Computação a aceitação da tese intitulada **"Aprimorando o Processo de Aprendizagem e Alocação de Agentes Inteligentes em Plataformas Multiagentes - Aplicação no Domínio do Jogo de Damas"** por **"Valquíria Aparecida Rosa Duarte"** como parte dos requisitos exigidos para a obtenção do título de Doutora em Ciência da Computação.

Uberlândia, _____ de _____ de _____

Orientadora: _____
Profa. Dra. Rita Maria da Silva Julia
Universidade Federal de Uberlândia

Banca Examinadora:

_____
Prof. Dr. George Darmiton da Cunha Cavalcanti
Universidade Federal de Pernambuco

_____
Prof. Dr. Rogério Andrade Flauzino
Universidade de São Paulo

_____
Prof. Dr. Igor Santos Peretta
Universidade Federal de Uberlândia

_____
Prof. Dr. Laurence Rodrigues do Amaral
Universidade Federal de Uberlândia

À minha filha Clara e ao meu esposo Bruno, com todo meu amor e carinho.

Agradecimentos

Agradeço a Deus, pelo dom da vida e a minha filha Clara por encher de amor a minha vida.

Ao meu esposo Bruno, pelo incentivo, compreensão e parceria diária.

À minha família por me ensinarem que o amor, o respeito, a dignidade e a honestidade são os principais valores que uma pessoa deve cultivar durante a vida. Em especial, minha mãe e minha irmã por terem sido a base para a construção de todos os meus ideais.

Aos meus velhos e novos amigos, que independente das adversidades da vida sempre foram e serão meus amigos.

À minha orientadora Profa. Dra Rita Julia, pela confiança, disponibilidade, dedicação, e profissionalismo.

Aos meus colegas de laboratório da UFU, pelas boas conversas, risadas e incentivos durante todos esses anos. Em especial ao Henrique e a Lídia, meus amigos de laboratório, pela fiel parceria e amizade cultivada durante esses anos.

Ao Erisvaldo, secretário da Pós-Graduação, por sua eficiência, disponibilidade e amizade.

A FAPEMIG, pelo apoio financeiro concedido para a realização deste trabalho.

A todos os demais, muito obrigada!

A sabedoria dos homens é proporcional não à sua experiência mas à sua capacidade de adquirir experiência.
(George Bernard Shaw)

Resumo

Um dos requisitos fundamentais para que um sistema multiagente não supervisionado atinja seus objetivos é que os agentes que o compõem possuam habilidades específicas e complementares que lhe permitam atuar como especialistas nos ambientes em que foram treinados. A representação adequada desses ambientes é fundamental para o aprendizado e para a boa performance dos agentes, principalmente quando esses atuam em ambientes competitivos que possuem elevado espaço de estados. Do mesmo modo, as decisões do sistema multiagente em alocar os agentes adequados para atuarem em determinadas situações que ocorrem nesses ambientes são cruciais para que este atinja, com êxito, seus objetivos. Nesse sentido, o presente trabalho apresenta três novas abordagens para otimizar o desempenho de sistemas multiagentes, as quais aprimoram: a arquitetura e o processo de aprendizagem dos agentes que compõem o sistema multiagente, a representação das informações relevantes dos ambientes de atuação desses agentes, assim como o processo de alocação dos agentes adequados para atuarem nas distintas situações que ocorrem nesses ambientes. Devido à sua complexidade espacial e técnica, o jogo Damas foi utilizado como ambiente de desenvolvimento e avaliação dessas abordagens, as quais foram implementadas na arquitetura do jogador automático MP-Draughts. Tal jogador corresponde a um sistema multiagente não supervisionado composto por agentes jogadores especialistas em fases distintas de um jogo. Para a implementação das abordagens propostas na arquitetura do MP-Draughts, foi adotada a seguinte sequência de trabalho: inicialmente, foi desenvolvida uma rede neural adaptativa, a ASONDE, que foi utilizada na arquitetura do MP-Draughts para definir os perfis (*clusters*) de conhecimentos necessários para representar a fase de final de jogo, nos quais os agentes especialistas devem ser treinados. Na sequência, foi implementada uma abordagem de seleção automática de características baseada na mineração de padrões frequentes, a qual extrai as mais adequadas para representar os diferentes ambientes (tabuleiros) que podem ocorrer durante a atuação do multiagente. Finalmente, foi desenvolvido um método de alocação de agentes que combina redes neurais artificiais e regras de exceção, as quais em conjunto, são responsáveis por indicar os agentes mais adequados para atuarem nas distintas situações

de um jogo. Os resultados parciais obtidos da implementação de cada abordagem, assim como o resultado final que aplica todas elas na arquitetura no MP-Draughts, confirmam que as mesmas foram eficientes para tratar os problemas para os quais foram propostas, além de contribuírem para o desempenho geral do sistema multiagente.

Palavras-chave: Aprendizagem de máquina; Sistemas multiagentes; Redes neurais adaptativas; Representação de ambiente; Mineração de padrões frequentes; Regras de exceção; Jogo de Damas.

Abstract

One of the fundamental requirements for a unsupervised multiagent system to reach its objectives is that the agents that make up the system possess specific and complementary abilities, which allow them to act as specialists in the environments where they were trained. The adequate representation of these environments is fundamental to both the learning and to the good performance on the part of the agents, mainly when these act in competitive environments that possess an elevated state space. Likewise, the decisions from multiagent systems, through their allocation of adequate agents into particular situations that occur in these environments, are crucial in order that these successfully reach their objectives. In this sense, the present work presents three new approaches to optimize the performance of multiagent systems, which improves: the architecture and the learning process of the agents that make up the multiagent system; the representation of relevant information of the environments where these agents perform, as well as the process of allocating the adequate agent for performing in distinct situations that occur in these environments. Due to the spatial and technical complexity, the game of Checkers was used as the developmental and evaluative environment for these approaches, which were implemented onto the automatic player MP-Draughts. This player corresponds to a unsupervised multiagent system composed of specialist player agents in distinct phases of a game. In order to implement the proposed approaches onto the MP-Draughts architecture, the following work sequence was adopted: initially, an adaptive neural network was developed, ASONDE, which was used in the MP-Draughts architecture to define the knowledge profiles (*clusters*) necessary for representing the endgame phase, on which the specialist agents should be trained. Following on, an automatic features selection approach based on the frequent pattern mining was implemented, which extracts the most adequate features to represent the different environments (boards) that can occur during the performance of the multi-agent. Finally, a method for the allocation of agents was developed, which combined clustering artificial neural networks and exception rules, which together are responsible for indicating the most suitable agents to act in the different situations of a game. The partial results obtained from the implementation of

each approach, as well as the final result, which applies all these into the MP-Draughts architecture, confirm that these were efficient in dealing with the problems for which they were proposed, in addition to contributing to the general performance of the multi-agent system.

Keywords: Machine learning; Multi-agent systems; Adaptive neural networks; Representation of environments; Frequent pattern mining; Exception rules; Checkers game.

Lista de ilustrações

Lista de tabelas

Lista de algoritmos

Lista de siglas

AG Algoritmo Genético

AI Aprofundamento Iterativo

AM Aprendizagem de Máquina

AR Aprendizagem por Reforço

ARI *Adjusted Rand Index*

ART *Adaptive Resonance Theory*

ASONDE *Adaptation of Self-Organizing Novelty Detection*

BDs Bases de Dados

BD Base de Dados

BMU *Best Matching Unit*

EGA *EndGame Agent*

FIS *Mamdani Fuzzy Inference Engine*

GWR *Growing When Required*

IA Inteligência Artificial

IIGA *Initial/ Intermidiate Game Agent*

KSOM Mapas Auto Organizáveis de Kohonen

KSOM-Cos Rede de Kohonen baseada em similaridade cosseno

KSOM-DE Rede de Kohonen baseada em distância Euclidiana

MP-Draughts *MultiPhase-Draughts*

MLP *MultiLayer Perceptron*

PDN *Portable Draughts Notation*

PG Programação Genética

RN Redes Neurais

SMA Sistemas Multiagentes

SONDE *Self-Organizing Novelty Detection*

TD *Temporal Differences*

TT Tabela de Transposição

YBWC *Young Brothers Wait Concept*

Sumário

Capítulo 1

Introdução

Nos últimos anos, a Inteligência Artificial (IA) têm sido aplicada a várias áreas de conhecimento para apoiar na resolução de variados tipos de problemas, e a Aprendizagem de Máquina (AM) é um dos ramos da IA que mais têm se destacado. A AM consiste no desenvolvimento e utilização de técnicas computacionais capazes de extrair conhecimento a partir de amostras de dados ou da observação do ambiente [5]. As pesquisas em AM têm se concentrado na investigação de problemas cuja resolução é viável utilizando técnicas de aprendizagem supervisionada, não supervisionada, ou uma mistura de ambas (semi-supervisionada). O presente trabalho se baseia essencialmente nas técnicas não supervisionadas de aprendizagem. Tais técnicas são aptas a resolver problemas cujas soluções podem ser, inclusive, desconhecidas, problemas de natureza complexa em que a solução deve ser investigada em ambientes com elevado espaço de estados e, ainda, problemas em que se deseja encontrar boas soluções sem depender de conhecimento de especialistas, conhecimento este que nem sempre está disponível [4], [6].

Outro ramo da IA que vem sendo continuamente pesquisado na literatura é o campo dos Sistemas Multiagentes (SMA), os quais coordenam um comportamento inteligente entre os agentes de um sistema, de modo que todos trabalhem conjunta ou competitivamente, para a resolução de problemas [7]. Tais sistemas possuem características que viabilizam a resolução de problemas complexos e de natureza descentralizada [8]. Em [9] os autores evidenciam algumas dessas características, que são: maior eficiência na resolução de problemas, pois existem vários agentes atuando para sua solução; maior flexibilidade de adaptação ao ambiente por possuirem agentes com diferentes habilidades colaborando na resolução do problema; possibilita a distribuição espacial do problema entre os diversos agentes do sistema. Dentre os vários problemas nos quais os SMAs têm sido aplicados, este trabalho destaca sua aplicação no domínio de jogos.

Atualmente, os jogos representam um dos ramos de investimentos mais promissores no mundo dos negócios, movimentando bilhões de dólares por ano. Jogos não são mais utilizados como meros ambientes de entretenimento, mas também como ambiente de apoio ao ensino e aprendizagem em salas de aula, ambiente de treinamento de mão de obra

especializada, simulação industrial, ambiental e de negócios, etc [10], [11], [12]. A teoria (inteligência) existente por trás da resolução de um jogo pode ser abstraída e aplicada na resolução de uma variedade de problemas práticos do dia-a-dia, o que fez com que a Teoria dos Jogos tornasse um importante ramo da Matemática e da Computação, especialmente após da publicação do livro "*The Theory of Games and Economic Behavior*" [13]. Neste trabalho, os autores averiguam estratégias de ações apropriadas em situações nas quais os resultados não dependem apenas do comportamento do próprio agente, mas também do ambiente de atuação e das estratégias dos outros agentes que interagem com ele. Inicialmente, tal teoria foi aplicada para a resolução de problemas na economia e, desde então, vêm sendo utilizada para resolução de problemas em diferentes áreas, tais como: biologia, administração, engenharia, computação, entre outros.

Desde o início das pesquisas envolvendo jogos têm-se buscado implementar não apenas jogos, mas também jogadores automáticos inteligentes que sejam capazes de interagir e competir em alto nível contra jogadores humanos. O precursor na criação de jogadores automáticos inteligentes e na aplicação de técnicas de AM no domínio de jogos foi Arthur L. Samuel, o qual propôs, em 1946, um jogador automático de Damas cujo objetivo era vencer o campeão mundial humano da época. Em seus trabalhos [14], [15], Samuel propôs várias técnicas relacionadas a AM, tais como: Aprendizagem por Reforço (AR), Diferenciação Temporal (do inglês *Temporal Differences* (TD) - TD(λ)), algoritmo de busca Alfa-Beta, treinamento por *self-play* com clonagem, entre outros. Os resultados obtidos destes trabalhos constituíram a base da AM como ciência e abriu um significativo campo de pesquisa dessas técnicas em diversos domínios [16]. O jogador automático de Damas proposto por ele é considerado um marco nas pesquisas sobre AM [17] e ainda é tido como fonte de inspiração para diversas pesquisas na área de aprendizagem automática e de jogos. Samuel considerou o domínio do jogo de Damas perfeito para o estudo de técnicas inteligentes, uma vez que a complexidade de resolução desse jogo se assemelha à complexidade de resolução de problemas práticos da vida real. De acordo com ele, muitas das complicações que surgem nos problemas da vida real podem ser modeladas no domínio de um jogo, permitindo que os pesquisadores foquem nos problemas de aprendizagem propriamente dito. Outro trabalho dessa época que obteve bons resultados na criação de jogadores automáticos inteligentes foi o jogador de Xadrez proposto por Claude Shannon [18]. Neste trabalho o autor define dois tipos de estratégias de jogo que ainda são muito utilizadas, tanto para a resolução de jogos quanto para outros problemas, que são: o uso de "força bruta" (ou busca exaustiva) para explorar todo o espaço de estados em busca de uma solução e o uso de conhecimento específico do domínio para examinar apenas uma parte desse espaço em busca da solução.

Em se tratando de resolução de jogos, em Herik *et. al.* [2] é apresentado uma análise das principais características de um jogo que influenciam na sua complexidade de resolução, que são: complexidade de espaço de estados e fator de ramificação da árvore de jogo. A

primeira é definida pelo conjunto de todos os estados possíveis, derivados de movimentos legais, que possam ser executados sucessivamente no jogo a partir de seu estado inicial. A segunda é definida pelo número de folhas obtidas na solução do jogo a partir do estado corrente. A principal conclusão da análise feita por Herik é que a complexidade do espaço de estados é mais relevante para determinar o grau de dificuldade de um jogo do que o seu fator de ramificação. A Tabela 1 apresenta a complexidade de espaço de estados e o fator de ramificação de alguns jogos, os quais possuem nível de complexidade relevante.

Tabela 1 – Complexidade do espaço de estados e fator de ramificação de alguns jogos [2], [3], [4].

Jogo	**Fator de Ramificação**	**Espaço de Estados**
Xadrez	30 - 40	10^{46}
Damas	8 - 10	10^{21}
Gamão	± 420	10^{20}
Othello	± 5	$< 10^{28}$
Go 19x19	± 360	10^{172}
Shogi	± 92	10^{71}

Com base nas considerações de Samuel, na conclusão de Herik e na complexidade de resolução do jogo de Damas, é possível concluir que tal jogo é, sem dúvida, um excelente laboratório para investigação e evolução de técnicas computacionais relacionadas a AM e a IA. A literatura atual de jogadores automáticos de Damas apresenta vários jogadores que se baseiam nas mais diversas técnicas inteligentes, os quais podem ser divididos em 2 grupos: aqueles que contam com conhecimento de especialistas para aprender e aprimorar suas habilidades de jogo (supervisão) e, os que aprendem a jogar com o mínimo possível de conhecimento externo (sem supervisão). Os melhores jogadores automáticos de Damas da atualidade pertencem ao grupo dos que contam com algum tipo de supervisão, os jogadores supervisionados.

Dentre os jogadores supervisionados, pode-se destacar o Chinook [19], o atual campeão mundial homem-máquina. Em 2007, a equipe proponente deste jogador anunciou que o jogo de Damas estava fracamente resolvido (*weakly solved*), ou seja, partindo do estado inicial do jogo existe uma prova computacional de que, ou o jogo é um empate ou o Chinook vence o jogo [20]. Isso quer dizer que, caso o jogo se inicie nas condições normais (tabuleiro 8×8 com todas as peças nas posições iniciais do jogo) e seu oponente efetue apenas jogadas perfeitas, o jogo termina em empate. Caso as jogadas do oponente não sejam perfeitas, o Chinook sempre vence o jogo. A resolução do jogo obtida por tal equipe foi baseada nos seguintes pilares: essencialmente supervisionada, visto que possui gigantescas Bases de Dados (BDs) onde estão mapeadas todas as jogadas perfeitas que devem ser executadas pelo Chinook em resposta a qualquer jogada do oponente; computacionalmente custosa, pois além de consumir muito espaço para o armazenamento dessas

BDs, precisa processar milhões de registros por segundo para encontrar a jogada perfeita; contou com o auxilio do maior especialista do jogo de todos os tempos, Mario Tinsley, e com dezenas de computadores trabalhando continuamente na resolução do jogo [20]. Vale destacar que a resolução do jogo só foi possível após aproximadamente 18 anos de pesquisas.

Apesar de o jogo de Damas ter sido fracamente resolvido e de o Chinook ser o jogador de Damas mais forte da literatura [20], existem lacunas provenientes desta resolução que permitem que o jogo de Damas ainda se configure como um excelente domínio para a investigação de técnicas de AM e IA. Dentre essas lacunas, pode-se destacar: dificuldade de adaptação da solução do jogo para a solução de outros tipos de jogos ou problemas; possibilidade de redução do investimento computacional para a solução do problema, visto que um investimento elevado nem sempre é conveniente; tentar evitar, ou pelo menos diminuir, a necessidade de conhecimento especializado do domínio, o que nem sempre é possível e; encontrar boas solução em tempo hábil. Outro fato que comprova que o jogo de Damas continua sendo um excelente laboratório de pesquisas, mesmo após a resolução do jogo, e que as lacunas existentes nesta solução podem ser tratadas por técnicas de AM, é que vários jogadores automáticos de Damas foram recentemente publicados na literatura, mesmo após o anúncio que o jogo estava fracamente resolvido [6,21–32].

No contexto de jogadores automáticos de Damas não supervisionados, a equipe na qual a presente pesquisa se insere possui uma série de bons jogadores, tais como o VisionDraughts [21], D-VisionDraughts [24], LS-VisionDraughts [29] e o *MultiPhase-Draughts* (MP-Draughts) [22], [23]. Dentre esses jogadores, o MP-Draughts é o único desenvolvido sobre uma plataforma multiagente, cujos agentes são especializados em fases distintas do jogo. Tal jogador foi criado com o objetivo de aprimorar as habilidades dos demais jogadores da equipe na fase de final do jogo, fase esta em que os demais não eram habilidosos o suficiente para aproveitar sua vantagem, avançar contra o oponente e vencer o jogo. Apesar de esse jogador ter obtido bons resultados em jogos e ter cumprido com o objetivo de melhorar as habilidades de um jogador não supervisionado nas fases finais de um jogo, existem algumas limitações em sua arquitetura que prejudicam seu desempenho global. Dentre essas limitações, destacam-se: a definição empírica da arquitetura multiagente para representar a fase de final de jogo; representação generalista de todos os ambientes que ocorrem durante um jogo, independente da fase do jogo; deficiência na alocação dos agentes adequados para atuar em determinadas situações de um jogo. Nesse sentido, na intenção de continuamente contribuir para a evolução das técnicas de aprendizagem não supervisionadas e de aprimorar a arquitetura no multiagente MP-Draughts, este trabalho visa propor técnicas de AM não supervisionadas capazes de resolver as fragilidades dessa arquitetura. As técnicas propostas serão desenvolvidas de modo que possam ser aplicadas a outros domínios de problemas.

1.1 Motivação e Justificativa

A eficiência dos jogadores automáticos supervisionados está condicionada ao conhecimento especializado a eles fornecidos, seja através do acesso à Base de Dados (BD) onde todas as melhores jogadas estão mapeadas, seja por meio de ajustes manuais das funções de avaliação que norteiam o aprendizado e as ações do jogador. Por outro lado, a eficiência dos jogadores automáticos não supervisionados está relacionada a sua capacidade de abstrair conhecimentos do domínio que sejam suficientes para direcioná-los em busca de boas soluções. Por exemplo, em [33] é apresentado um estudo que mostra que grandes mestres do Xadrez raramente buscam mais do que 100 ramificações para escolher um movimento. Por que, então, esses jogadores conseguem obter tão bons desempenhos nos jogos? Tecnicamente falando, a inteligência dos humanos está na sua habilidade de realizar "podas" na árvore do jogo, de modo que apenas as posições consideradas relevantes sejam analisadas. Isso só é possível porque os humanos são capazes de utilizar diversas formas de aprendizagem e raciocínio, tais como indução, dedução, analogia, probabilidade, etc. Formas essas, que podem ser adaptadas e eficientemente implementadas na arquitetura de qualquer jogador automático não supervisionado, dispensando grande parte da necessidade de processamento e armazenamento vinculadas às técnicas supervisionadas utilizadas nesse domínio.

Além disso, em [34], os autores argumentam que, apesar de jogos como Damas e Xadrez terem obtido excelentes jogadores baseados em técnicas computacionais fortemente supervisionadas, o mesmo não seria possível de ser obtido em jogos como Go e Shogi, em que o espaço de estados é consideravelmente maior (conforme Tabela 1). A ineficiência na utilização de técnicas supervisionadas para resolver jogos com espaço de busca muito elevado, mostra que esse tipo de abordagem (busca exaustiva) não representa uma boa solução para tais problemas. Nesse sentido, uma alternativa pode ser utilizar técnicas inteligentes não supervisionadas, ou semi-supervisionadas, que sejam capazes de reduzir o espaço de busca e proporcionar a obtenção de uma boa solução. A AM não supervisionada tanto se configura como boa alternativa para apoiar a solução desse tipo de problema que, por exemplo, o jogador automático de Go *AlphaGo* [35] (recentemente proposto pela equipe de pesquisas do Google), o qual é baseado em redes neurais profundas (do inglês *Deep Neural Networks*) e AR, conseguiu vencer o campeão europeu do jogo. Tal resultado até então não havia sido alcançado por nenhum jogador automático de GO baseado apenas em técnicas supervisionadas.

Baseado no exposto até aqui e seguindo a vertente de explorar técnicas de AM não supervisionadas que possam ser utilizadas na resolução de problemas cujo espaço de solução seja elevado, o presente trabalho propõe novas técnicas de aprendizagem não supervisionadas que sejam eficientes em solucionar os problemas arquiteturais do multiagente jogador MP-Draughts. A versão preliminar do MP-Draughts, proposta em [22] e [23], corresponde a um sistema multiagente jogador de Damas não supervisionado cuja arquitetura é base-

ada em Redes Neurais (RN) de Kohonen (Mapas Auto Organizáveis de Kohonen (KSOM)) e em *perceptron* multicamadas (do inglês *MultiLayer Perceptron* (MLP)). Tal versão é composta por 26 agentes especialistas em distintas fases do jogo. Um agente - *Initial/ Intermidiate Game Agent* (IIGA) - é especialista em fases iniciais e intermediárias de jogo e os outros 25 agentes - *EndGame Agent* (EGA) - são especialistas em fases de final de jogo. Cada EGA foi treinado para ser especialista num determinado perfil de final de jogo, sendo que cada perfil representa um *cluster*. Tais *clusters* foram extraídos por uma Rede de Kohonen baseada em distância Euclidiana (KSOM-DE), de uma BD contendo cerca de 4000 estados de tabuleiros de final de jogo. A quantidade de 25 *clusters* e, consequentemente 25 EGAs, foi empiricamente definida. Cada agente que compõe o MP-Draughts corresponde a uma MLP que utiliza o algoritmo de busca Alfa-Beta, combinado com Tabela de Transposição (TT) e Aprofundamento Iterativo (AI) para escolher o melhor movimento em função do estado corrente do jogo. Além disso, o jogador utiliza o método de AR por TD(λ) aliado à estratégia de treinamento por *self-play* com clonagem, como ferramentas para atualizar os pesos da MLP. O tabuleiro do jogo é representado por um conjunto de 12 funções (características) que descrevem aspectos relevantes do jogo de Damas, conhecidas como *NetFeatureMap* [14]. A dinâmica de jogo adotada pelo MP-Draughts se configura pelo seguinte percurso: o IIGA é o agente responsável por iniciar o jogo e o conduzir até a fase de final de jogo; neste ponto, o estado corrente do jogo é submetido a KSOM-DE que seleciona, dentre os 25 EGAs, àquele que deve conduzir o jogo até o final. A KSOM-DE é a mesma RN que minerou os *clusters* nos quais os EGAs foram treinados.

Apesar do bom desempenho obtido pelo MP-Draughts, este jogador apresenta várias limitações em sua arquitetura, que vão desde limitações estruturais até comportamentais [22], [23], a saber: sua arquitetura de final de jogo foi empiricamente pré-definida pela KSOM-DE após uma série de testes variando a quantidade de *clusters* gerados pela RN, uma vez que seus autores não são especialistas no jogo e não tinham conhecimento suficiente para tomar tal decisão; as características utilizadas para representar os tabuleiros do jogo, além de terem sido escolhidas manualmente de um conjunto maior contendo 26 delas [14], são as mesmas utilizadas para representar todas as fases do jogo, de modo que todos os agentes (tanto o IIGA quanto todos os EGAs) veem o jogo sob a mesma perspectiva; análises feitas sobre o comportamento dos EGAs na fase de final de jogo mostraram que em algumas situações excepcionais o EGA indicado pela KSOM-DE para atuar no jogo não desempenhava adequadamente seu papel de modo a conduzir o jogo para uma derrota. Estas análises mostraram também que outros EGAs, diferente daquele definido pela RN, eram capazes de obter melhores resultados (vitórias) nessas mesmas situações. Diante disso, a motivação que impulsiona a realização deste trabalho é a de propor técnicas de AM não supervisionadas capazes de resolver as limitações do MP-Draughts de modo a melhorar seu desempenho geral, além de contribuir para a evolução

da qualidade dos jogadores automáticos de Damas não supervisionados e para a evolução das técnicas de AM não supervisionada. Estas técnicas têm por objetivo: melhorar o processo de agrupamento que define os perfis adequados para representar a fase de final de jogo, os quais são responsáveis por definir a arquitetura de final de jogo do multiagente; otimizar a representação dos ambientes de atuação dos agentes, focando apenas nas informações que sejam de fato relevantes para a tomada de decisão de cada agente em cada ambiente e; definir um processo de alocação eficiente capaz de indicar adequadamente, independente da situação do jogo, os agentes que devem atuar no mesmo. Tais técnicas foram desenvolvidas de maneira que possam ser adaptadas e aplicadas a outros domínios de problemas.

1.2 Objetivos da Pesquisa

Considerando as questões apresentadas na Seção 1.1, o objetivo geral deste trabalho é explorar e resolver as limitações do sistema multiagente MP-Draughts utilizando técnicas de AM não supervisionadas. Pretende-se ao final deste trabalho que a arquitetura do multiagente seja a mais adequada possível para representar a fase de final de jogo, permitindo que cada agente tenha condições de reconhecer e avaliar apenas informações relevantes do ambiente que contribuam para sua tomada de decisão, os quais devem ser adequadamente alocados para cada situação de jogo.

No intuito de cumprir o que se propõe, os seguintes objetivos específicos devem ser alcançados:

1. Propor uma alternativa de RNs adaptativas para agrupamento de dados que seja capaz de, dinamicamente, identificar e agrupar todos os perfis de conhecimentos inerentes a fase de final de jogo, uma vez que não existe nenhum conhecimento a priori desta informação. A partir dos perfis identificados pela RN, é que devem ser definidos quantos agentes de final de jogo devem compor a arquitetura do MP-Draughts;

2. Definir quais são as características mais apropriadas para representar os tabuleiros que ocorrem nas diferentes fases do jogo, de modo que cada agente possa visualizar e avaliar apenas o que de fato é relevante para sua atuação (tomada de decisão) no ambiente. Apesar de existir outra técnica que propõe características adequadas para representar o jogo de Damas, o presente trabalho visa propor uma alternativa mais eficiente que possa ser aplicada, em tempo hábil, nas diferentes fases e perfis do jogo;

3. Identificar, a partir do comportamento ineficiente dos agentes de final de jogo, as situações excepcionais em que a RN não foi capaz de alocar o agente mais adequado

para atuar no jogo e, com base nessas situações, propor um método baseado em regras de exceção que seja capaz de tratar tais situações, complementando a indicação da RN. Tal método deve ser utilizado, após a indicação da RN, como um identificador de situações excepcionais e, caso necessário, como um otimizador da indicação da RN.

1.3 Hipóteses

A hipótese central desta pesquisa é de que as técnicas de AM não supervisionada, quando aplicadas a sistemas multiagentes, podem ser aprimoradas de modo a reforçar as habilidades desses sistemas, tornando-os cada vez mais eficazes e eficientes na resolução dos problemas para os quais foram propostos. Para alcançar a hipótese central deste trabalho considerando o sistema multiagente MP-Draughts e perseguindo os objetivos apresentados na Seção 1.2, as seguintes hipóteses devem ser validadas:

1. Jogadores automáticos de Damas multiagentes são mais eficientes que agentes jogadores de agentes simples. A divisão do jogo em fases e a obtenção de agentes especialistas em cada fase tende a aumentar o desempenho geral de um jogador automático;

2. Existe uma configuração adequada de SMA para atuar na fase final do jogo de Damas, a qual é passível de ser definida sem o auxílio de um especialista do domínio, utilizando apenas RNs adaptativas. Tais RNs são capazes de detectar e delimitar, a partir de várias amostras de tabuleiros, os diferentes perfis de conhecimentos existentes nesta fase do jogo;

3. Sabe-se que representar apenas as características relevantes do ambiente sobre o qual agentes inteligentes atuam é fundamental para sua boa performance. Sendo assim, a definição de quais são as características relevantes que um jogador automático de Damas deve considerar durante sua atuação no jogo é fundamental para sua boa performance. Além disso, quando tais informações são refinadas para representar perfis específicos que ocorrem nas diferentes fases do jogo, a performance do jogador tende a ser ainda melhor;

4. As alocações inadequadas de agentes para atuarem em situações para as quais não foram habilitados, quando ocorrem, são determinantes para o baixo desempenho do SMA. Como tais alocações ocorrem com menor frequência e em algumas situações específicas, é possível identificar e tratar localmente essas situações utilizando uma técnica complementar à utilizada para a alocação geral.

1.4 Contribuições

A contribuição geral deste trabalho está relacionada as novas propostas e as otimizações de técnicas de AM não supervisionadas aplicáveis ao domínio de SMAs, as quais podem contribuir para o desempenho geral do sistema. Considerando o domínio do jogo de Damas, são contribuições deste trabalho:

1. Proposta de uma arquitetura de SMA para atuar na fase final do jogo de Damas, cuja quantidade de agentes seja suficientemente adequada para esta fase do jogo;

2. Proposta de uma RN adaptativa apropriada para definir e delimitar os perfis de conhecimentos existentes na fase final do jogo de Damas, os quais possam ser utilizados na formação do conhecimento (treinamento) dos agentes especialistas nessa fase;

3. Proposta de conjuntos de características adequados para representar os diferentes ambientes sobre os quais agentes jogadores com diferentes habilidades atuam. Tais conjuntos focam apenas nos conhecimentos que de fato contribuem para a boa atuação desses agentes;

4. Proposta híbrida de um método alocador de agentes em SMA que combina RNs de agrupamento e regra de exceção, as quais em conjunto definem os agentes adequados para atuarem em cada situação do ambiente. Neste método, as regras de exceção são utilizadas localmente para refinar a indicação da RN em algumas situações especiais do ambiente.

1.5 Organização da Tese

Os próximos capítulos deste trabalho estão organizados da seguinte forma: no *Capítulo 2* são explicados os fundamentos teóricos e os trabalhos correlatos, relacionados ao desenvolvimento das abordagens propostas. A seção 2.1 apresenta as regras do jogo de Damas. A Seção 2.2 apresenta os principais jogadores automáticos de Damas, os quais estão divididos em: Jogadores Automáticos de Damas com Aprendizagem Supervisionada e Jogadores Automáticos de Damas com Aprendizagem Não Supervisionada. Finalmente, na Seção 2.3 são apresentados os fundamentos relacionados a: Sistemas Multiagentes; Redes Neurais MLP; Aprendizagem por Reforço e o Método TD(λ); Algoritmo de Busca Alfa-Beta; Representação do Tabuleiro do Jogo de Damas; Redes Neurais aplicadas à Agrupamento de Dados: KSOM, ART 2A e SONDE; Técnicas de Mineração de Dados: mineração de itens frequentes, regras de exceção e mineração de árvores de decisão.

O *Capítulo 3* apresenta a arquitetura geral do multiagente MP-Draughts, assim como a explicação de cada módulo que compõe sua arquitetura. Também é apresentado uma

nova versão de RN adaptativas, a ASONDE, a qual foi utilizada para a adequação da arquitetura do multiagente para a fase de final de jogo. Além disso, este capítulo apresenta uma investigação acerca das melhores medidas de similaridade para representar os estados de tabuleiros a serem agrupados pela RN ASONDE.

O *Capítulo 4* apresenta a abordagem utilizada para obter os conjuntos de características adequados para representar o tabuleiro do jogo de Damas nas distintas fase do jogo. Foram obtidos conjuntos de características que são relevantes durante todo o jogo e também, aqueles relevantes para a fase de final de jogo. Para a obtenção desses conjuntos foi aplicada uma técnica de mineração de itens frequentes de BDs, cujas BDs continham milhares de configurações de estados de tabuleiros que ocorreram em jogos envolvendo grandes mestres do jogo.

O *Capítulo 5* explica o método híbrido de alocação de agentes em SMAs, o qual é baseado em RN de agrupamento e regras de exceção. Para tanto, é apresentado neste capítulo: a estratégia utilizada para identificar as situações especiais de baixo desempenho da RN, as quais foram utilizadas para obter as regras de exceção; o processo de obtenção e seleção das melhores regras de exceção a serem combinadas com a RN; o processo de combinação de ambas as técnicas na tarefa de alocação dos agentes, além da dinâmica de jogo utilizando tal método de alocação.

Finalmente, no *Capítulo 6* são apresentadas as conclusões e as propostas para trabalhos futuros, assim como as produções bibliográficas obtidas do cumprimento dos objetivos aqui propostos.

Capítulo 2

Fundamentos Teóricos e Estado da Arte

Nesse capítulo são apresentados os principais conceitos utilizados para o desenvolvimento deste trabalho. Inicialmente são apresentadas as regras do jogo de Damas Inglesas, cujo domínio foi utilizado como ambiente de desenvolvimento das abordagens propostas. Na sequência são apresentados os principais jogadores automáticos de Damas propostos na literatura, os quais influenciaram no desenvolvimento do trabalho e por fim, são apresentadas as principais fundamentações técnicas utilizadas.

2.1 O Jogo de Damas

Damas é um jogo de tabuleiro jogado por 2 jogadores, os quais tentam imobilizar ou capturar todas as peças de seu adversário, sendo que vence o jogo aquele jogador que atingir tal objetivo primeiro. A versão do jogo de Damas utilizada neste trabalho é a inglesa, em que tabuleiro é composto por 64 casas, alternadamente casas claras e escuras, dispostas em uma matriz quadrada de 8 linhas e 8 colunas (8×8). As linhas oblíquas formadas pelas casas escuras são chamadas de diagonais, totalizando 15 diagonais sobre o tabuleiro do jogo. A maior diagonal, conhecida como grande diagonal, contém 8 casas escuras e une os dois cantos do tabuleiro. Coloca-se o tabuleiro entre os jogadores, de modo que a grande diagonal comece à esquerda de cada jogador e a primeira casa à esquerda de cada um seja escura. O jogo deve acontecer apenas nas casas escuras (também conhecida como casas ativas) e cada jogador começa o jogo com doze peças simples localizadas nas três primeiras linhas mais próximas do seu lado no tabuleiro. A Figura 1 mostra a disposição de peças iniciais de um tabuleiro de Damas 8×8.

As regras do jogo de Damas Inglesas, as quais são adotadas pelo jogador MP-Draughts, são as seguintes:

- Uma peça simples movimenta-se em diagonal, sobre as casas escuras vazias, apenas para frente e uma casa por movimento;

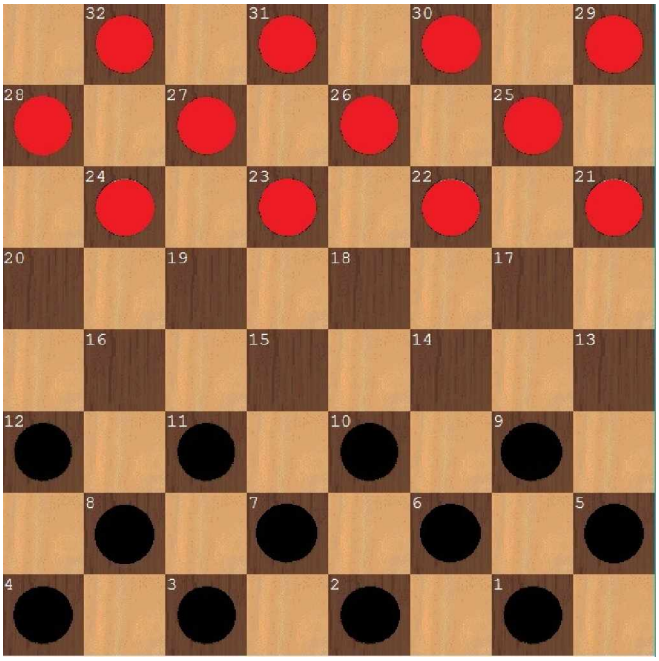

Figura 1 – Configuração inicial de um tabuleiro 8×8 do jogo de Damas Inglesas.

- Se uma peça simples atinge a última linha do outro lado do tabuleiro (lado oposto ao que começou o jogo), é promovida para dama (ou rainha). Marca-se a dama colocando outra peça sobre ela. A dama possui mais possibilidades de movimentos, podendo se mover tanto para frente quanto para trás em diagonal, porém, apenas uma casa por movimento;

- O jogador pode saltar apenas sobre peças do adversário e nunca sobre as suas próprias peças;

- Se uma peça está de frente para uma peça adversária e a casa subsequente à peça adversária está livre, então o jogador deve capturar a peça adversária saltando por cima dela, retirando-a do tabuleiro e ocupando a casa subsequente. Se, após o salto, existirem mais peças adversárias que possam ser capturadas, o jogador deve capturá-las. É importante destacar que a captura de peça(s) adversária(s) é um movimento *obrigatório* e tem prioridade em relação a qualquer outro movimento. A dama pode capturar peças do adversário movimentando-se tanto para frente quanto para trás e uma peça simples pode capturar apenas com movimentos para frente;

- O jogo de Damas Inglesas não utiliza a *lei da maioria* para captura de peças, ou seja, se existir mais de uma opção de captura, o jogador não é obrigado a executar o movimento que capture a maior quantidade de peças. Sendo assim, se o jogador possuir mais de uma possibilidade de capturar peças adversárias, ele pode optar por qualquer uma delas;

- Se o jogador estiver movimentando uma peça simples e optar por uma captura que

atinja o lado oposto do tabuleiro (a última linha), tal captura deve ser interrompida nesse ponto com a promoção da peça simples para dama. Ou seja, quando uma peça simples é promovida, a jogada deve ser finalizada nesse instante;

- Duas ou mais peças do adversário que estejam juntas na mesma diagonal não podem ser capturadas, a não ser que haja uma única casa vazia entre elas;
- Na execução de um movimento de captura, é permitido passar mais de uma vez por uma mesma casa vazia;
- Se ambos jogadores executarem os mesmos movimentos em suas 5 últimas jogadas, ou se o jogo se extender por mais de 100 movimentos, o jogo é finalizado e é declarado um empate.

2.2 Trabalhos Relacionados

A literatura apresenta vários jogadores automáticos de Damas, os quais podem ser divididos em 2 grupos: aqueles que contam com forte supervisão humana durante o processo de aprendizagem e, aqueles que aprendem a jogar com o mínimo de intervenção humana (classificados aqui como jogadores não supervisionados). No grupo dos jogadores supervisionados, este trabalho apresenta os dois principais jogadores, o Chinook [19] e o Cake [36]. No grupo dos jogadores não supervisionados apresenta-se os principais jogadores desenvolvidos pela equipe a qual o trabalho se insere, além de apresentar outros jogadores recentemente propostos, os quais possuem uma metodologia de aprendizagem diferente da utilizada pelos jogadores desta equipe.

2.2.1 Jogadores Automáticos de Damas - Aprendizagem Supervisionada

2.2.1.1 Chinook

Chinook é o mais famoso e melhor jogador automático de Damas de todos os tempos. Este jogador obteve, em 1994, o título de campeão mundial de Damas ao empatar 6 jogos com o jogador Mario Tinsley, que defendia o título mundial de melhor jogador de Damas a mais de 40 anos [19], [37]. O projeto de pesquisa que deu origem a este jogador foi iniciado em 1989 [38]. A versão campeã mundial do Chinook conta com uma BD de inicio de jogo, conhecida como *opening books*, que contém uma série de movimentos que devem ser evitados na fase inicial do jogo (no jogo de Damas, é fundamental evitar algumas posições no início de um jogo). Este jogador também conta com uma BD de final de jogo que contém cerca de 39 trilhões de estados do tabuleiro com valor teórico provado de vitória, empate ou derrota. Tal BD mapeia todos os possíveis movimentos a serem

executados pelo jogador em estados de tabuleiros com 10 peças ou menos. Para escolher o melhor movimento a ser executado, o jogador utiliza um procedimento de busca Minimax distribuída com poda Alfa-Beta, Aprofundamento Iterativo e Tabela de Transposição. Para o Chinook, o jogo é dividido em 4 fases e cada fase possui 21 características que foram ajustadas manualmente para totalizar os 84 parâmetros de sua função de avaliação. Estes parâmetros foram ajustados, ao longo de 5 anos, a partir de testes exaustivos em jogos contra si mesmo e contra os melhores jogadores humanos, inclusive o Mario Tinsley. O processo de aprendizagem deste jogador foi altamente supervisionado durante este período.

Em 2007, a equipe do Chinook anunciou que o jogo de Damas estava fracamente resolvido (*weakly solved*). Isso significa dizer que, a partir da posição inicial do jogo existe uma prova computacional de que, no pior dos casos, o Chinook termina o jogo em empate. A prova consiste de uma estratégia explícita com a qual o jogador nunca perde, isto é, o jogador pode alcançar o empate contra qualquer oponente jogando tanto com peças pretas quanto vermelhas [20]. Tal feito foi possível porque esta versão do jogador utiliza BDs que contêm todos os movimentos do jogo que devem ser executados pelo jogador de modo a alcançar a vitória. Note que a resolução fraca do jogo de Damas foi possível devido a uma extensiva investigação de movimentos executada pela equipe de desenvolvimento deste jogador durante aproximadamente 18 anos, a qual resultou em extensas BDs que mapeiam todos os melhores movimentos do jogo partindo do estado inicial do jogo. Além disso, tal pesquisa contou com o apoio do maior especialista de todos os tempos no assunto, Mario Tinsley. Vale reforçar aqui que este tipo de solução, essencialmente supervisionada, não é uma estratégia fácil, muito menos viável, de ser aplicada para a resolução de outros problemas com complexidade de resolução semelhante à do jogo de Damas.

2.2.1.2 Cake

O Cake está entre os jogadores de Damas mais competitivos do mundo. Este jogador é superior a versão campeã mundial do Chinook [19] que venceu o campeonato mundial contra Mario Tinsley [36]. O Cake também utiliza BD de fim de jogo, a qual contém conhecimento perfeito do valor de qualquer movimento para tabuleiros com até 8 peças. Ele também utiliza um livro de abertura de jogos, (*openning book*), que contém cerca de 2 milhões de movimentos. Para atuar no jogo entre as fases de inicio e fim de jogo (fase em que não possui acesso as BDs), ele usa o algoritmo de busca MTD(f) para escolher o melhor movimento a ser executado. Este algoritmo efetua uma média de avaliação de 2 milhões de movimentos por segundo. Assim como o Chinook, o Cake é um jogador fortemente supervisionado.

Tal jogador encontra-se disponível na plataforma *CheckerBoard*, que é a mais completa interface livre para programas jogadores de Damas. Esta plataforma suporta arquivos *Portable Draughts Notation* (PDN), que é o formato padrão para arquivos de jogos

de Damas. A *CheckerBoard* disponibiliza também BDs contendo milhares de jogos de competições entre os grandes jogadores de Damas de todos os tempos, inclusive jogos do Mario Tinsley e do Chinook.

Devido a disponibilidade de acesso à plataforma *CheckerBoard* na qual o Cake está inserido e por se tratar de um excelente jogador automático de Damas, este jogador foi utilizado para comprovar a evolução da performance do MP-Draughts ao longo do desenvolvimento das pequisas aqui propostas. Sabe-se no entanto que o MP-Draughts, que foi concebido baseado apenas em abordagens não supervisionadas, não é capaz de vencer este jogador fortemente supervisionado.

2.2.2 Jogadores Automáticos de Damas - Aprendizagem Não Supervisionada

2.2.2.1 Jogadores não supervisionados propostos pela equipe do MP-Draughts

O VisionDraughts [21] é um jogador automático de Damas, cujo principal objetivo de seus criadores foi construir um jogador automático competitivo que tenha o mínimo de intervenção humana no processo de aprendizagem. Ele é inspirado no jogador de Mark Lynch, o NeuroDraughts [39]. O VisionDraughts foi implementado como uma MLP que utiliza a busca Minimax com poda Alfa-Beta, TT e AI para, em função do estado corrente do jogo, escolher o melhor movimento. O estado corrente do jogo é apresentado ao algoritmo de busca utilizando a representação vetorial. Durante o processo de aprendizagem, a MLP utiliza o método de aprendizagem por reforço TD(λ) aliado à estratégia de treino por *self-play* com clonagem. O tabuleiro do jogo é apresentado à MLP utilizando a representação *NetFeatureMap*. Este jogador utiliza um subconjunto de 12 características para representar o estado de tabuleiro do jogo. Os resultados apresentados em [21] comprovam que o VisionDraughts é, pelo menos, 50% mais eficiente com tempo de busca 95% inferior quando comparado ao seu predecessor NeuroDraughts. Devido a este jogador ser utilizado como base para a criação do MP-Draughts, este também será utilizado para comprovar a evolução do MP-Draughts.

O D-VisionDraughts [24], [27] é o jogador de Damas que corresponde a versão distribuída do VisionDraughts. Este jogador substitui o algoritmo de busca sequencial Alfa-Beta do VisionDraughts pelo algoritmo de busca distribuído *Young Brothers Wait Concept* (YBWC) [40]. Além disso, o D-VisionDraughts ordena a árvore de busca do jogo através de heurísticas. A paralelização do algoritmo de busca permitiu ao jogador aumentar a visão futura (*look-ahead*) do estado corrente do jogo para a escolha do melhor movimento, bem como uma redução significativa do tempo de treinamento. A ordenação da árvore de busca contribuiu para a redução do tempo de busca do algoritmo, visto que os nós mais prováveis de causar uma poda são avaliados primeiro. Experimentos mostraram que a busca distribuída, juntamente com o uso de heurísticas para ordenação da

árvore de busca, reduziu em 95% o tempo de busca quando comparado com o algoritmo serial sem ordenação usado pelo VisionDraughts. Além disso, esta redução permitiu ao D-VisionDraughts, nas mesmas condições de treinamento, obter melhor desempenho de jogo do que o VisionDraughts, o que representa cerca de 15% a mais de jogos vitoriosos para este jogador.

O MP-Draughts [22], [23] foi proposto com o objetivo de resolver o problema de *loops*[1] de final de jogo que frequentemente ocorrem nos jogos de Damas envolvendo jogadores automáticos, os quais nem sempre são benéficos para os jogadores. A primeira versão deste jogador corresponde a um sistema multiagente composto por 26 agentes especialistas em fases distintas do jogo de Damas. Um agente, o IIGA, é especializado nas fases iniciais e intermediárias de um jogo. Os outros 25 agentes, os EGAs, são especialistas na fase final do jogo. Todos os agentes possuem a mesma estrutura do VisionDraughts, sendo que o IIGA é o próprio. Cada EGA foi treinado para ser capaz de lidar com um determinado perfil de final de jogo. Esses perfis (*cluster*) foram minerados, por uma RN KSOM-DE, de uma BD de estados de tabuleiros de final de jogo. O treinamento de cada EGA foi realizado em todos os estados de tabuleiros pertencentes ao *cluster* o qual ele foi designado a representar. Os resultados obtidos indicaram que, apesar do aumento no tempo de treinamento devido a quantidade maior de agentes, este jogador conseguiu reduzir a incidência de *loops* de final de jogo, além de ser mais eficiente em torneios contra o VisionDraughts e o NeuroDraughts. O MP-Draughts foi utilizado como ambiente de desenvolvimento e investigação das técnicas propostas neste trabalho, de modo que a arquitetura geral do jogador e as otimizações efetuadas sobre a mesma serão apresentadas nos capítulos subsequentes.

O LS-VisionDraughts [29] é um agente não supervisionado jogador de Damas que utiliza Algoritmo Genético (AG) para automatizar a escolha das características mais apropriadas para representar os estados de tabuleiros dos jogos. A arquitetura deste jogador também foi inspirada no VisionDraughts. Desse modo, a combinação do Algoritmo Genético com o módulo de busca Alfa-Beta com TT e AI permitiu que esse jogador, treinado num subconjunto de 9 características, fosse mais eficiente que o VisionDraughts. Além disso, o LS-VisionDraughts também reduziu o tempo de treinamento da MLP, o que foi possível devido a redução da quantidade de características (funções) avaliadas por esta RN no momento de predizer a qualidade de cada movimento. Dentre os monoagentes jogadores de Damas criados pela equipe de pesquisa ao qual o MP-Draughts se insere, o LS-VisionDraughts é o melhor deles. Por isso, este jogador também será utilizado para comprovar a evolução de desempenho do MP-Draughts.

[1] Um *loop* ocorre quando o agente, mesmo em vantagem, não consegue pressionar o adversário e alcançar a vitória. Ao invés disso, o agente começa uma sequência repetitiva de movimentos alternando-se entre posições inúteis do tabuleiro, os quais não modificam o estado do jogo.

2.2.2.2 Jogadores não supervisionados propostos por outras equipes

Fogel utilizou um processo coevolutivo para implementar um jogador automático de Damas baseado em MLP que fosse capaz de aprender a jogar sem utilizar conhecimento humano sobre o domínio do jogo. De acordo com Fogel, a MLP era capaz de "criar" características para representar o jogo baseando-se apenas na representação espacial das peças sobre o tabuleiro, utilizada por esta MLP. O melhor jogador obtido por Fogel, conhecido como Anaconda [41], foi resultado da evolução de 30 MLPs ao longo de 840 gerações. Cada geração tinha em torno de 150 jogos de treinamento, sendo que foram necessários 126.000 jogos de treinamento e 6 meses de execução para obter o Anaconda. Em um torneio de 10 jogos contra uma versão de baixo desempenho do Chinook, o Anaconda obteve 2 vitórias, 4 empates e 4 derrotas. Esse resultado foi suficiente para classificá-lo como *expert* [42].

Um jogador recentemente proposto é o jogador de Al-Khateeb e Kendall [26]. Este jogador é uma evolução do jogador do Fogel, o qual acrescenta um mecanismo de aprendizagem social e individual na fase de aprendizagem da MLP coevolutiva com o objetivo de melhorar seu processo de aprendizagem. De acordo com os autores, este jogador obteve melhor desempenho que seu predecessor.

Outro jogador não supervisionado recentemente proposto, é o jogador de Cheheltani e Ebadzadeh [25]. Este jogador é um agente imunológico difuso que usa células de memória permanente para representar os movimentos do jogo. Estas células ajudam um sistema de inferência difuso de Mamdani (do inglês *Mamdani Fuzzy Inference Engine* (FIS)) a decidir qual é o melhor movimento a ser executado. Para isso, são considerados o estado anterior e o próximo estado do jogo. Segundo o autor, num torneio de 50 jogos contra o jogador de Fogel [42], o agente imunológico obteve 66% de vitórias contra 10% de vitórias do oponente.

Em [28] foi proposto um agente jogador de Damas, o MPACA, o qual é composto por um módulo de processamento de imagem que detecta os movimentos do oponente humano, um *hardware* robô que executa os movimentos do agente sobre o tabuleiro e um módulo de IA, responsável por definir qual a melhor ação a ser executada. Tal módulo é composto por uma versão distribuída do algoritmo NegaMax [43] com poda Alfa-Beta e utilizando uma BD que armazena e recupera valores de tabuleiros avaliados anteriormente pelo jogador.

Frankland e Pillay propuseram em [32] uma arquitetura híbrida jogadora de Damas baseada em Programação Genética (PG) e AR, a qual evolui estratégias do jogo de Damas baseada em algumas heurísticas que dividem o tabuleiro do jogo em áreas estratégicas, tais como: regiões propícias para ataque e defesa. Assim, cada indivíduo da PG corresponde a uma estratégia diferente que, através de um conjunto de heurísticas, define qual o movimento a ser executado sobre um determinado tabuleiro. O processo evolutivo do sistema pode ser resumido assim: toda vez que o sistema deve executar um movimento

sobre um determinado tabuleiro do jogo, uma execução separada de um ciclo inteiro da PG é realizada com o objetivo de obter o melhor indivíduo (estratégia), também chamado pelos autores como jogador alfa, o qual é responsável por definir a ação a ser executada nesse tabuleiro. Este processo repete até o fim do jogo e usa AR para melhorar as estratégias evoluídas. Segundo os autores, os resultados obtidos por esta arquitetura se mostraram bastante promissores.

Por último, uma abordagem particular de jogador automático de Damas Chinesas foi proposto por [44], o qual usa mineração de padrões frequentes para extrair conhecimento de BD de históricos de jogos. Os conhecimentos extraídos da BD são utilizados para aprimorar a inteligência do jogador e também suas estratégias de jogo. Técnicas relacionadas a mineração de dados tem sido eficientemente utilizadas otimizar jogadores automáticos [45], [46].

2.3 Fundamentos Teóricos

Esta seção apresenta as técnicas utilizadas para o desenvolvimento desse trabalho. Da Seção 2.3.1 até a 2.3.5 são apresentadas as técnicas utilizadas para o desenvolvimento do multiagente jogador automático MP-Draughts, o qual foi utilizado como ambiente de desenvolvimento de todas as abordagens aqui propostas. Nas seções 2.3.6 e 2.3.7 são descritas as técnicas que foram utilizadas para desenvolvimentos de tais propostas.

2.3.1 Sistemas Multiagentes - SMAs

Adotar a abordagem de SMA para a resolução de problemas tem sido um dos assuntos mais explorados no contexto de sistemas inteligentes nas últimas décadas. Inúmeros trabalhos apresentam conceitos, técnicas e aplicação de SMA em diversos domínios de problemas, os quais foram implementados tanto em softwares como em hardwares [47], [48], [49], [50]. Isso se deve ao fato da abordagem multiagente possuir várias características que viabilizam a resolução de problemas, adequando-se a problemas complexos e de natureza descentralizada [8].

Um SMA é um sistema composto por dois ou mais agentes autônomos que interagem entre si em prol de realizarem determinadas tarefas ou alcançarem determinados objetivos. Nesses sistemas, cada agente pode atuar numa determinada parte do problema e interagir com os demais para tentar atingir o objetivo ou ainda, para ajudar os outros a alcançarem seus objetivos. Tais agentes exibem duas características fundamentais: são capazes de agir de forma autônoma tomando decisões que os levam à satisfação de seus objetivos e, também interagem com outros agentes no intuito de coordenar, cooperar, competir e/ou negociar as ações que os levam à satisfação do objetivo [7].

Russell e Norvig [51] exemplificam um comportamento cooperativo entre agentes em um ambiente em que um agente A (motorista de um veículo) interage com outro agente

B (outro veículo), de modo que ambos atuem colaborativamente nesse ambiente (no caso, na mesma rua) evitando conflitos (por exemplo, uma colisão). Um exemplo de comportamento competitivo entre agentes num mesmo ambiente pode ser observado no domínio dos jogos de tabuleiros, em que eles competem entre si pelo mesmo objetivo, que é vencer o jogo. No entanto, num ambiente de comportamento competitivo também é possível existir o comportamento cooperativo em que, dentre os diversos agentes que atuam no ambiente, existem aqueles que cooperam entre si em prol do mesmo objetivo e competem com outros pelo objetivo. No caso do jogo de Damas, por exemplo, pode haver um conjunto de agentes que cooperam entre si para ganhar o jogo e competem, contra o adversário pelo mesmo objetivo.

Neste trabalho, a técnica de SMA foi utilizada para criar um multiagente jogador de Damas, cujos agentes componentes atuam em busca do objetivo de vencer o jogo. Tais agentes são especializados em distintas fases do jogo de Damas.

2.3.2 Redes Neurais *Perceptron* Múltiplas Camadas - MLP

Uma rede neural artificial é um modelo computacional, baseado em redes neurais biológicas, que consiste em uma rede de unidades básicas simples chamadas neurônios (nós). O primeiro modelo matemático de um neurônio artificial foi proposto por McCulloch e Pitts [1] em 1943, sendo ainda o modelo mais utilizado por diversas arquiteturas de RNs. Na representação de McCulloch, um neurônio pode ser representado conforme a Figura 2. O neurônio é uma unidade processadora que aplica uma função de ativação g a um conjunto de n entradas, as quais são combinadas por uma função de entrada in (função soma). Os dados de entrada do neurônio são representados pelo conjunto $\{x_0, ..., x_n\}$, os pesos sinápticos são representados pelo conjunto $\{w_{0,j}, ..., w_{n,j}\}$. Um peso sináptico $w_{i,j}$ representa a força da conexão entre uma entrada x_i e um neurônio j. A relevância de cada uma das entradas x_i do neurônio é obtida a partir de sua multiplicação pelos respectivos pesos sinápticos w_i, ponderando, dessa maneira, todas as informações que chegam ao neurônio.

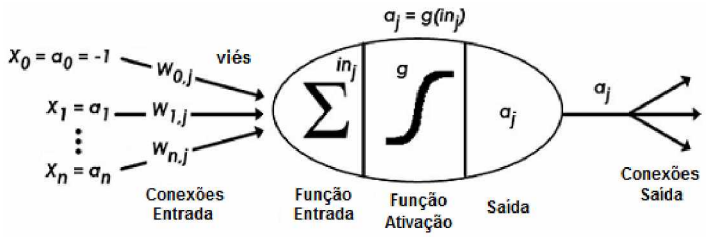

Figura 2 – Modelo de um neurônio artificial [1].

O limiar (ou viés) é representado pelo sinal de entrada x_0, o qual é conectado ao neurônio j através do peso sináptico $w_{0,j}$. O viés é a variável que estabelece o patamar apropriado para que o resultado produzido pela função soma possa gerar (ou não) um

valor de disparo (a_j) em direção à saída do neurônio. Em outras palavras, o neurônio j emite um sinal de disparo a_j se, e somente se, a restrição da Equação 1 for satisfeita:

$$\sum_{i=1}^{n} w_{i,j}.x_i > w_{0,j}.x_0. \tag{1}$$

A saída a_j, correspondente a um neurônio j, pode ser obtida pela Equação 2:

$$a_j = g(in_j) = g(\sum_{i=0}^{n} w_{i,j}.x_i). \tag{2}$$

A função de ativação g deve ser definida de acordo com as necessidades de cada problema. Existem várias funções de avaliação propostas na literatura, as quais são apropriadas para cada tipo de problema. Estas funções podem ser divididas em dois grupos principais: funções parcialmente diferenciáveis e funções totalmente diferenciáveis, considerando para tanto o domínio de definição das mesmas [52].

Em resumo, uma RN é composta por um conjunto de neurônios interconectados, onde a saída a_i de um neurônio i corresponde a entrada x_j de um neurônio j ao qual i está conectado com peso sináptico $w_{i,j}$. O processo de treinamento de um RN consiste em ajustar os pesos sinápticos que conectam seus neurônios até encontrar uma conexão adequada entre eles, a qual seja capaz de produzir a saída desejada correspondente ao sinal de entrada. Note que o conhecimento de uma RN reside nos seus pesos sinápticos [53].

As MLPs são um tipo especial de RNs, caracterizadas pela presença de pelo menos uma camada intermediária (oculta) de neurônios, situada entre a camada de entrada dos dados e a camada neural de saída. Estas RNs têm a capacidade de resolver problemas não-linearmente separáveis. A Figura 3 apresenta uma RN MLP. Cada saída a_j^m (aqui referenciado como O_j) relativa ao *j-ésimo* neurônio de saída da camada m é dada na equação 3:

$$O_j = g(in_j) = g(\sum_{i=0}^{n} w_{i,j}^{m-1}.a_i^{m-1}) = a_j^m, \tag{3}$$

onde a_i^{m-1} é a saída do *i-ésimo* neurônio da camada *m-1* conectada ao neurônio de saída j com peso $w_{i,j}^{m-1}$ (note que a saída a_i^{m-1} corresponde ao sinal de entrada x_i do neurônio j).

Considerando que uma MLP possui aprendizado supervisionado, o ajuste dos pesos sinápticos é feito de forma a minimizar a diferença (*erro*) entre os valores O_j obtidos pela RN e a saída desejada. Considerando o aprendizado não supervisionado (aprendizagem por reforço), onde o valor desejado não está disponível, esta diferença é substituída pela diferença entre as duas saídas sucessivas produzidas pela RN.

O multiagente utilizado neste trabalho corresponde a um conjunto de MLPs de aprendizagem não supervisionada, cujo processo de aprendizagem é guiado pelo método TD(λ).

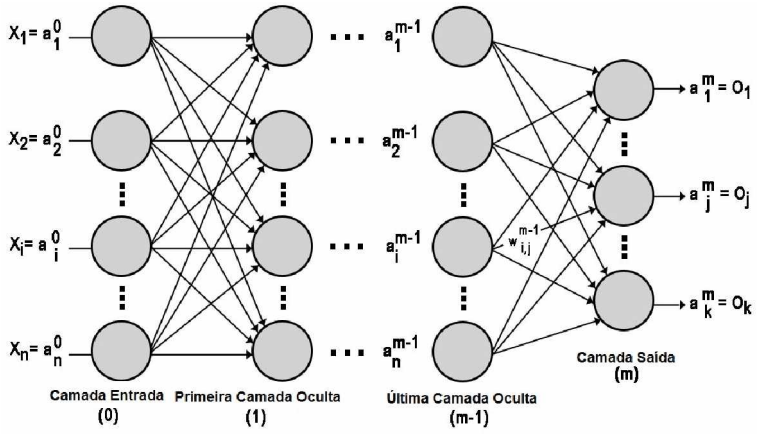

Figura 3 – Arquitetura de uma *Perceptron* Múltiplas Camadas - MLP

2.3.3 Aprendizagem por Reforço e o Método das Diferenças Temporais - TD(λ)

O paradigma da AR é um tema de grande interesse na área de AM, visto que dispensa um "professor" para nortear o processo de aprendizagem do agente (fornecimento de exemplos de treinamento). Este fato torna a AR particularmente adequada para domínios em que a obtenção de exemplos de treinamento seja difícil ou até mesmo impossível [51]. Segundo Sutton [54], na AR um agente deve aprender a partir da sua interação com o ambiente onde se encontra, através do conhecimento do seu próprio estado no ambiente, das ações efetuadas e das mudanças de estado que aconteceram no ambiente depois de efetuadas as ações. O sucesso da utilização de AR como uma técnica de AM está diretamente relacionada ao fato de se obter uma política apropriada de ações, a qual corresponde ao comportamento que o agente deve adotar para alcançar seu objetivo.

Entre as técnicas de AR tradicionalmente conhecidas, este trabalho destaca o método das Diferenças Temporais TD(λ) [55]. Neste método, o processo de aprendizado é guiado pela diferença entre valores sucessivos dos reforços (particularmente, no caso de TD(λ), denominados *predições*) que são calculados pelo agente a cada vez que ele executa uma ação no ambiente. O valor de predição indica o quanto um novo estado resultante da execução de uma determinada ação é favorável ao agente. Desta forma, se um agente executa uma sequência de ações m_0,..., m_{i-1}, m_i, m_{i+1},..., m_j como uma etapa preliminar de planejamento para tentar alcançar um determinado objetivo f, o sistema calcula o valor da predição associado a cada estado resultante da execução de cada uma destas ações. Os valores das sucessivas diferenças entre as predições serão usados para atualizar a função de aprendizado do sistema. Consequentemente, no método TD(λ), todas as ações executadas durante o treinamento do agente têm impacto no processo de aprendizagem, o que caracteriza a propriedade do método de considerar o impacto ponderado de cada ação m_i executada ao longo do tempo. Logo, a eficiência do aprendizado depende, diretamente, do valor de precisão dos estados envolvidos que são estimados pela *função de avaliação*.

Particularmente, no caso de agentes jogadores, o objetivo f é um estado de final de jogo, o qual é atingido em uma situação particular em que a sequência de ações executada é tal que a execução da última ação m_j leva ao estado f de final de jogo.

É importante destacar que as predições associadas a estados intermediários (todos estados exceto o de final de jogo f) são calculados por meio da função de avaliação adotada pelo sistema. Por outro lado, a predição associada ao estado de final de jogo é obtida por um valor de reforço retornado pelo ambiente de acordo com o resultado final obtido pelo agente (por exemplo, a predição correspondente para um estado de final de jogo em que o jogador vence deve indicar uma recompensa, enquanto que a derrota deve indicar uma punição).

2.3.4 Algoritmo de Busca Alfa-Beta

Estratégias de busca tradicionais efetuam a busca numa árvore que descreve todos os estados possíveis a partir de um determinado estado inicial I_0. Formalmente, o espaço de busca é constituído por um conjunto de nós conectados por arcos e, cada arco pode ou não estar associado a um valor que corresponde ao custo c de transição de um nó para o outro. A cada nó tem-se associada uma profundidade d, sendo que a mesma tem valor 0 no nó raiz e aumenta de uma unidade para um nó filho. A aridade a de um nó é a quantidade de filhos que o mesmo possui, e a aridade de uma árvore é definida pela maior aridade de um de seus nós. O objetivo de uma busca é encontrar um caminho (ótimo ou não) do estado inicial I_i até um estado final I_f explorando, sucessivamente, os nós conectados ao nós já explorados até a obtenção de uma solução para o problema [29]. O jogo de Damas pode ser visto como uma árvore de possíveis estados de tabuleiros, sendo que a raiz da árvore representa o estado atual do jogo, os nós filhos representam os possível estados de tabuleiros obtidos a partir de movimentos legais (representados por arcos) executados neste tabuleiro.

A estratégia de busca utilizada por muitos jogadores automáticos de Damas, por várias décadas, foi o algoritmo de busca Minimax. Este algoritmo determina a estratégia ótima num cenário de jogo com dois jogadores (jogador *min* e jogador *max*). O objetivo do Minimax é buscar o melhor movimento para *max* de modo a minimizar os efeitos do movimento a ser executado pelo oponente *min* [51]. No Minimax, cada nível da árvore de busca corresponde, alternadamente, a um nível de maximização e um nível de minimização, sendo que a raiz da árvore é um nível de maximização. No nível de minimização, o algoritmo escolhe o movimento que leva ao seu sucessor de menor avaliação enquanto, no nível de maximização, o algoritmo escolhe o movimento que leva ao seu sucessor de maior avaliação. A Figura 4 (a) mostra o exemplo de uma árvore gerada pelo algoritmo Minimax na busca pelo melhor movimento para *max*. Observe que o *movimento A* é a melhor opção para *max*, visto que este é o movimento que leva para ao sucessor de maior avaliação.

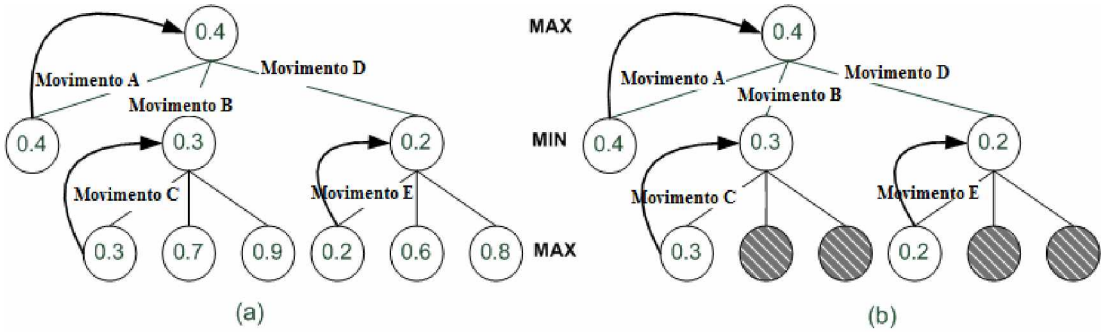

Figura 4 – (a) Árvore de busca expandida pelo algoritmo Minimax. (b) Árvore de busca expandida pelo algoritmo Alfa-Beta

Apesar de ser um algoritmo de busca tradicional, o Minimax não é eficiente, visto que ele avalia mais nós da árvore de busca do que o necessário. Uma alternativa ao Minimax, que vem sendo empregada na arquitetura de vários jogadores automáticos, é a utilização do mecanismo de busca com poda Alfa-Beta. O Alfa-Beta elimina seções da árvore de busca do Minimax que, definitivamente, não contém o melhor movimento a ser executado pelo jogador. Tal algoritmo pode ser resumido como um procedimento recursivo que escolhe o melhor movimento a ser executado efetuando uma busca em profundidade, da esquerda para a direita, na árvore de busca [56], [57]. O Alfa-Beta recebe este nome devido aos parâmetros *alfa* e *beta* que são passados como argumentos para o algoritmo em conjunto com o estado atual do jogo. Alfa e beta delimitam, respectivamente, o intervalo inferior e superior da janela de busca pelo melhor movimento correspondente ao estado atual do jogo I_i. A avaliação dos nós filhos de um determinado nó no nível de minimização (nó minimizador) pode ser interrompida quão breve a avaliação de um desses nós seja inferior ao limite mínimo estabelecido (*poda alfa*). Por exemplo, na Figura 4 (b), o algoritmo Alfa-Beta detecta que não é necessário avaliar 2 dos nós filhos dos movimentos B e D (destacados de cinza), visto que os filhos já avaliados possuem valores inferiores à janela alfa (0.4). Dessa maneira, o melhor movimento *(movimento A)* é encontrado mais rapidamente do que se fosse procurado pelo Minimax. Analogamente, a avaliação dos filhos de um nó maximizador pode ser interrompida quão logo a avaliação de um desses nós seja superior ao limite máximo estabelecido (*poda beta*).

O algoritmo Minimax com poda Alfa-Beta, aqui referenciado apenas como algoritmo Alfa-Beta, é utilizado neste trabalho para selecionar, na árvore de busca do jogo, o melhor movimento a ser executado pelo multiagente em função do estado corrente do jogo.

2.3.5 Representação do Tabuleiro do Jogo de Damas

Existem diversas maneiras de representar o tabuleiro do jogo de Damas e a escolha de qual representação utilizar depende das necessidades e particularidades de cada jogador. Nesse trabalho, os estados de tabuleiros são representados de duas maneiras:

representação vetorial e representação por características, também conhecida como *Net-FeatureMap* [14].

A representação vetorial é usada para representar o tabuleiro na entrada do algoritmo Alfa-Beta durante o processo de busca pelo melhor movimento e também na representação dos tabuleiros de final de jogo nas BDs de final de jogo. A representação *NetFeatureMap* é usada para: representar o tabuleiro corrente à MLP no momento de calcular a predição dos movimentos; representar os tabuleiros de final de jogo às RNs clusterizadoras durante o processo de agrupamento da BD; representar os tabuleiros durante o processo de obtenção das regras de exceção e, no momento de escolher o EGA para atuar na fase de final de jogo.

Outras maneiras de representar o tabuleiro do jogo de Damas já foram propostas por diferentes jogadores automáticos, dentre as quais pode-se citar a *BitBoards* [19], [36], [14] e a representação espacial [41], [26].

2.3.5.1 Representação Vetorial

A escolha da estrutura de dados usada para representar o tabuleiro em um jogo é fundamental para a eficiência de um jogador automático de Damas. Algoritmos de busca em profundidade (Alfa-Beta, por exemplo) estão presentes nos melhores projetos da história dos jogos de tabuleiro e a escolha adequada da estrutura de dados para representar o tabuleiro afeta, consideravelmente, a velocidade de execução desse tipo de algoritmo. A representação vetorial é recomendada para casos em que algoritmos de busca em profundidade são utilizados, visto que esta é uma representação precisa do estado de tabuleiro.

A representação vetorial do tabuleiro é uma estrutura do tipo *BOARD* implementada como um vetor de 32 elementos do tipo *BoardValues*. Cada elemento do vetor *BOARD* representa uma posição do tabuleiro e cada posição possui um dos valores presentes em *BoardValues* (empty, blackman, redman, blackking, redking). A estrutura *BOARD* é a seguinte:

```
BOARD{
BoardValues p[32]
}
BoardValues{
empty = 0,
blackman = 1,
redman = 2,
blackking = 3,
redking = 4
}
```

Um exemplo da representação vetorial de um determinado tabuleiro é apresentado na Figura 5. As peças simples pretas ocupam a base (posições de 1 a 12) e as peças simples vermelhas ocupam o topo do tabuleiro (posições de 21 a 32).

1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2	2	2	2	2	2
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32

Figura 5 – Representação vetorial do tabuleiro inicial do jogo de Damas

2.3.5.2 Representação *NetFeatureMap*

A *NetFeatureMap* representa um determinado estado de tabuleiro utilizando um conjunto de características que capturam conhecimentos relevantes sobre o domínio do jogo de Damas. Tais características fornecem medidas quantitativas e qualitativas para melhorar a representação de diversas propriedades posicionais das peças sobre o tabuleiro. Esta representação foi proposta por Samuel [14] como resultado da análise do comportamento de especialistas humanos durante vários jogos. O autor implementou um conjunto de 26 características que representam o domínio do jogo de Damas.

A primeira versão do MP-Draughts, utilizada para o início das pesquisas que envolvem o presente trabalho, utiliza um subconjunto de 15 dessas características, as quais estão descritas na Tabela 2. As demais versões desse jogador, originadas a partir das pesquisas aqui realizadas, utilizam subconjuntos dessas 15 características. Cada característica é representada por um valor absoluto correspondente a quantidade de peças que a representa no tabuleiro. Este valor é convertido em *bits* os quais, em conjunto com os *bits* das outras características, constituem a saída do mapeamento *NetFeatureMap*.

A conversão de um determinado tabuleiro vetorial I_i para a representação *NetFeatureMap* é definido pelo mapeamento C:
C: $I_i \longrightarrow \mathbb{N}^n$,
$C(I_i) =< f_1(I_i), ..., f_n(I_i) >$, e
$f_j(I_i) = a$, em que:
n é a quantidade de características f_j $(1 \leq j \leq n)$ usadas para representar o tabuleiro, a representa a presença (se $a > 0$) e quantidade de peças que representam a característica f_j em I_i. Em suma, cada I_i é representado por uma *n-tupla* composta de n atributos, os quais correspondem as características f_1,..., f_n, respectivamente.

A Figura 6 exemplifica a representação de um tabuleiro utilizando o subconjunto de característica da Tabela 2. O atributo f_1 indica que existem 4 peças pretas no centro do tabuleiro (*CentreControl*), enquanto f_2 indica que existem 2 posições livres no centro do

4	2	0	9	2	0	1	1	2	0	0	1	3	0	0
f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15

Figura 6 – Representação *NetFeatureMap* de um determinado tabuleiro do jogo de Damas

Tabela 2 – Subconjunto de características propostas por Samuel que são utilizadas neste trabalho.

Características	Descrição Funcional	Bits
CentreControl	Total de peças pretas no centro do tabuleiro.	3
XCentreControl	Total de quadrados no centro do tabuleiro onde tem peças vermelhas ou para onde elas possam mover.	3
DoubleDiagonal	Total de peças pretas que estão na diagonal dupla do tabuleiro.	4
DiagonalMoment	Total de peças vermelhas que estão localizadas na posição 1 ou na posição 2 de uma diagonal dupla mais as peças passivas que estão na ponta da diagonal.	4
Exposure	Total de peças pretas que são rodeadas por quadrados vazios em diagonal.	3
TotalMobility MOB	Total de quadrados vazios para onde as peças vermelhas podem se mover.	4
Threat	Total de posições para qual uma peça preta pode se mover e assim poder ameaçar uma peça vermelha em um movimento subsequente.	3
Taken	Total de posições para qual uma peça vermelha pode se mover e, assim, poder ameaçar uma peça preta em um movimento subsequente.	3
PieceAdvantage	Contagem de peças em vantagem para o jogador preto (no caso, número de peças que o jogador tem a mais).	4
PieceDisadvantage	Contagem de peças em desvantagem para o jogador preto (no caso, número de peças que o jogador tem a menos).	4
PieceThreat	Total de peças pretas que estão sob ameaça.	3
PieceTake	Total de peças vermelhas que estão sob ameaça de peças pretas.	3
Advancement	Total de peças pretas que estão na $5^{\underline{a}}$ e $6^{\underline{a}}$ linha do tabuleiro menos as peças que estão na $3^{\underline{a}}$ e $4^{\underline{a}}$ linha.	3
Backrowbridge	Indica se existe damas pretas tabuleiro ou se as posições 30 e 32 (bridge) estão ocupadas por peças. vermelhas.	1
Kingcentrecontrol	Total de damas pretas no centro do tabuleiro.	3

tabuleiro para onde uma peça vermelha (vista como peça do oponente) pode ser mover (*XCentreControl*).

2.3.6 Redes Neurais aplicadas ao Agrupamento de Dados

Nesta seção são apresentadas 3 tipos de RNs não-supervisionadas, as quais são utilizadas para agrupamento de dados. Quando se diz que uma RN possui aprendizado não-supervisionado, significa dizer que ela consegue aprender tendo como entrada padrões não rotulados, ao contrário das RNs com aprendizado supervisionado que recebem

um conjunto de treinamento previamente classificado e rotulado.

As RNs aqui descritas, foram utilizadas para obter os *clusters* de estados de tabuleiros, os quais devem ser utilizados para treinar as MLPs que representam os agentes de final de jogo que compõem o MP-Draughts. Tais RNs também são utilizadas para alocar os agentes na fase de final de jogo, em torneios envolvendo o multiagente

2.3.6.1 Redes Neurais Auto-Organizáveis de Kohonen - KSOM

A RN K-SOM, também conhecida como Mapas Auto-Organizados de Kohonen, foi proposta pelo Prof. Teuvo Kohonen [58], [59]. Esta RN possui aprendizado não-supervionado e competitivo, cuja arquitetura estática é pré-definida pelo usuário. A arquitetura de uma KSOM, apresentada na Figura 7, é composta por duas camadas: uma camada de entrada cujos neurônios X_i $(1 \leq i \leq n)$ representam o padrão de entrada I_i e uma camada de saída cujos neurônios Y_j $(1 \leq j \leq m)$ competem para representar o padrão. Os neurônios X_i são completamente conectados aos neurônios Y_j. Os pesos sinápticos que conectam cada X_i a cada Y_j são representados por w_{ij}.

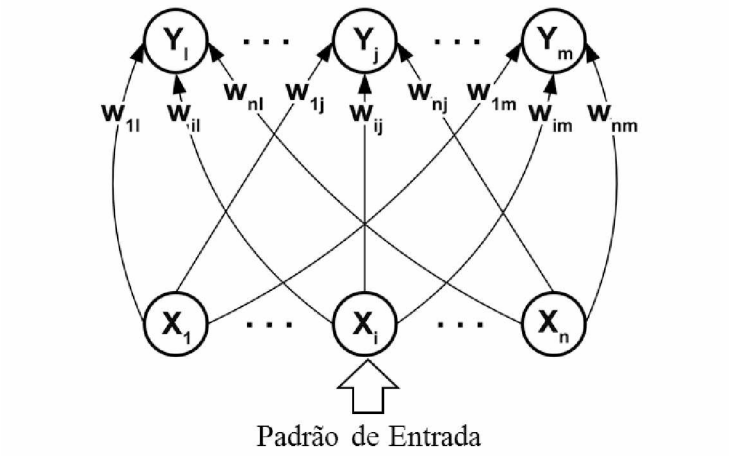

Figura 7 – Arquitetura básica de uma Kohonen-SOM

Para cada padrão de entrada I_i apresentado a KSOM, a mesma identifica o neurônio vencedor Y_J que mais se assemelha ao padrão e atualiza seus pesos e os pesos dos neurônios à ele mais próximos (neurônios vizinhos) a fim de incorporar o padrão. Em outras palavras, quando um padrão I_i é submetido à camada de entrada da KSOM (representado por X_i , ..., X_n), é calculado a medida de similaridade entre I_i e cada Y_j; na sequência, os pesos do neurônio que possui maior similidade ao padrão , denotado por Y_J, são atualizados assim como os pesos de seus vizinhos (numa proporção menor) para representar o padrão.

A medida de similaridade frequentemente utilizada pelas KSOM é a Distância Euclidiana, porém outras medidas podem ser utilizadas [59].

2.3.6.2 Redes Neurais ART 2A - *Adaptive Resonance Theory*

A RN de aprendizado não supervisionado *Adaptive Resonance Theory* (ART) 2A [60] pertence à família das RNs ART, as quais são baseadas na teoria da ressonância adaptativa proposta por Grossberg [61], [62]. Tal RN possui a habilidade de aprender novos padrões sem destruir os conhecimentos obtidos de padrões anteriormente apresentados. Esta característica está relacionada ao dilema da plasticidade/estabilidade, no qual a RN deve ser adaptativa o suficiente para incorporar mudanças ocorridas no ambiente, ao passo que deve ser estável a fim de preservar o conhecimento já adquirido ao longo do tempo [63]. Os neurônios são criados dinamicamente pela ART 2A.

A arquitetura desta RN, apresentada na Figura 8, é composta por 2 subsistemas: um subsistema de atenção e um de orientação. O subsistema de atenção, composto pelas camadas F_0, F_1 e F_2, é responsável por escolher o melhor neurônio para representar o padrão de entrada. O subsistema de orientação controla (através do parâmetro de vigilância ρ) o nível de similaridade entre os padrões representados por um mesmo neurônio de saída. Este subsistema decide se um padrão de entrada deve ser representado pelo neurônio indicado pelo subsistema de atenção.

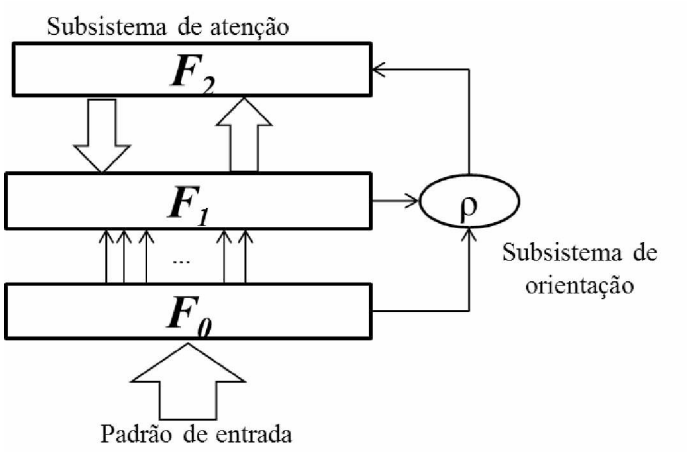

Figura 8 – Arquitetura básica de RN ART

Quando um padrão de entrada I_i é apresentado a ART 2A, ele é pre-processado na camada F_0 e repassado à camada F_1. A camada F_1 calcula a similaridade entre I_i e os neurônios Y_j da camada F_2. O neurônio com maior similaridade, denotado por Y_j, torna-se candidato para representar I_i. Caso a similaridade entre I_i e Y_j seja superior a vigilância estabelecida por ρ, o neurônio candidato é considerado vencedor, Y_J, e se adapta para representar I_i. Caso contrário (a similaridade é inferior a ρ), a RN cria um novo neurônio para representar I_i.

Existem diversas variações da ART 2A, as quais se diferem principalmente pela medida de similaridade utilizada. A ART 2A utiliza similaridade cosseno [64].

2.3.6.3 Redes Neurais SONDE - *Self-Organizing Novelty Detector*

A *Self-Organizing Novelty Detection* (SONDE) é uma RN adaptativa cujo aprendizado é contínuo e não supervisionado. Esta RN foi projetada para integrar os recursos das RN SOM, *Growing When Required* (GWR) e ART para detecção de novidade em fluxos de dados [65], [66]. As unidades básicas de representação de conhecimento da SONDE são neurônios adaptativos, os quais são criados a medida que as novidades são detectadas. A Figura 9(a) ilustra a estrutura de conhecimento de um neurônio desta RN. Cada neurônio y_i é definido pela tendência média (centróide) $\vec{w}_i$ dos padrões de entrada agrupados por y_i , pela dispersão média (raio médio) rad_i dos padrões de entrada em torno do centróide e pelo grau mínimo de similaridade (limiar de ativação do neurônio) α_i para reconhecer novos padrões.

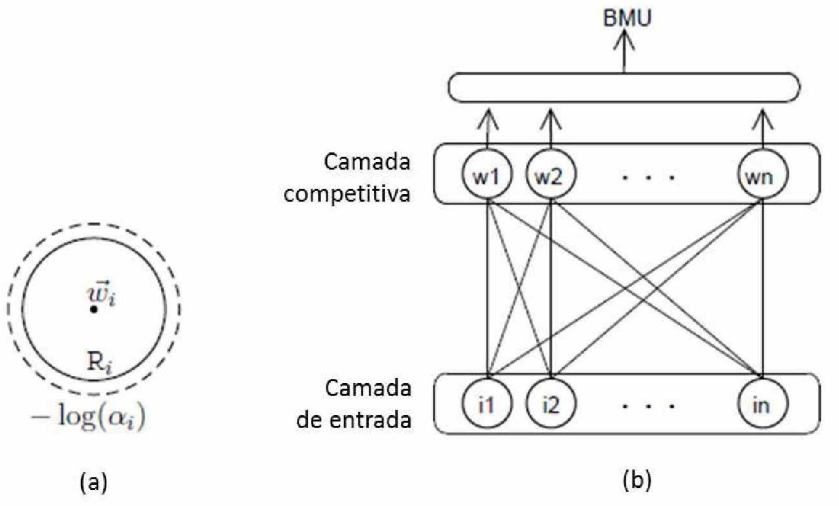

Figura 9 – (a) Exemplo de um neurônio y_i e suas estruturas de conhecimento. (b) Arquitetura da SONDE

A arquitetura da SONDE, representada pela Figura 9 (b), é composta por três camadas: uma camada de entrada e pré-processamento, na qual padrões de entrada são opcionalmente normalizados; uma camada competitiva, na qual os neurônios disputam para representar o padrão de entrada; e uma camada de saída em que o melhor neurônio (do inglês *Best Matching Unit* (BMU)), ou neurônio com maior ativação, é estimulado para representar o padrão de entrada.

O algoritmo da SONDE é apresentado no Algoritmo 1.

Os parâmetros γ, Ω e α_0 possuem valores constantes $\in [0, 1]$ definidos pelo usuário. O γ e o Ω são parâmetros de aprendizagem, os quais definem a influência de padrões do passado na situação corrente da RN. Quando maior os valores desses parâmetros, maior é o grau de esquecimento da SONDE. Isso quer dizer que, quanto maior os valores de γ e Ω, menor é a influência dos padrões do passado no conhecimento corrente da RN. Para cada padrão de entrada $\vec{I}_t$ (opcionalmente normalizado da linha 7), o valor de ativação

Algoritmo 1 SONDE

1: {A dimensão do padrão de entrada $\vec{I}_t$ é n}
2: {K representa o conjunto dos centróides}
3: {Os parâmetros γ, Ω e α_0 são constantes pré-definidas}
4: {Um neurônio y_i é composto por: α_i, rad_i e $\vec{w}_i$}
5: {A ativação do neurônio y_i é dada por a_i}
6: **for all** $\vec{I}_t$,do instante de tempo $t \in \mathbb{N}$ **do**
7: {Fase de normalização do vetor de entrada. Executada quando n > 1}
8: $\vec{I}_t = \frac{\vec{I}_t}{\|\vec{I}_t\|}$
9: {Cálculo da ativação de cada neurônio e busca pelo melhor neurônio BMU}
10: **for all** $\vec{w}_i \in K$ **do**
11: {Cálculo da distância entre $\vec{w}_i$ e $\vec{I}_t$}
12: $dist_i = \|\vec{w}_i - \vec{I}_t\|$
13: {Cálculo da ativação do neurônio corrente y_i}
14: $a_i = exp(-dist_i)$
15: **if** $a_i > a_{BMU}$ **and** $a_i > \alpha_i$ **then**
16: {Encontra o neurônio com maior ativação}
17: $y_{BMU} = y_i$
18: **end if**
19: **end for**
20: **if** BMU foi encontrado **then**
21: {Adaptação do melhor neurônio y_{BMU}}
22: $\vec{w}_{BMU_t} = (1-\gamma) * \vec{w}_{BMU_{t-1}} + \gamma * \vec{I}_t$
23: $rad_{BMU_t} = (1-\Omega) * rad_{BMU_{t-1}} + \Omega * \|\vec{I}_t - \vec{w}_{BMU_{t-1}}\|$
24: $p = \|\frac{rad_{BMU_t} - rad_{BMU_{t-1}}}{max(rad_{BMU_t}, rad_{BMU_{t-1}})}\|$
25: $\alpha_{BMU_t} = min((1+p) * \alpha_{BMU_{t-1}}, exp^{(} - rad_{BMU_t} * (1+p)))$
26: **else**
27: {Cria novo neurônio $y_{new} \in K$}
28: $\vec{w}_{new} = \vec{I}_t$
29: $\alpha_{new} = \alpha_0$
30: $rad_{new} = -ln(\alpha_0)$
31: **end if**
32: **end for**

a_i de cada neurônio da camada competitiva é calculado (linhas 10-14). Se o valor de ativação a_i é maior que o limiar de ativação do neurônio ($a_i > \alpha_i$), o neurônio com maior valor de ativação ($a_i > a_{BMU}$) é eleito como neurônio vencedor BMU (linhas 15-17). O BMU é o neurônio que melhor representa o padrão de entrada. O centróide $\vec{w}_{BMU}$ (linha 22) e o raio médio rad_{BMU} (linha 23) do neurônio vencedor são adaptados para representar $\vec{I}_t$. Esta adaptação é feita utilizando médias móveis exponenciais ponderadas, que provêm adaptação incremental ao conhecimento do nerônio de acordo com as variações presentes nos padrões de entrada [65]. O limiar de ativação α_{BMU} é atualizado para melhor representar o padrão de entrada nas próximas ativações (linha 25). A atualização do α_{BMU} é proporcional ao grau de modificação p do novo raio médio comparado com o raio médio anterior (linha 24). Quando nenhum neurônio é capaz de representar o padrão de entrada $\vec{I}_t$ (caso em que a condição da linha 15 não é satisfeita), um novo neurônio

é criado. O centróide $\vec{w}_{new}$ do novo neurônio é igual ao padrão $\vec{I}_t$ responsável por sua criação e o limiar de ativação α_{new} é igual ao valor pré-definido α_0. O raio médio inicial rad_{new} é igual a $-ln(\alpha_0)$ (linhas 26-31).

2.3.7 Mineração de Dados

A mineração de dados é um campo multidisciplinar que inclui áreas de IA, Estatística, Reconhecimento de Padrões, entre outros. A tarefa de minerar dados permite extrair padrões ou conhecimentos interessantes de BD, os quais são uteis para auxiliar na tomada de decisão em vários domínios de aplicações, tais como a análise de mercado, diagnósticos médicos, controle de produção, jogos, etc [67]. A mineração de dados pode ser classificada em duas categorias: descritiva e preditiva. A mineração descritiva encontra padrões e propriedades que frequentemente estão presentes no dados analisados, enquanto a mineração preditiva realiza indução nos dados de modo a extrair modelos preditivos que possam ser usados para efetuar predições do comportamento de novos dados [68].

Esta seção apresenta 3 técnicas de mineração de dados, sendo uma descritiva e duas preditivas, que foram utilizadas para o desenvolvimento deste trabalho.

2.3.7.1 Mineração de Padrões Frequentes

Mineração de padrões frequentes é um tipo de mineração descritiva que extrai padrões que ocorrem frequentemente numa BD. Geralmente, esses padrões representam informações potencialmente úteis, não triviais e previamente desconhecidas pelos especialistas do domínio. O problema de minerar padrões frequentes de um BD pode ser descrito como: dado uma BD D contendo as transações $T_1, ..., T_N$, encontre todos os padrões P que estão presentes em pelo menos numa fração s dessas transações. A fração s é denominada de suporte mínimo, o qual pode representar a frequência absoluta (valor absoluto) ou a frequência relativa com que as transações ocorrem na BD. Cada transação T_i na BD corresponde a uma tupla na BD [69]. No modelo original de mineração de padrões frequentes, proposto em [70], o problema de encontrar relacionamento (ou associações) entre os itens de uma BD foi proposto como sendo um "segundo estágio" do processo de mineração, o qual é derivado dos itens frequentes desta BD. Neste modelo, inicialmente é feita uma mineração descritiva e, a partir dos padrões frequentes encontrados na BD, é feito uma mineração preditiva com o objetivo de encontrar relacionamentos entre os itens que representam esses padrões, os quais poderiam ser usados para analisar e predizer o comportamento de novos dados.

Neste trabalho entretanto, foi aplicado apenas o "primeiro" estágio da mineração proposta em [70], o qual obtém os itens frequentes de uma determinada BD. Tais itens descrevem a frequência com que determinadas características ocorrem na BD.

2.3.7.2 Mineração de Regras de Exceção

Extrair conhecimento interessante a partir grande quantidade de dados é uma das principais preocupações da mineração de dados. Uma classe de conhecimento que atrai muito a atenção das pesquisas na área são as regras de exceção [71], [72], [73], [74], [75] [76], [77]. Uma exceção pode ser definida como algo diferente da maioria, que contradiz o senso comum e geralmente representa algum conhecimento interessante.

As regras de exceção permitem que a precisão de regras gerais sejam otimizadas, uma vez que representam conhecimentos excepcionais e interessantes não representados pelas regras gerais. Uma regra de exceção pode ser expressa pela combinação de uma regra geral e associado a essa regra, uma exceção. Enquanto a regra geral representa o conhecimento comum, a regra de exceção representa conhecimentos locais que contradizem esse conhecimento. O conceito de localidade das regras de exceção está relacionado ao fato delas serem obtidas a partir de situações especiais que o conhecimento geral não foi capaz de representar, e de serem usadas para complementá-lo. Uma regra geral possui alto valor de suporte e confiança, enquanto regras de exceção possuem baixo suporte e confiança semelhante às regras gerais [76]. O suporte representa a frequência relativa com que as instâncias cobertas por uma regra ocorrem na BD e a confiança representa a precisão de cobertura dessa regra.

A mineração de regras de exceção foi proposta por Hussain [76], o qual define uma regra de exceção de acordo com o apresentado na Tabela 3, onde A e B representam um item ou um conjunto de itens e B também representa disjunções não vazias de restrições sobre os itens. Por exemplo, se houver uma regra geral "*se a pessoa está desempregada, então não lhe é dado crédito*" ($A \rightarrow X$), uma exceção seria "*se a pessoa está desempregada e seu cônjuge está empregado, então lhe é dado crédito*" ($A \wedge B \rightarrow \neg X$). Neste caso, a regra "*seu cônjuge está empregado, então lhe é dado crédito*" ($B \rightarrow \neg X$) representa uma regra de referência, a qual explica a exceção. Regras de referência podem ter baixo suporte e/ou baixa confiança e são regras difíceis de serem descobertas (mineradas).

Tabela 3 – Estrutura das regras com exceções proposta por Hussain.

$A \rightarrow X$	Regra geral: alto suporte, alta confiança.
$A \wedge B \rightarrow \neg X$	Regra de exceção: baixo suporte, alta confiança.
$B \rightarrow \neg X$	Regra de referência: baixo suporte e/ou baixa confiança.

Um conjunto de regras isoladas é pouco intuitivo e dificulta o entendimento de qualquer problema. Geralmente os indivíduos expressam o conhecimento em termos de padrões gerais e casos especiais. Desse modo, o par de regras [regra geral, regra de referência], que formam as exceções são mais confortáveis e familiares no que diz respeito as necessidades

de entendimento desses indivíduos, sendo que as regras gerais são verificadas primeiro e exceções à essas regras são modeladas como um posterior refinamento do conhecimento induzido pela regra geral. Exceções podem, por exemplo, representar posições raras de damas sobre o tabuleiro, as quais quando identificadas podem conduzir o jogo para uma situação de vitória.

O conceito de regras de exceção proposto por Hussain é utilizado neste trabalho para tratar o baixo desempenho da RN em alocar os agentes em algumas situações especiais do jogo.

2.3.7.3 Mineração de Árvore de Decisão - C4.5

Árvore de Decisão (AD) é um dos métodos mais utilizados da literatura de mineração de dados para tratar problemas de classificação. Este método pertence a família de algoritmos "*dividir para conquistar*", o qual divide um conjunto de treinamento, até que cada subconjunto obtido deste particionamento contenha apenas exemplo de uma mesma classe [78]. A construção de uma AD pode ser descrita como um procedimento recursivo, o qual divide o conjunto de dados em subconjuntos de exemplos cada vez mais puros em relação a uma determinada classe. Sua estrutura é composta por: *nós folhas*, que correspondem as classes do problema; *nós de decisão*, os quais correspondem aos nós internos da árvore responsáveis por testar as condições sobre os atributos e; *arestas*, que conectam os nós de decisão que satisfazem determinadas condições até a classe que os representam [67]. Para classificar um novo exemplo numa AD, um caminho é traçado a partir do nó raiz (neste caso, representa um nó de decisão), descendo pelas arestas de acordo com os resultados das condições, até chegar em um nó folha, que representa a classe de predição do exemplo [68]. Uma AD pode ser facilmente mapeada em um conjunto de regras, transformando cada ramo da árvore (cada caminho da raiz até um dos nós folha) em uma regra. As regras traduzidas são disjuntas, de modo que apenas uma única regra dispara quando um novo exemplo é classificado. Tais árvores apresentam como principal vantagem estruturas simples e de grande legibilidade, as quais podem ser facilmente entendidas e usadas diretamente pelo usuário. O grande representante desse grupo é o algoritmo C4.5.

O C4.5 foi proposto por Quilan [79] como um melhoramento do algoritmo ID3 [80]. As principais vantagens do C4.5 em relação ao seu sucessor são: lida com atributos contínuos e com valores desconhecidos; lida com problemas em que os atributos possuem custos diferenciados; utiliza a medida *gain ratio* (razão do ganho) para selecionar o atributo que melhor divide os exemplos, gerando árvores mais precisas e menos complexas; efetua pós poda nas árvores geradas. O C4.5 é um dos métodos de AD mais consagrados na literatura, o qual é considerado um padrão na comparação de algoritmos de aprendizagem simbólica. Este algoritmo têm obtido ótimos resultados em problemas de classificação ao longo de décadas.

Capítulo 3

MP-Draughts - Proposta de Arquitetura para a Fase de Final de Jogo baseada em Redes Neurais Adaptativas

O sistema multiagente MP-Draughts apresentado neste capítulo corresponde a uma otimização da versão preliminar do jogador apresentada na Seção 2.2.2, a qual adapta a arquitetura do jogador para a fase de final de jogo. Na versão preliminar deste jogador, o módulo de agrupamento é representado por uma KSOM baseada em distância Euclidiana (KSOM-DE). Neste agente, devido a arquitetura pré-definida da KSOM, o número de *clusters* para minerar a BD de treinamento foi definido manual e empiricamente em 25, produzindo assim, 25 agentes para representar a fase de final de jogo (EGAs). Contudo, não existe nenhuma garantia de que este número é adequado para representar os perfis de jogo existentes nesta fase. Da mesma maneira, não foi feito uma investigação sobre os dados (estados de tabuleiros) para saber se a distância Euclidiana (DE) era a melhor medida para estimar a similaridade entre eles. A quantidade de 25 foi estabelecida após uma série de testes variando a quantidade de *clusters* gerados pela KSOM-DE, uma vez que por não possuir nenhum conhecimento do jogo e também por não contar com o auxílio de especialistas no domínio, os autores deste trabalho não tinham conhecimento suficiente para tomar tal decisão.

Neste sentido, a versão do MP-Draughts apresentada neste capítulo otimiza a anterior com as seguintes contribuições: investigação de qual medida é mais adequada para estimar a similaridade entre os estados de tabuleiros e, a substituição da arquitetura pré-fixada da KSOM-DE por uma arquitetura baseada em RN adaptativa. Tal RN define, dinamicamente, a quantidade mais apropriada de *clusters* para representar a BD de final de jogo e, consequentemente, define a melhor arquitetura do multiagente para representar esta fase do jogo. A alocação do EGA que deve atuar na fase de final de jogo também é feita por

essa RN. Como contribuição para a área de AM não supervisionada e para o multiagente jogador, este trabalho propõe uma nova versão de RN adaptativa, a ASONDE, a qual adapta sua versão original SONDE [65], de modo a torná-la apta para lidar com BDs finitas e estáveis. A ASONDE, baseada em similaridade cosseno, foi a RN adaptativa dentre as várias investigadas neste capítulo, que obteve a melhor proposta de arquitetura de final de jogo para o MP-Draughts.

Para facilitar o entendimento do que é fase inicial/intermediária de jogo e fase de final de jogo citada neste e nos próximos capítulos deste trabalho, considere a expressão *"fase inicial/intermediária de jogo"* como sendo os tabuleiros do jogo com, no mínimo, 15 peças e a expressão *"fase de final de jogo"* como sendo os tabuleiros com, no máximo, 14 peças.

As próximas seções deste capítulo estão organizadas da seguinte maneira: a Seção 3.1 apresenta a arquitetura geral do MP-Draughts e sua dinâmica de atuação em jogos contra outros jogadores; as Seções 3.2 e 3.3 explicam os módulos que representam os agentes especialistas do multiagente assim como o processo de aprendizagem e de busca desses agentes; a Seção 3.4 apresenta o processo de investigação da medida de similaridade e da RN adaptativa que devem compor o módulo *RN Adaptativa* da arquitetura geral do jogador, sendo que esta seção contém as principais contribuições do capítulo, que contemplam o cumprimento de parte objetivo geral e do objetivo específico 1 desta pesquisa (apresentados na Seção 1.2).

3.1 Arquitetura do MP-Draughts

MP-Draughts é um sistema multiagente não supervisionado jogador de Damas cuja arquitetura é baseada em RNs adaptativas e MLPs [6]. Os agentes que o compõem são especializados em fases distintas do jogo: um agente é especialista nas fases iniciais e intermediárias de jogo (*IIGA*) e o restante são especialistas na fase de final de jogo (*EGAs*). Cada agente corresponde a uma MLP que aprende a jogar por reforço através dos métodos TD(λ). O algoritmo Alfa-Beta combinado com Tabela de Transposição e Aprofundamento Iterativo é utilizado pelo multiagente para indicar o melhor movimento a ser executado considerando o estado de tabuleiro corrente do jogo. Os estados de tabuleiros são representados tanto por características *NetFeatureMap* quanto vetorialmente, dependendo do momento que em o sistema se encontra.

A arquitetura do MP-Draughts, apresentada na Figura 10, é composta por 4 módulos:

- **Módulo IIGA - *Initial/ Intermediate Game Agent***: corresponde ao agente especialista nas fases iniciais e intermediárias do jogo de Damas;
- **Módulo EGAs Treinados - *EndGame Agents***: representa o conjunto dos agentes especialistas nas fases de final de jogo. Cada agente deste módulo é treinado para ser especialista num determinado perfil (*cluster*) de final de jogo;

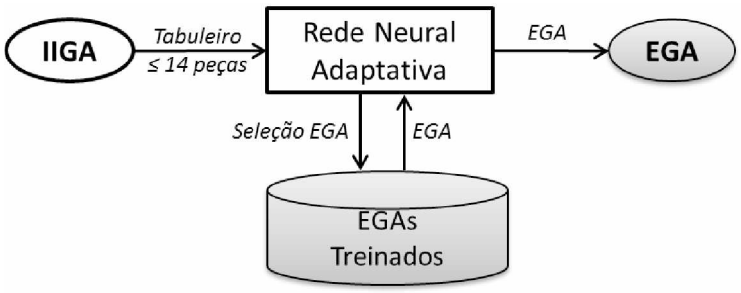

Figura 10 – Arquitetura geral do MP-Draughts: sistema multiagente jogador de Damas

- **RNs Adaptativas**: a RN adaptativa que compõe este módulo possui duas responsabilidades distintas: primeiro, ela é responsável por obter os *clusters* de treinamento dos EGAs, minerando uma BD de estados de tabuleiros de final de jogo. Segundo, é responsável por alocar, dentre os EGAs treinados, àquele que possui maior capacidade de atuar na fase de final de jogo, dado a situação de final de jogo a ela apresentada;

- **Agente de final de jogo - EGA**: corresponde ao agente de final de jogo escolhido pela RN adaptativa, para assumir o jogo na fase final e o conduzí-lo até o final. Esse agente, por ter sido treinado num *cluster* cujo perfil se assemelha ao estado corrente do jogo, é considerado pela RN o mais habilitado para atuar no jogo.

A dinâmica de jogo adotada pelo MP-Draughts em jogos contra outros jogadores é a seguinte: o IIGA é o agente que atua no jogo desde o início (tabuleiro 8×8 completo) até a fase de final de jogo. Neste momento, o estado corrente do jogo, representado por *NetFeatureMap*, é apresentado à RN adaptativa que aloca o EGA que melhor representa o estado corrente do jogo. Este EGA conduz o jogo até o final.

Vale ressaltar que outras duas estratégias de atuação dos EGAs foram testadas. Na primeira delas, a cada movimento a ser executado pelo MP-Draughts na fase de final de jogo era verificado qual EGA era o mais apropriado para a ação, o qual deveria indicar o melhor movimento. Nesta estratégia cada movimento poderia ser executado por um EGA diferente. Na segunda estratégia, a cada movimento a ser executado nesta fase do jogo eram escolhidos os K EGAs mais similares ao estado corrente do jogo, os quais indicavam o que consideravam ser o melhor movimento para o tabuleiro. Caso alguma indicação representasse a maioria, esta era considerada como a indicação do melhor movimento, caso contrário era feito uma escolha aleatória considerando todas as indicações. Foram feitos testes para K igual a 3 e 5. Contudo, nenhuma dessas estratégias conseguiram resultados melhores que a estratégia de escolher um único agente para atuar nesta fase do jogo, o que faz muito sentido quando se considera a estratégia geral de um jogo de Damas. Durante um jogo, um agente costuma escolher um movimento analisando os movimentos futuros que este movimento pode lhe proporcionar, estabelecendo uma estratégia sequencial de movimentos. Se o próximo movimento deste agente for executado por um outro agente,

a estratégia estabelecida pelo primeiro pode ser diferente da estabelecida pelo segundo, de modo que a estratégia de ambos pode ser perdida. Tal situação pode ser ainda pior quando a quantidade de agentes atuando num mesmo movimento é aumentada.

3.2 Agente de Início e Meio de Jogo - *Módulo IIGA*

O IIGA corresponde a uma MLP treinada para ser especialista nas fases iniciais e intermediárias do jogo de Damas. Este agente aprende a jogar por reforço utilizando os métodos TD(λ) para ajustar os pesos sinápticos da MLP durante o processo de treinamento. O treinamento é feito utilizando a estratégia de *self-play* com clonagem, durante vários ciclos de jogos de treinamento realizados a partir do tabuleiro inicial do jogo. O algoritmo Alfa-Beta combinado com TT e AI é utilizado pelo agente para buscar o melhor movimento a ser executado no estado de tabuleiro corrente do jogo.

A arquitetura da MLP é composta por 3 camadas, sendo uma de entrada com 48 neurônios, uma oculta com 20 neurônios e uma de saída com um único neurônio, cuja saída indica a avaliação de um determinado movimento no jogo. Cada neurônio da MLP está conectado a todos os outros das camadas subsequentes (fortemente conectada). Na primeira camada são representados os valores quantitativos de todas as características *NetFeatureMap* utilizadas para representar o tabuleiro do jogo (somatório da coluna *Bits* da Tabela 2). Na camada oculta foi mantida a mesma quantidade de neurônios utilizada pelas versões predecessoras do MP-Draughts. Isso foi feito para manter a rastreabilidade das otimizações efetuadas neste jogador, visto que a estrutura da MLP não foi alterada em nenhuma das versões do jogador investigadas neste trabalho.

A Figura 11 apresenta o processo de aprendizagem do agente em jogos de treinamento. Resumindo, sempre que o agente precisa escolher um novo movimento m_{t+1} (em que $_{t+1}$ representa um estado futuro ao tempo atual t), o estado de tabuleiro corrente I_t (que representa o estado corrente no tempo t, cuja predição P_t foi calculada no ciclo anterior relacionado à escolha do último movimento m_t) é apresentado ao algoritmo Alfa-Beta (#1), o qual gera a árvore de busca do jogo cuja raiz é o estado I_t. Cada estado de tabuleiro associado aos nós folha da árvore de busca é convertido na representação *NetFeatureMap* (#2), e apresentado na camada de entrada da MLP (#3). A MLP avalia cada um desses nós folha e retorna um valor (*predição*) que indica o quanto cada estado folha é favorável para o agente. Este valor é retornado para o a algoritmo Alfa-Beta (#4), que escolhe o melhor movimento m_{t+1} (movimento com maior predição) a ser executado em I_t (#5). O agente executa este movimento (#6). O novo estado de tabuleiro I_{t+1} é convertido na representação *NetFeatureMap* (#7) e apresentado à MLP que o avalia gerando uma predição P_{t+1} (#8). As predições P_{t+1} referente ao estado I_{t+1} (novo estado) e P_t referente ao estado I_t (antigo estado corrente cuja predição foi calculada no ciclo anterior) são usadas pelo Módulo de *Aprendizagem TD(λ)* para reajustar os pesos da

MLP (#9 e #10). Em seguida, o estado I_{t+1} é reavaliado pela MLP (agora com os pesos reajustados)(#11) gerando um segundo valor de predição P'_{t+1} (#12). O valor de P'_{t+1} instanciará a variável P_t (#13) para o próximo ciclo de treinamento. O novo estado I_{t+1} (produzido pelo movimento m_{t+1}) passa a ser o próximo estado corrente I_t (#14) do ciclo de treinamento com predição P_t (instanciada pela predição P'_{t+1}).

Durante os *jogos de não treino* (em que os pesos da MLP não são atualizados), o processo é semelhante ao ciclo de treinamento apresentado na Figura 11, exceto pelo fato de desconsiderar o Módulo de *Aprendizagem TD(λ)*(passos #9, #10, #11 e #12) e, na etapa #13 a variável P_t é instanciada com a própria predição P_{t+1} referente ao novo estado I_{t+1} (produzido pelo movimento m_{t+1}), pois neste caso não há geração de um novo valor P'_{t+1} (que ocorre apenas quando a MLP está treinando).

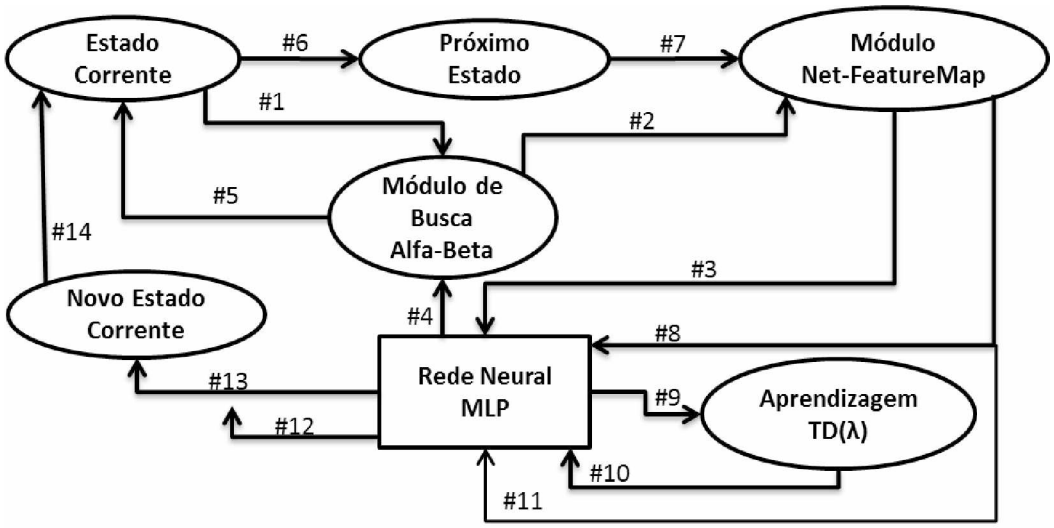

Figura 11 – Processo de aprendizagem do agente

As subseções que seguem (3.2.1, 3.2.2 e 3.2.3) descrevem com mais detalhes o processo de reajuste dos pesos da MLP pelo método das TD(λ) durante os jogos de treinamento, a estratégia de treinamento por *self-play* com clonagem e o processo de busca pelo melhor movimento.

3.2.1 Reajuste dos Pesos da MLP

O reajuste dos pesos da MLP é executado durante seu processo de aprendizagem, ou seja, a medida que o agente joga contra o seu clone por meio do treinamento por *self-play* com clonagem, os pesos da RN são reajustados pelo método TD(λ). O reajuste é feito de acordo com a escolha dos melhores movimentos efetuados pelo agente e os estados resultantes desses movimentos. Considerando um conjunto de movimentos que o agente executa durante um jogo de treinamento (m_0,..., m_{i-1}, m_i, m_{i+1}, m_f), em que m_f se refere ao estado final do jogo (veja Seção 2.3.3)), um reforço final ainda é fornecido pelo ambiente à MLP , de acordo com o resultado obtido da execução do movimento m_f.

O agente seleciona o melhor movimento m_{t+1} a ser executado a partir de um estado I_t com o auxílio do procedimento de busca Alfa-Beta e dos pesos atuais da MLP. O estado I_{t+1} resulta do movimento m_{t+1} sobre o estado I_t. A partir de então, o estado I_{t+1} é mapeado na entrada da rede neural e tem sua predição P_{t+1} calculada (esta predição corresponde ao valor de saída O_t no neurônio da última camada da MLP, veja Figura 2). Os pesos da MLP são reajustados com base na diferença entre as predições P_{t+1} e P_t (calculada anteriormente para o estado I_t). Após o fim de cada jogo de treino, um reforço final é fornecido pelo ambiente informando o resultado obtido pelo agente jogador em função da sequência de movimentos que executou (+1 para vitória, -1 para derrota e 0 para empate).

Formalmente, o cálculo do reajuste dos pesos é definido pela equação do método TD(λ) [54]:

$$\begin{aligned} w_{ij}^{(l)} &= w_{ij}^{(l)}(t-1) + \Delta w_{ij}^{(l)}(t) \\ &= w_{ij}^{(l)}(t-1) + \alpha^{(l)}.(P_{t+1} - P_t).\sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k \\ &= w_{ij}^{(l)}(t-1) + \alpha^{(l)}.(P_{t+1} - P_t).elig_{ij}^{(l)}(t), \text{ onde:} \end{aligned} \tag{4}$$

- $\alpha^{(l)}$ é o parâmetro da taxa de aprendizagem na camada l. Foi utilizado a mesma taxa de aprendizagem para todas as conexões sinápticas de uma mesma camada l;
- $w_{ij}^{(l)}(t)$ representa o peso sináptico da conexão entre a saída do neurônio i da camada l e a entrada do neurônio j da camada $(l+1)$ no instante de tempo t. A correção aplicada a esse peso no instante de tempo t é representada por $\Delta w_{ij}^{(l)}(t)$;
- o termo $elig_{ij}^{(l)}(t)$ é único para cada peso sináptico $w_{ij}^{(l)}(t)$ da RN e representa o traço de eligibilidade das predições calculadas pela RN para os estados resultantes dos movimentos executados pelo agente desde o instante de tempo 1 do jogo até o instante de tempo t;
- $\nabla_w P_k$ representa a derivada parcial de P_k em relação aos pesos da RN no instante k. Cada predição P_k é uma função dependente do vetor de entrada $X\vec{(k)}$ e do vetor de pesos $W\vec{(k)}$ da RN no instante de tempo k.
- O termo λ^{t-k}, para $0 \leq \lambda \leq 1$, tem o papel de fornecer uma "pesagem exponencial" para a taxa de variação das predições calculadas em k passos anteriores de t. Quanto maior for λ, maior o impacto dos reajustes anteriores ao instante de tempo t sobre o reajuste dos pesos $w_{ij}^{(l)}(t)$.

O processo de reajuste dos pesos por Diferenças Temporais TD(λ) é descrito nas seguintes etapas:

1. o vetor $W\vec{(k)}$ de pesos é inicializado aleatoriamente;

2. as eligibilidades associadas aos pesos da RN são inicialmente nulas;

3. dadas duas predições sucessivas P_t e P_{t+1}, referentes a dois estados consecutivos I_t e I_{t+1}, calculadas em consequência de movimentos executados pelo agente durante o jogo, define-se o sinal de erro pela equação:

$$e(t) = (\gamma P_{t+1} - P_t), \tag{5}$$

onde o parâmetro γ é uma constante de compensação da predição P_{t+1} em relação a predição P_t;

4. cada eligibilidade $elig_{ij}^{(l)}(t)$ está vinculada a um peso sináptico $w_{ij}^{(l)}(t)$ correspondente. Assim, as eligibilidades vinculadas aos pesos da camada l, para $0 \leq l \leq 1$, no instante de tempo t $elig_{ij}^{(l)}(t)$ são calculadas observando as equações dispostas a seguir:

   - para os pesos associados às ligações diretas entre as camadas de entrada ($l = 0$) e saída ($l = 2$):

$$elig_{ij}^{(l)}(t) = \lambda.elig_{ij}^{(l)}(t-1) + g'(P_t).a_i^{(l)}, \tag{6}$$

   em que λ tem o papel de fornecer uma "pesagem exponencial" para a taxa de variação das predições calculadas em k passos anteriores de t; $a_i^{(l)}$ é o sinal de saída do neurônio i na camada l; $g'(x) = (1 - x^2)$ representa a derivada da função de ativação (tangente hiperbólica) [39].

   - para os pesos associados às ligações entre as camadas de entrada ($l = 0$) e a oculta ($l = 1$):

$$elig_{ij}^{(l)}(t) = \lambda.elig_{ij}^{(l)}(t-1) + g'(P_t).w_{ij}^{(l)}(t).g'(a_j^{(l+1)}).a_i^{(l)}, \tag{7}$$

   onde $a_j^{(l+1)}$ é o sinal de saída do neurônio j na camada oculta $(l + 1)$;

   - para os pesos associados as ligações entre as camadas oculta ($l = 1$) e de saída ($l = 2$):

$$elig_{ij}^{(l)}(t) = \lambda.elig_{ij}^{(l)}(t-1) + g'(P_t).a_i^{(l)}; \tag{8}$$

5. calculado as eligibilidades, a correção dos pesos $w_{ij}^{(l)}(t)$ da camada l, para $0 \leq l \leq 1$, é efetuada através da seguinte equação:

$$\Delta w_{ij}^{(l)}(t) = \alpha^{(l)}.e(t).elig_{ij}^{(l)}(t), \tag{9}$$

onde o parâmetro de aprendizagem $\alpha^{(l)}$ é definido como:

$$\alpha^{(l)} = \begin{cases} \dfrac{1}{n}, & \text{para } l{=}0 \\ \\ \dfrac{1}{20}, & \text{para } l{=}1 \end{cases}$$

6. existe um problema típico associado ao uso de RNs, que é o fato da convergência estar assegurada para um mínimo local do erro e não necessariamente para o mínimo global do erro. Quando a superfície de erro é "bem comportada" isto não representa um problema, mas quando a superfície apresenta muitos mínimos locais, a convergência não é assegurada para o melhor valor. Neste sentido, utilizou-se o termo momento μ para tentar solucionar esse tipo de problema. Para isso, foi empregado uma checagem de direção na Equação 4, ou seja, o termo momento μ é aplicado somente quando a correção do peso atual $\Delta w_{ij}^{(l)}(t)$ e a correção anterior $\Delta w_{ij}^{(l)}(t-1)$ estiverem na mesma direção. Portanto, a equação final TD(λ) utilizada para calcular o reajuste dos pesos da rede neural na camada l, para $0 \leq l \prec 1$, é definida por:

$$w_{ij}^{(l)}(t) = w_{ij}^{(l)}(t-1) + \Delta w_{ij}^{(l)}(t); \tag{10}$$

onde $\Delta w_{ij}^{(l)}(t)$ é obtido nas seguintes etapas:

a) calcule $\Delta w_{ij}^{(l)}(t)$ pela Equação 4;

b) se $(\Delta w_{ij}^{(l)}(t) > 0$ e $\Delta w_{ij}^{(l)}(t-1) > 0)$ ou $(\Delta w_{ij}^{(l)}(t) < 0$ e $\Delta w_{ij}^{(l)}(t-1) < 0)$ (ambos na mesma direção), faça:

$$\Delta w_{ij}^{(l)}(t) = \Delta w_{ij}^{(l)}(t) + \mu \Delta w_{ij}^{(l)}(t-1);$$

Observe que o termo momento μ é utilizado para reforçar tendências de estabilização nas direções dos reajustes dos pesos já manifestadas em tempos anteriores e mantidas no instante presente. Caso não haja tal tendência, a parcela do termo momento não é aplicada (o que "freia" o processo de reajuste dos pesos).

3.2.2 Estratégia de Treinamento do Agente

O treinamento do IIGA é realizado em 3 sessões de 500 jogos, sendo que o agente joga metade dos jogos (250 jogos) com as peças pretas e a outra metade com as peças vermelhas. Esta estratégia de trocar as cores das peças do agente é interessante por 2 motivos: primeiro, treinar o agente para jogar em ambas as situações e; segundo, porque algumas características da representação *NetFeatureMap* estabelecem restrições relacionadas às cores das peças, as quais tendem a beneficiar o agente que joga com as peças pretas. Por exemplo, a característica *CentreControl* contabiliza a quantidade de peças pretas no centro do tabuleiro.

A estratégia de treinamento por *self-play* com clonagem consiste em treinar um agente por vários jogos (no caso, 500) contra uma cópia de si próprio. A medida que o jogador aumenta seu desempenho até ponto de conseguir vencer sua cópia, uma nova clonagem é realizada e o jogador passa a treinar contra esse novo clone. O processo se repete por um número pré-determinado de ciclos de treinamento (no caso, 3).

Na prática, o treinamento do agente se divide nas seguintes etapas:

1. primeiro, os pesos da MLP que representa o agente *opp1* (nome atribuído a MLP original) são gerados aleatoriamente;

2. antes de iniciar a sessão de treinamento, a MLP *opp1* é clonada para gerar seu primeiro clone *opp1-clone*;

3. inicia-se então a sessão de treinamento da MLP *opp1* que joga contra a MLP *opp1-clone*. As RNs efetuam uma sessão de n jogos de treinamento (no caso, $n = 500$), em que somente os pesos da *opp1* são reajustados durante esses jogos;

4. ao final da sessão de treinamento, dois jogos-testes são realizados para verificar qual das MLPs é a melhor. Caso o desempenho da *opp1* (MLP com reajuste) seja superior ao desempenho da *opp1-clone*, é feita a cópia dos pesos da *opp1* para a *opp1-clone*. Caso contrário, os pesos da MLP original *opp1-clone* permanecem inalterados para a próxima sessão de treinamento;

5. volte para etapa 3 e execute uma nova sessão de n jogos de treinamento entre *opp1* e o seu último clone *opp1-clone*. Repita o processo até que o número máximo de sessões de treinamento seja alcançado. Observe que ao fim de cada sessão de treinamento pode ser gerado um clone de *opp1* (caso ele seja superior a sua versão sem reajustes);

6. Ao final de todas as sessões de treinamento, realiza-se um torneio de 2 jogos entre a ultima versão do *opp1* (obtida na última sessão) e todos os seus clones obtidos em sessões anteriores. Finalmente, o vencedor deste torneio é considerada a melhor MLP para representar o agente.

3.2.3 Processo de Busca utilizado pelo Agente

O jogo de Damas pode ser visto como uma árvore de possíveis estados de tabuleiros, sendo que cada nó representa um estado do jogo e cada ramo representa um possível movimento [51]. Cada vez que o agente precisa executar um movimento, o algoritmo de busca é usado para obter a árvore de busca do jogo e escolher o melhor movimento. A raiz da árvore de busca corresponde ao estado corrente do jogo, I. O próximo nível da árvore (profundidade 1) corresponde aos possíveis estados que podem ser obtidos a partir de movimentos válidos executados em I. Os demais estados da árvore (profundidade 2, ..., d) são obtidos sucessivamente, a partir de movimentos válidos, até atingir o nível de profundidade máxima d. A profundidade d é controlada por um valor previamente estabelecido e/ou pelo tempo que o agente tem para efetuar o movimento. A MLP calcula a predição P para cada estado pertencente à profundidade d da árvore de busca e, essas predições são retornadas para o algoritmo de busca, que indica ao agente o melhor movimento a ser executado em I.

O IIGA utiliza o algoritmo de busca Alfa-Beta combinado com TT e AI para efetuar a busca pelo melhor movimento, dado o estado corrente do jogo. A profundidade máxima estabelecida para a busca foi de $d = 8$, sendo que para valores maiores, o tempo de espera para a execução de um movimento (relacionado ao tempo de busca pelo movimento) fica impraticável em alguns casos.

As próximas subseções explicam os detalhes do algoritmo de busca Alfa-Beta, assim como sua integração com a TT e o AI.

3.2.3.1 O algoritmo Alfa-Beta

O algoritmo Alfa-Beta pode ser resumido como um procedimento recursivo que escolhe o melhor movimento a ser executado fazendo uma busca em profundidade, da esquerda para a direita, na árvore do jogo. Existem duas versões do algoritmo Alfa-Beta, versão *fail-soft* e *hard-soft*, as quais se diferem pelo valor de predição P retornado para um estado *quando ocorre uma poda*. A versão *fail-soft* sempre retorna o valor real da predição para o estado avaliado, independente se houve ou não poda. Já a versão *hard-soft* retorna como predição para o estado, o valor correspondente ao limite (*min* ou *max*) que efetuou a poda, desconsiderando a predição real do estado. Essa característica da versão *hard-soft* restringe sua combinação com a TT, uma vez que o valor de predição de um estado pode não corresponder a predição real para o mesmo (no caso de ocorrer poda). Caso o valor de predição retornado pela versão *hard-soft* seja armazenado na TT e recuperado num momento futuro, este valor provavelmente não corresponderá a predição real do estado e o algoritmo poderá a efetuar um movimento diferente do que seria executado caso o algoritmo não recuperasse a predição da TT. Tal fato, faz com que os movimentos executados pelo algoritmo *hard-soft* não sejam compatíveis quando comparado com sua versão sem a TT, o que inviabiliza sua combinação com a TT.

Dessa maneira, como a TT compõe o módulo de busca do agente, a versão utilizada neste trabalho será a *fail-soft*. O Algoritmo 2 apresenta a versão *fail-soft* do algoritmo Alfa-Beta, que é explicado na sequência:

- **Linha 1:** para a escolha do melhor movimento, os seguintes parâmetros de entrada são fornecidos ao Alfa-Beta: o estado corrente do jogo, I, a profundidade d de busca, o valor *alfa* que representa o limite inferior do intervalo de busca e o valor *beta* que representa o limite superior do intervalo de busca; *alfa* e *beta* são inicializados com os valores *-infinito* e *+infinito*, respectivamente, e seus vão sendo atualizados a medida que os estados folhas da árvore de busca vão sendo avaliados. O parâmetro de saída *bestmove* corresponde ao melhor movimento (m) a ser executado sobre o estado corrente I.

- **Linhas 2 a 4:** por se tratar de um procedimento recursivo, exige-se uma condição de parada: verificar se o estado do tabuleiro I é uma folha da árvore, ou seja, se não

Algoritmo 2 *Fail-soft* Alfa-Beta

```
1: alfaBeta(node:I, int:d, int:alfa, int:beta, move:bestmove)
2: if leaf(I) or d=0 then
3:    return evaluate(I)
4: end if
5: if I is a max node then
6:    besteval := alfa
7:    for each child of I: do
8:       v := alfaBeta(child,d-1,besteval,beta,bestmove)
9:       if v > besteval then
10:         besteval:= v
11:         thebest = bestmove
12:      end if
13:   end for
14:   if besteval >= beta then
15:      return besteval
16:   end if
17:   bestmove = thebest
18:   return besteval
19: end if
20: if I is a min node then
21:   besteval := beta
22:   for each child of I: do
23:      v := alfaBeta(child,d-1,alfa,besteval,bestmove)
24:      if v < besteval then
25:         besteval:= v
26:         thebest = bestmove
27:      end if
28:   end for
29:   if besteval <= alfa then
30:      return besteval
31:   end if
32:   bestmove = thebest
33:   return besteval
34: end if
```

possui filhos. Neste caso, a função *evaluate*(I) retorna a predição dada pela MLP para o estado I;

- **Linha 5:** Verifica se I é um nó maximizador, caso positivo, as linhas 6 a 19 são executadas;

- **Linha 6:** *besteval* representa a melhor avaliação encontrada para o nó I até o presente momento. Como I representa um nó maximizador, inicialmente o valor de *besteval* é configurado como o maior valor negativo possível. Note que *besteval* será incrementado até o máximo valor das predições associadas aos filhos de I (linhas de 7 a 13);

- **Linhas 7 a 13:** para cada um dos filhos *child*, do estado I, o algoritmo Alfa-Beta

é chamado, recursivamente, com profundidade *d - 1*. O intervalo de busca utilizado para a chamada recursiva será [*besteval; beta*] (linha 8). No nível de maximização, sempre que ocorrer atualização de um dos limites do intervalo de busca, a atualização acontecerá no limite inferior (no caso, em *besteval*). Isso acontece sempre que a predição v calculada para *child* superar o valor de *besteval* (linha 9). Caso a predição de v seja maior que *besteval* (melhor avaliação encontrada até o momento para I), o algoritmo atualiza *besteval* com o valor de v e o valor de *thebest* com o valor do melhor movimento (*bestmove*) encontrado até o momento para I (linha 10 e 11);

- **Linhas 14 a 16:** se acontecer da predição armazenada em *besteval* ultrapassar o limite superior do intervalo de busca ($besteval \geq beta$), o algoritmo retornará, imediatamente (poda beta), o valor de *besteval* como predição associada ao estado I. Tal fato expressa a ideia de que a predição associada ao estado I é, no mínimo, *besteval*;

- **Linhas 17 e 19:** após a avaliação do todos os filhos *child* do estado I, *thebest* conterá o melhor movimento a ser executado a partir do estado I; *besteval* conterá a maior predição entre todos os filhos do estado I (pois I é maximizador), e será retornado como valor da predição associada ao estado I;

- **Linhas 20 a 34:** será executado o mesmo processo descrito entre as linhas 5 e 19, porém considerando que o estado I é um nó minimizador. Neste caso, a predição v associada ao nó minimizador será igual a *menor* predição dos seus filhos e o intervalo de busca utilizado será [*alfa; besteval*] (linha 23). No nível de minimização, sempre que ocorrer atualização de um dos limites do intervalo de busca, ela será um decremento no limite superior. Isso acontece sempre que a predição v calculada para *child* for inferior ao valor de *besteval* (linha 24). Caso a predição de v seja menor que *besteval* (melhor avaliação encontrada até o momento para I), o algoritmo atualiza *besteval* com o valor de v e o valor de *thebest* com o melhor movimento encontrado para I até o momento (linha 25 e 26). Se acontecer da predição armazenada em *besteval* ser menor que o limite inferior do intervalo de busca ($besteval \leq alfa$, linha 29), o algoritmo retornará, imediatamente (poda alfa), o valor de *besteval* como predição associada ao estado. Tal fato expressa a ideia de que a predição associada ao estado é, no máximo, *besteval*. Após a avaliação do todos os filhos *child* do estado I (linhas 32 e 33), *thebest* conterá o melhor movimento a ser executado a partir do estado I; *besteval* conterá a menor predição de todos os filhos do estado I (pois I é minimizador), e será retornado como valor da predição associada ao estado I;

Observe que uma **poda alfa** ocorre tão logo uma predição v, calculada para um dos filhos de um nó minimizador I seja menor que o parâmetro alfa (linha 29). Já uma **poda**

beta ocorre tão logo uma predição v calculada para um dos filhos de um nó maximizador I seja maior que o parâmetro beta (linha 14).

3.2.3.2 Integração do algoritmo Alfa-Beta com Tabela de Transposição - Alfa-Beta + TT

O algoritmo *fail-soft* Alfa-Beta, apresentado na Seção 3.2.3.1, não mantém um histórico dos estados da árvore do jogo visitados anteriormente. Assim, se um estado de tabuleiro I for apresentado 2 vezes (ou mais) para o algoritmo Alfa-Beta, a mesma rotina será executada 2 vezes (ou mais) a fim de encontrar a predição associada a I. A reapresentação de alguns estados de tabuleiros são comuns no jogo de Damas devido ao fenômeno de transposição e também, devido a combinação do algoritmo de busca com a técnica de aprofundamento iterativo (detalhado na Seção 3.2.3.3). Durante um mesmo jogo, pode-se chegar ao mesmo estado de tabuleiro várias vezes e, quando isso ocorre, diz-se que houve uma transposição, daí a origem do nome *Tabela de Transposição*.

Para evitar o trabalho de reexecutar o algoritmo de busca para o mesmo estado de tabuleiro tantas vezes quanto ele aparecer num mesmo jogo, é recomendado o uso de TT, a qual funciona como um repositório de predições passadas associadas aos estados de tabuleiro do jogo que já foram submetidos ao algoritmo de busca. No IIGA, a TT utilizada para armazenar informações relevantes sobre os estados de tabuleiro que já foram explorados pelo algoritmo Alfa-Beta, é a tabela *hash*. Para isso, os estados de tabuleiro do jogo são representados na forma de *chaves hash* seguindo a técnica de Zobrist [81] e [82]. Tal técnica usa uma sequência de bits aleatórios distintos de comprimento fixo, chamado *chave Zobrist*, para representar as 32 posições do estado de tabuleiro do jogo de Damas. Como cada posição do tabuleiro pode representar 5 estados diferentes: peça simples preta ou vermelha, dama preta ou vermelha e vazio, então a *chave Zobrist* para Damas requer 32 x 4 = 128 entradas (observe que as posições vazias não precisam ser representadas). A Figura 12 mostra as 128 *chaves Zobrist* composta de uma sequência fixa de 64 bits.

Com o objetivo de garantir a qualidade dos números aleatórios gerados para as *chaves Zobrist* apresentadas na Figura 12 (primeira coluna), tais inteiros foram gerados a partir do gerador de bits aleatórios *Quantum Random Bit Generator Service* [83], utilizando a técnica descrita em [84], que garante a aleatoriedade da sequência gerada baseando-se na aleatoriedade intrínseca de processos físicos em que fótons são detectados ao acaso [29]. Utilizando a *chave Zobrist*, o processo de criar uma *chave hash*, para um determinado tabuleiro do jogo de Damas, é dado da seguinte forma [85]:

1. Considere um estado I do tabuleiro do jogo de Damas como sendo o mostrado na Figura 13;

2. Para conseguir o número aleatório associado à peça preta simples localizada na posição 2 do tabuleiro I, basta fazer uma consulta ao vetor da Figura 12 e encontrar

RANDOM INT64	PIECE	SQUARE
14787540466645868636	black man	1
2120251484556677534	white man	
584882445155849028	black king	
3760951787791404667	white king	
17903615704209920410	black man	2
5781218707178284009	white man	
7894141919871615785	black king	
3578131985066232389	white king	
1817657397089932766	black man	3
9537396155164801519	white man	
5808583100557493539	black king	
3651659200175719294	white king	
11250323712845617096	black man	4
15592542546949822810	white man	
16204138130260099375	black king	
9585321403807695269	white king	
15915542026527195059	black man	5
16248679709773236148	white man	
6685379756495787903	black king	
6977407078633077238	white king	
1729081295984380347	black man	6
6892212846999406827	white man	
632708781781195948	black king	
8082145037705841596	white king	
11740811010298599996	black man	7
348921443543585631	white man	
14579749940077582302	black king	
6486449913624012919	white king	
34664923411378331 91	black man	8
471079928059731524	white man	
12658037930106435315	black king	
11963310641682407293	white king	
...		...
...		
...		
...		

RANDOM INT64	PIECE	SQUARE
...		...
...		
...		
...		
8978665553187022367	black man	25
6792129980026176469	white man	
11106003084864057887	black king	
5684749757081299935	white king	
3967728617316940461	black man	26
16232032669744814011	white man	
13546780321862426801	black king	
3009792841844867034	white king	
13422590923753360614	black man	27
10221763887329211198	white man	
5616157223557226974	black king	
2865046354894257591	white king	
14642594631129895935	black man	28
8381146724961928037	white man	
3023307655632321181	black king	
8375086150794650026	white king	
11810041679881260088	black man	29
1213308520865758682	white man	
9734715559513728574	black king	
12184937488032720561	white king	
4993510297519374450	black man	30
12124137870041646186	white man	
2664161134633443445	black king	
327774891080306970	white king	
14888968537176605210	black man	31
6271745259985944523	white man	
14507257672045050736	black king	
8740695389947450601	white king	
9487810991141940225	black man	32
14639527447367762922	white man	
8795549574004575914	black king	
18030604617695974466	white king	

Figura 12 – Vetor de 128 elementos inteiros aleatórios com as chaves *Zobrist* utilizadas pelo IIGA para a montagem da Tabela de Transposição.

a *chave Zobrist* = 17903615704209920410 (valor da coluna *random int64* referente a primeira linha do grupo de *chaves* do *square* 2);

3. Para conseguir o número aleatório associado ao rei preto, localizado na casa 31 do tabuleiro I, basta fazer uma consulta ao vetor da Figura 12 e encontrar a *chave Zobrist* = 14507257672045050736 (valor da coluna *random int64* referente a terceira linha do grupo de *chaves* do *square* 31);

4. Para conseguir o número aleatório associado ao rei branco, localizado na casa 3 do tabuleiro I, basta fazer uma consulta ao vetor da Figura 12 e encontrar a *chave Zobrist* = 3651659200175719294 (valor da coluna *random int64* referente a quarta linha do grupo de *chaves* do *square* 3).

5. Assim, para conseguir a *chave hash* C associada ao estado do tabuleiro I basta aplicar o operador XOR nas três chaves obtidas nas etapas 2, 3 e 4, isto é, $C =$

1790361570420992041 0 ⊕ 14507257672045050736 ⊕ 3651659200175719294, ou seja, $C = 17159320952789611153$.

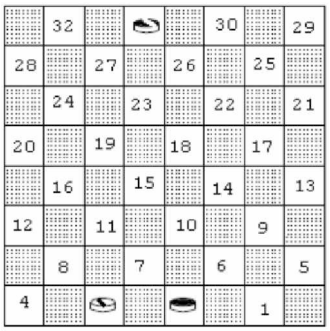

Figura 13 – Exemplo de um estado do tabuleiro de Damas utilizado para criação de uma chave *hash*.

A técnica de Zobrist é, provavelmente, o método mais rápido para calcular uma *chave hash* associada a um estado de tabuleiro do jogo de Damas, já que uma operação XOR é eficientemente executada por uma CPU, além de possibilitar uma atualização incremental na *chave hash* com movimentações de peças sobre um tabuleiro [86].

Tratamento do Problema de Colisão na TT

A TT utilizada pelo IIGA trata dois problemas de colisão identificados por Zobrist em [82]: *erro tipo 1* e *erro tipo 2*. O *erro tipo 1*, também conhecido por *clash*, ocorre quando dois estados distintos de tabuleiro de Damas são mapeados com a mesma *chave hash*. Se tal erro não for tratado adequadamente, pode acontecer de predições incorretas serem retornadas pela rotina de busca Alfa-Beta ao consultar a TT. Para reduzir a probabilidade de ocorrência de *erro tipo 1*, são utilizadas duas *chaves hash*: a *hashvalue* de 64 bits e a *checksum* de 32 bits que combinadas, praticamente eliminam as ocorrências de *clashes* (mais detalhes consulte [86]).

O segundo tipo de erro tratado é a colisão de *erro tipo 2*. Ele acontece quando dois estados de tabuleiro distintos, apesar de serem mapeados com *chaves hash* diferentes, são direcionados para o mesmo endereço na TT. Isso ocorre devido a limitação de memória disponível para a TT e a grande quantidade de estados alcançáveis num jogo de Damas. Para resolver este problema, foram utilizados dois esquemas de substituição (*replacement scheme*), chamados *Deep* e *New*, propostos por Breuker em [87]. De acordo com Breuker, se uma TT tiver dois níveis, isto é, se tiver a capacidade de armazenar informações referentes a dois estados de tabuleiro no mesmo endereço, ela terá melhor performance do que uma outra TT com o dobro de capacidade de armazenamento, mas com apenas um nível de armazenamento. Mais especificamente, de acordo com a estratégia *Deep* e *New*, na primeira vez que um determinado endereço é selecionado para armazenar informação de um determinado estado I, ele será gravado no primeiro nível. O segundo nível só será

usado se ocorrer alguma colisão de *erro tipo 2* (aí é utilizado o esquema de substituição *New*) ou em caso do primeiro nível já ter informação armazenada para o estado I. Neste caso, o primeiro nível sempre mantém a informação mais precisa do estado I utilizando o esquema de substituição *Deep* (neste esquema, a predição calculada para I referente a subárvore mais profunda é preservada). Portanto, sempre que um determinado endereço de entrada na TT armazena informações referentes ao mesmo estado de tabuleiro I em ambos os níveis, isto significa que a informação armazenada no primeiro nível foi obtida a partir de uma posição mais profunda na árvore do jogo (ou seja, informação mais precisas). Por outro lado, se o mesmo endereço de entrada na TT armazena informações referentes a dois estados diferentes, isto significa que o segundo nível foi utilizado para resolver o problema de colisão de *erro tipo 2* (mais detalhes veja [86]).

Armazenamento e Recuperação de Estados de Tabuleiro da TT

As informações relacionadas a cada estado de tabuleiro do jogo I, avaliado pela rotina de busca Alfa-Beta, são armazenados na TT de acordo com a estrutura *entry* mostrada abaixo:

```
struct TranspTable{
INT64   hashvalue = v1;
FLOAT   prediction = v2;
MOVE    best_move = v3;
INT     depth = v4;
STRING  scoretype = v5;
INT     checksum = v6;
}
```

onde *v1* e *v6* representam, respectivamente, as duas *chaves hash* calculada para o estado I: *hashvalue* de 64 bits e *checksum* de 32 bits; *v2* e *v3* são, respectivamente, a predição para I e o melhor movimento para este estado, obtido pelo algoritmo de busca Alfa-Beta; *v4* indica a profundidade da busca que foi calculada a predição *v2*; e, finalmente, *v5* informa se *v2* corresponde ao valor exato da predição de I (neste caso, *v5* é igual a *Exact*) ou um limite superior para este valor (neste caso, *v5* é igual a *AtMost*) ou ainda, um limite inferior para este valor (neste caso, *v5* é igual a *AtLeast*).

Considerando a estratégia de poda do algoritmo Alfa-Beta apresentado na Seção 3.2.3.1, o valor de *v5* para o estado I é definido da seguinte forma:

- **Exact:** sempre que não ocorrer qualquer poda durante o cálculo da predição *v2*;
- **AtMost:** sempre que ocorrer uma poda *alfa* durante o cálculo da predição *v2* (ou seja, I é um nó minimizador);

- **AtLeast:** sempre que ocorrer uma poda *beta* durante o cálculo da predição *v2* (ou seja, I é um nó maximizador);.

Por exemplo, na estrutura *entry* da TT para o estado de tabuleiro raiz I_1 da Figura 4 do Capítulo 2, as duas *chaves hash v1* e *v6* são calculados conforme explicado na Seção 3.2.3.2, e os valores *v2*, *v3*, *v4* e *v5* são, respectivamente: *0,4*, *Movimento A*, *2* e *AtLeast*.

Sempre que o algoritmo de busca recebe um estado de tabuleiro I para ser explorado, ele executa, para cada um de seus filhos C, o seguinte método:

$$retrieve(C,\ d,\ nodeType,\ besteval,\ bestmove), \tag{11}$$

onde C representa o estado de tabuleiro do jogo que está sendo procurado na TT; d representa a profundidade de busca associada a C; *nodeType* indica se o estado pai de C (no caso, I) é um nó minimizador ou maximizador; *besteval* e *bestmove* são parâmetros de saída que indicarão, caso ocorra sucesso no procedimento de recuperação do estado C na TT, a predição e o melhor movimento associados ao estado C, respectivamente.

Sempre que o método *retrieve* é executado, ele primeiro verifica se a informação relacionada ao estado C (isto é, os valores *besteval* e *bestmove*) está disponível na TT (*teste de ocorrência*). Se este teste for bem-sucedido, o método verifica se esta informação satisfaz a *restrição de uso* (definida mais adiante nesta seção), as quais devem ser satisfeitas a fim de garantir que o algoritmo Alfa-Beta combinado com TT sempre produz, para qualquer estado de tabuleiro arbitrário, o mesmo valor de predição que seria retornado pelo algoritmo *Minimax*. Se ambos testes forem bem sucedidos, então o algoritmo Alfa-Beta, em vez de avaliar C, simplesmente recupera da TT os valores *v2* e *v3* correspondente a predição e o melhor movimento a ser executado em C. Em seguida, o método instancia o valor *v2* à variável de saída *besteval* e o valor *v3* à variável de saída *bestmove*, retornando-os ao algoritmo Alfa-Beta. Se, por outro lado, um dos testes realizados pelo método *retrieve* no início da sua execução (ou ambos) falharem, o algoritmo de Alfa-Beta avalia o nó corrente C, através de chamada recursiva do próprio algoritmo Alfa-Beta, com o objetivo de obter os valores das variáveis *besteval* e *bestmove*.

Sempre que o *teste de ocorrência* for bem sucedido durante a execução do método *retrieve*, as seguintes restrições (referida como *restrição de uso*) devem ser satisfeitas a fim de garantir que os valores da TT possam ser usados [21]:

- depth $\geq$ d é a primeira restrição a ser respeitada para permitir o uso dos valores *besteval* e *bestmove* de qualquer nó C disponível na TT (*depth* é o valor *v4* que indica a profundidade de C na TT). Esta restrição está vinculada ao fato de que quanto mais profundo os valores (*besteval* e *bestmove*) são calculados para um determinado estado de tabuleiro, mais preciso eles se tornam;

- Se $depth \geq d$ e C é um nó minimizador, seu valor de predição *v2* na TT pode ser usado se seu *scoretype* (ou *v5*) é *Exact*; caso contrário (isto é, seu *scoretype* ou *v5*

é *AtMost* - é importante destacar que o *scoretype* de um nó minimizador pode ser apenas *Exact* ou *AtMost*), ele só pode ser usado se $v2 \leq \alpha$, onde α é o valor alfa da janela de busca corrente;

- Se $depth \geq d$ e C é um nó maximizador, seu valor de predição $v2$ na TT pode ser usado se seu *scoretype* (ou $v5$) é *Exact*; caso contrário (isto é, seu *scoretype* ou $v5$ é *AtLeast* - é importante destacar que o *scoretype* de um nó maximizador pode ser apenas *Exact* ou *AtLeast*), ele só pode ser usado se $v2 \geq \beta$, onde β é o valor beta da janela de busca corrente.

3.2.3.3 Integração do algoritmo Alfa-Beta com Tabela de Transposição e Aprofundamento Iterativo - Alfa-Beta + TT + AI

A qualidade de um agente jogador que utiliza o algoritmo de busca Alfa-Beta está relacionada a nível de profundidade que ele consegue atingir na árvore do jogo durante a busca pelo melhor movimento. A profundidade da busca está relacionada ao quanto o jogador consegue olhar adiante (*look-ahead*) e prever as jogadas do adversário. No jogo de Damas, o *look-ahead* do agente pode ser restringido devido às limitações impostas pelos recursos computacionais ou ainda, pelo limite de tempo que o agente tem para executar um movimento. A maioria dos jogadores automáticos de Damas utilizam mecanismos para delimitar o tempo máximo permitido de busca. Como o algoritmo Alfa-Beta realiza uma busca com profundidade fixa, não existe garantia de que a busca irá se completar antes que o tempo máximo de busca se esgote. Para evitar que o tempo se esgote e o algoritmo não escolha o melhor movimento, buscas com profundidade fixa devem ser evitadas. Nesse sentido, a combinação do Alfa-Beta com a técnica de AI garante que o algoritmo retorne um movimento antes que o tempo se esgote e ainda, caso haja tempo, permite que o algoritmo busque movimentos com *look-ahead* mais profundos, até a profundidade máxima estabelecida.

O AI é um mecanismo que controla do tempo de execução durante a expansão de uma árvore de busca [88]. A ideia básica do AI é realizar uma série de buscas em profundidade, independentes, cada uma com um *look-ahead* acrescido de um nível. Inicialmente, o algoritmo Alfa-Beta pesquisa com profundidade 2, depois com profundidade 4, depois com profundidade 6 e assim, sucessivamente, até que o tempo máximo de busca se esgote. A desvantagem desta técnica é o processamento repetido de estados de níveis mais rasos da árvore de busca. Dessa forma, a combinação do AI com a TT se faz necessário, uma vez que acelera o processo de busca iterativa. Os estados de tabuleiros já avaliados em níveis mais rasos do AI são recuperados da TT para a iteração do algoritmo em níveis mais profundos.

Particularmente, o IIGA executa buscas iterativas de profundidade acrescidas de 2 níveis que são limitadas aos critérios de restrição de profundidade máxima (*max-depth*)

e intervalo de tempo máximo (*max-time*). Em outras palavras, o algoritmo Alfa-Beta é chamado com profundidade *depth* = 4, 6, ..., *max-depth*, até que o tempo de busca *max-time* se esgote em uma profundidade qualquer *depth* = d, tal que $4 \leq d \leq max - depth$.

Outro benefício da combinação do AI com o Alfa-Beta e a TT, é a ordenação parcial da árvore de busca em que se coloca no ramo mais à esquerda da árvore, o nó filho que obtiver melhor predição obtida na iteração anterior. Assumindo que uma busca mais rasa é uma boa aproximação para outra mais profunda, a melhor ação para um estado I na profundidade d será, possivelmente, a melhor ação para o estado I na profundidade $d+1$ [89] e, com a ordenação da árvore colocando o melhor movimento no primeiro ramo da árvore de busca (ramo mais a esquerda), o melhor movimento deverá ser encontrado mais rapidamente. Por exemplo, a Figura 14 mostra o resultado de duas iterações sucessivas do algoritmo Alfa-Beta: veja que a iteração com profundidade d retornou como melhor predição o valor 0,20 (referente ao nó B como melhor movimento a ser realizado a partir da raiz). Assumindo que o resultado obtido pela iteração d contém uma boa aproximação do melhor movimento a ser realizado na iteração $d+1$, a árvore de busca é ordenada de forma que o nó filho B fique mais à esquerda da mesma. No exemplo, após a execução da iteração $d+1$, o nó filho B mostrou-se, realmente, como a melhor opção de movimento com uma predição 0,30.

Com a ordenação, em cada nível da árvore de busca o movimento com mais possibilidade de ser o melhor ocupará sempre a primeira posição da árvore, sendo o primeiro a ser explorado pelo Alfa-Beta. Isso faz com que o Alfa-Beta fique mais ágil, visto que aumenta as chances de poda do algoritmo (pois os possíveis melhores movimentos são os primeiros a serem explorados). Uma vez que torna o processo de busca mais eficiente (graças ao aumento do número de podas), o AI consegue explorar níveis mais profundos da árvore na tentativa de encontrar melhores movimentos, aumentando *look-ahead* do jogador.

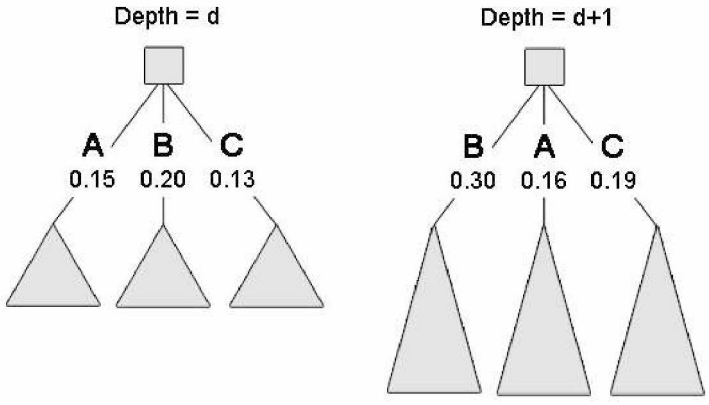

Figura 14 – Exemplo de ordenação da árvore de busca em duas iterações da busca com aprofundamento iterativo.

3.3 Agentes Especialistas na Fase de Final de Jogo - *Módulos EGAs Treinados e EGA*

O MP-Draughts é composto por vários EGAs especializados nos vários perfis de final de jogo identificados pela RN adaptativa. Cada EGA possui a mesma arquitetura do IIGA, ou seja, cada um é uma MLP que aprende a jogar por reforço utilizando os métodos TD(λ), durante o processo de treinamento por *self-play* com clonagem. O algoritmo Alfa-Beta combinado com TT e AI é utilizado pelo EGA para indicar o melhor movimento a ser executado considerando o estado corrente do jogo.

O processo de treinamento dos EGAs é análogo ao do IIGA (ver Seção 3.2), diferenciando-se apenas pelos estados de tabuleiros utilizados durante o processo de treinamento. Cada EGA é treinado para representar um determinado perfil de final de jogo, sendo que cada perfil representa um *cluster* de estados de tabuleiros de final de jogo. O treinamento de cada EGA é executado nos estados de tabuleiros de final de jogo pertencentes ao *cluster* ao qual ele foi designado à representar. O processo de obtenção dos *clusters* de treinamentos dos EGAs será explicado na Seção 3.4.

Ao final do processo de treinamento dos EGAs, o conjunto composto por todos os EGAs treinados corresponde ao módulo *EGAs Treinados* da Figura 10, apresentada no início deste capítulo. O módulo *EGA* da Figura corresponde ao EGA responsável por atuar na fase de final de jogo. Este *EGA* é alocado por uma RN adaptativa do módulo de *EGAs Treinados*.

3.4 Agrupamento da Base de Dados de Treinamento e Alocação dos EGAs - *Módulo RN Adaptativa*

O MP-Draughts possui vários EGAs, cada um treinado para lidar com um determinado *cluster* de final de jogo. Tais *clusters* foram minerados, pelo módulo *RN Adaptativa*, de uma BD contendo 689 estados de tabuleiros de final de jogo. Este mesmo módulo também é responsável por, durante os jogos, alocar dentre os EGAs treinados, àquele que possui maior capacidade de atuar na fase de final. A obtenção da BD de final de jogo é descrito na Subseção 3.4.1.

O processo geral de agrupamento da BD de final de jogo dos EGAs, apresentado na Figura 15 é o seguinte: um estado de tabuleiro presente na BD de final de jogo é convertido para a representação *NetFeatureMap* e apresentado à camada de entrada da RN adaptativa, a qual verifica a similaridade entre o estado de tabuleiro e todos os seus neurônios. O neurônio com maior similaridade torna-se candidato para representá-lo. Caso a similaridade entre o neurônio e o estado de tabuleiro seja superior ao critério de similaridade mínimo estabelecido para o agrupamento (limiar de ativação do neurônio),

o neurônio torna-se apto para agrupar o estado de tabuleiro e tem seus pesos ajustados para representá-lo. Caso a similaridade entre eles seja inferior ao limite mínimo pré-estabelecido, a RN cria um novo neurônio para agrupar o estado de tabuleiro. A representação vetorial do estado de tabuleiro é armazenada no *cluster* que representa o neurônio que o agrupou. Ao final do processo de agrupamento, haverão *n clusters*, os quais são suficientes para representar os perfis de final de jogo representados na BD.

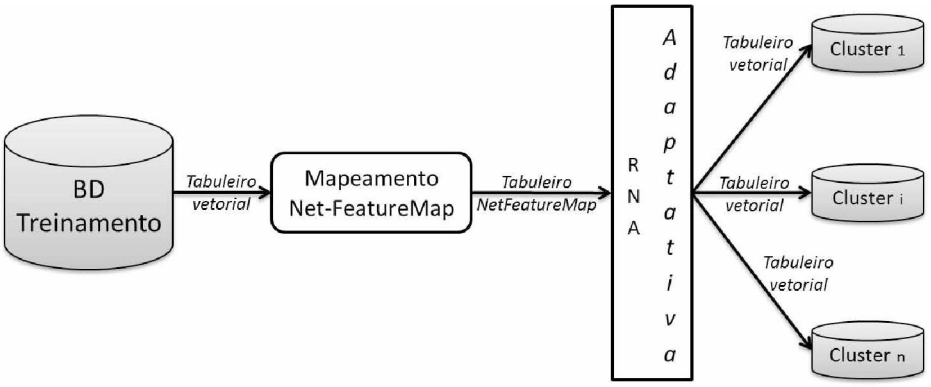

Figura 15 – Processo geral de agrupamento da BD de treinamento dos EGAs.

Como citado na introdução deste capítulo, a versão do MP-Draughts proposta neste trabalho é uma otimização da versão do agente descrito na Seção 2.2.2.1. A presente versão otimiza a anterior pela substituição da KSOM-DE (cuja arquitetura é fixa e previamente definida pelo usuário) por uma RN adaptativa, a qual define dinamicamente a quantidade mais apropriada de *clusters* para representar os perfis de final de jogo representados numa BD.

Para encontrar qual RN adaptativa é mais eficiente no processo de agrupamento da BD de final de jogo, as seguintes RN adaptativas foram investigadas nesta seção: ART 2A, SONDE e a *Adaptation of Self-Organizing Novelty Detection* (ASONDE) (proposta neste trabalho), cujas medida de similaridade utilizada é a similaridade cosseno. Considerando que a medida de similaridade utilizada pela KSOM-DE (utilizada na versão preliminar) e as RN adaptativas investigadas são diferentes, também foi feita uma investigação a cerca dessas medidas para averiguar qual delas melhor representa a similidade entre os estados de tabuleiros de final de jogo. Sendo assim, para manter a coerência dos resultados e garantir que os ganhos estão relacionados a adaptabilidade das RNs e não apenas a medida de similaridade usada por um outra abordagem (adaptativa ou não), a versão da Rede de Kohonen baseada em similaridade cosseno (KSOM-Cos) também foi investigada.

As próximas subseções estão organizadas de modo a conduzir o entendimento do leitor quanto a evolução desta investigação.

3.4.1 Obtenção da Base de Dados de Treinamento dos Agentes de Final de Jogos

A escolha de usar tabuleiros com 14 peças para representar a fase de final de jogo foi feita após a observação de vários estágios do jogo de Damas. No estágio de jogo com 14 peças sobre o tabuleiro, na maioria dos casos, o jogo apresenta-se equilibrado quanto a quantidade de peças de cada jogador. Além disso, analisando os próximos movimentos do jogo após esse estágio, pôde-se observar o aumento do interesse dos jogadores em avançar para o território do oponente em busca de formar Damas e finalizar o jogo. Foram analisados tabuleiros com 12 e 10 peças, porém nesses, a vantagem de um jogador em relação ao seu oponente já é mais visível devido à diferença na quantidade de peças de cada jogador e posição dessas peças no tabuleiro. Tal situação não é favorável para o aprendizado dos EGAs, uma vez que sua eficiência poderia estar vinculada a situação de vantagem (ou desvantagem) do jogo. Além disso, nesses tabuleiros o período de aprendizagem de um agente é reduzido, visto que o jogo tende a finalizar em poucas jogadas adiante.

Em se tratando de tabuleiros com 14 peças, existem mais de 49 quatrilhões de configurações possíveis de tabuleiros [20], conforme apresentado na Figura 16, porém nem todas essas configurações são prováveis de ocorrerem num jogo (por exemplo 13 peças damas de um jogador contra 1 peça simples do oponente). Desse modo, para garantir que o treinamento dos EGAs ocorra em estados de tabuleiros factíveis de ocorrerem num jogo, optou-se por filtrar os estados de tabuleiros que já ocorreram em jogos envolvendo bons jogadores de Damas. Dessa maneira, a BD de final de jogos utilizada no processo de treinamento dos EGAs foi obtida a partir de jogos de campeonatos mundiais envolvendo o melhor jogador de Damas de todos os tempos, Mario Tinsley [90], contra outros oponentes humanos e, também, contra o campeão mundial homem-máquina Chinook. Tais jogos estão numa BD disponível na plataforma *CheckerBoard* [91] no formato PDN.

A BD com os jogos do Tinsley contém 724 jogos, e o processo de filtragem obteve 689 estados de tabuleiros de final. Note que, mesmo procurando o equilíbrio para a representação do que é final de jogo (no caso, tabuleiros com 14 peças), 35 jogos foram finalizados antes disso. Isso acontece porque, dependendo da situação do jogo, o mesmo é finalizado como empate, vitória ou derrota antes mesmo de chegar ao fim do jogo.

O processo de filtragem dos estados de tabuleiros de final de jogo pode ser resumido da seguinte forma: um jogo completo é recuperado da BD com os jogos do Tinsley. Em seguida, os movimentos do jogo são reexecutados até que o tabuleiro atinja a quantidade de 14 peças. Nesse momento, o jogo é paralisado e o respectivo estado de tabuleiro é armazenado na BD de final de jogo. O estado de tabuleiro é armazenado na representação vetorial da plataforma CheckerBoard, que é diferente da utilizada pelo MP-Draughts. Ao fim do processo de filtragem, os estados de tabuleiros da BD de final de jogo são convertidos para a representação vetorial utilizada pelo MP-Draughts (veja exemplo na Figura 5).

Peças	Número de Posições
1	120
2	6,972
3	261,224
4	7,092,774
5	148,688,232
6	2,503,611,964
7	34,779,531,480
8	406,309,208,481
9	4,048,627,642,976
10	34,778,882,769,216
Total 1–10	39,271,258,813,439
11	259,669,578,902,016
12	1,695,618,078,654,976
13	9,726,900,031,328,256
14	49,134,911,067,979,776
15	218,511,510,918,189,056
16	852,888,183,557,922,816
17	2,905,162,728,973,680,640
18	8,568,043,414,939,516,928
19	21,661,954,506,100,113,408
20	46,352,957,062,510,379,008
21	82,459,728,874,435,248,128
22	118,435,747,136,817,856,512
23	129,406,908,049,181,900,800
24	90,072,726,844,888,186,880
Total 1–24	500,995,484,682,338,672,639

Figura 16 – Espaço de estados para o jogo de Damas: quantidade de estados possíveis de acordo com o número de peças sobre o tabuleiro.

Finalizado o processo de filtragem, a BD de final de jogo foi dividida em BD de treinamento e BD de teste utilizando o método de validação de cruzada (*k-foldcross-validation*) com k=5 [92], [93]. Ou seja, a BD foi dividida, aleatoriamente, em 5 conjuntos (1-CV, ..., 5-CV). Cada conjunto contém uma BD de treinamento e uma BD de teste, sendo que a BD de treinamento contém, aproximadamente, 80% dos estados de tabuleiros, os quais são diferentes dos estados da BD de teste.

3.4.2 Agrupamento da Base de Dados de Treinamento

As RNs utilizadas para o agrupamento da BD de treinamento foram: ART 2A, SONDE, ASONDE (adaptação da SONDE, proposta neste trabalho) e KSOM-Cos, sendo que as três primeiras foram investigadas no intuito de encontrar a melhor RN adaptativa para detectar os perfis existentes na fase de final de jogo e, a última, para manter a coerência dos resultados no que diz respeito às medidas de similaridade DE e similaridade Cosseno. Todas as RN investigadas são baseadas em similaridade cosseno e, pelo fato de a primeira versão do MP-Draughts utilizar uma KSOM-DE para minerar a BD, achou-se conveniente investigar sua versão baseada em similaridade cosseno. Tal investigação visa delinear o ganho obtido pela adaptabilidade das RNs e o ganho obtido pela mudança da medida de similaridade.

Vale ressaltar aqui que outras variações de RNs adaptativas foram investigadas, porém os resultados obtidos por elas foram muito inferiores aos obtidos pela versão preliminar do MP-Draughts, de modo que estas foram desconsideradas. Essas RN são: ART 2A-E, ART 2A-C e ART C-2A [64]. Inclusive uma variação híbrida da ART 2A que combina suas características adaptativas com o conceito de vizinhança da KSOM foi implementada

e os resultados também não foram satisfatórios.

Considerando que a BD foi minerada por 4 RN distintas, para descobrir qual delas seria a melhor para o problema, a seguinte estratégia foi adotada: a BD foi minerada pelas 4 RNs, gerando 4 conjuntos de *clusters* distintos. Cada *cluster* de cada conjunto de *clusters* foi associado a um EGA para treinamento, de modo que ao final do treinamento haveriam 4 conjuntos de EGAs treinados (cada conjunto de EGA se tornou especialista num conjunto de *clusters* obtidos por uma única RN). Cada conjunto de EGAs treinados, em conjunto com a RN que gerou os *clusters* de treinamento desses EGAs, representam uma versão do MP-Draughts para a fase de final de jogo (módulos *RNs Adaptativa*, *EGAs treinados* e *EGA* da Figura 19 responsáveis pela fase de final de jogo). A fim de verificar qual dessas versões era a melhor, foram realizados jogos competitivos entre todas elas, inclusive contra a versão preliminar (baseada em KSOM-DE), de modo a medir o desempenho de cada uma. A versão de final do jogo que obteve os melhores resultados em jogos, em conjunto com o IIGA, representa então a arquitetura geral do MP-Draughts proposta neste capítulo.

Os estados de tabuleiros da BD de final de jogo foram convertidos para a representação *NetFeatureMap* antes de serem apresentados às RNs para o agrupamento. A conversão é justificável pelo fato de as características utilizadas por esta representação agregarem mais conhecimentos do jogo do que a mera percepção da localização e do tipo das peças no tabuleiro inerente à representação vetorial. Foram utilizadas as 15 características apresentadas na Tabela 2 para representar um estado de tabuleiro, o qual foi representado utilizando 48 *bits*. Sendo assim, as RNs devem possuir 48 neurônios na camada de entrada para receber esta representação. Maiores informações sobre o número de *bits*, consulte Seção 2.3.5.

Abaixo seguem os passos realizados para o agrupamento da BD em cada uma das RNs acima citadas, assim como a adaptação feita na SONDE.

Agrupamento pela ART 2A

Para agrupar os estados de tabuleiros da BD utilizando a ART 2A, cada estado de tabuleiro I_i é convertido para a representação *NetFeatureMap* e apresentado na camada de entrada da ART 2A. Tal RN localiza o neurônio Y_j com maior capacidade (maior similaridade) para representar I_i. Este neurônio, se torna candidato à representar I_i. Caso a similaridade entre I_i e Y_j seja superior a vigilância estabelecida por ρ, o neurônio candidato é considerado vencedor, Y_J, e se adapta para representar I_i. Caso contrário (a similaridade é inferior a ρ), a ART 2A cria um novo neurônio,Y_J, para representar I_i. A representação vetorial de I_i é armazenado no *cluster* J que representa o neurônio Y_J.

A ART 2A foi treinada por 1000 épocas, com taxa de aprendizagem β iniciando em 1 e sendo decrementada em 5% a cada 10 épocas. O melhor valor para o parâmetro de vigilância ρ foi 0,88. Vários valores de ρ foram testados, e 0,88 foi o que obteve

os melhores agrupamentos. Ou seja, os dados foram bem distribuídos entre os *clusters* obtidos do agrupamento, de modo que nenhum *cluster* centralizou grande parte dos dados e, do mesmo jeito, que nenhum ficou com poucos. Foram criados 17 *clusters* para agrupar os estados de tabuleiro da BD de final de jogo. Em outras palavras, ART 2A foi capaz de detectar 17 perfis de final de jogo, sendo que os estados de tabuleiros pertencentes a um mesmo perfil são, pelo menos, 88% similares entre si. A versão do MP-Draughts cujos EGAs foram treinados nos *clusters* obtidos pela ART 2A, possui 17 agentes especialistas na fase de final de jogo (EGAs).

Agrupamento pela KSOM-Cos

Para agrupar os estados de tabuleiros da BD utilizando a KSOM-Cos, cada estado de tabuleiro I_i é convertido para a representação *NetFeatureMap* e apresentado na camada de entrada da KSOM-Cos. Esta RN calcula a similaridade entre I_i e cada neurônio Y_j da camada de saída. O neurônio que possuir maior similaridade com I_i é eleito o neurônio vencedor Y_J. Os pesos sinápticos de Y_J são ajustados para representar I_i, assim como os pesos de seus vizinhos (num grau menor). A representação vetorial de I_i é armazenado no *cluster* J que representa Y_J.

A arquitetura da KSOM-Cos foi fixada em 17 neurônios na camada de saída (mesma quantidade de neurônios criados pela ART 2A). A KSOM-Cos foi treinada por 1000 épocas, com taxa de aprendizagem β iniciando em 1 e sendo decrementada em 5% a cada 10 épocas. O raio de vizinhança considerado foi 1. A distribuição dos dados entre os *clusters* foi equilibrada, de modo que nenhum deles ficou com muitos ou poucos registros. Assim como na ART 2A, a versão do MP-Draughts cujos EGAs foram treinados nos *clusters* obtidos pela KSOM-Cos, possui 17 EGAs.

Agrupamento pela SONDE

O processo de agrupamento de um estado de tabuleiro I_i pela SONDE é similar ao processo executado pela ART 2A, ou seja, a SONDE procura pelo neurônio com maior similaridade para agrupar I_i, caso a similaridade a_j entre eles seja superior ao limiar de ativação do neurônio, α_j, o neurônio é eleito como vencedor Y_J (nesta RN chamado de BMU) e tem seus pesos ajustados para representar I_i. A diferença em relação a ART 2A, está no reajuste dos pesos do neurônio vencedor BMU. Na SONDE, o centróide $\vec{w}_{BMU}$ (tendência média), o raio médio rad_{BMU} (dispersão média) e o limiar de ativação do neurônio α_{BMU} (grau mínimo de similaridade) são ajustados para representar I_i. Caso a similaridade a_j seja inferior a α_j, a RN cria um novo neurônio para representar I_i. Neste caso, o centróide $\vec{w}_{new}$ do novo neurônio é igual ao padrão I_i responsável por sua criação, e o limiar de ativação α_{new} é igual a α_0 (valor pré-definido pelo usuário) e o raio médio inicial rad_{new} é igual a $-ln(\alpha_0)$. Da mesma maneira, a representação vetorial do estado

de tabuleiro I_i é armazenado no *cluster* J que representa BMU. Para relembrar como é feito o reajuste do pesos do neurônio vencedor veja Seção 2.3.6.3.

Por se tratar de uma RN projetada para processar dados contínuos, a SONDE não tem etapa de treinamento. Os parâmetros Ω e γ foram definidos em 0,1 e o α_0 em 0,98.

O agrupamento obtido pela SONDE foi muito diferente do agrupamento obtido pelas ART 2A e KSOM-Cos. Enquanto a ART 2A criou 17 clusters para agrupar os estados de tabuleiro com critério de similaridade mínima igual a 0,88 ($\rho = 0,88$), a SONDE criou apenas 2 *clusters* com valor do critério de similaridade mínima $\alpha_0 = 0,98$ (valor muito alto). Além disso, a distribuição dos dados nos *clusters* não foi boa, sendo que aproximadamente 85% dos estados de tabuleiro foram agrupados no primeiro *cluster*. Simulando o mesmo cenário na KSOM-Cos (arquitetura da KSOM-Cos com apenas 2 neurônios) a distribuição dos estados de tabuleiros nos *clusters* ficou em torno de 47 e 53%. Sabendo da diversidade de situações possíveis de estados de tabuleiros de final de jogo representados na BD, acredita-se que 2 *clusters* não sejam suficientes para representar todo o conhecimento da BD e nem mesmo, que os estados de tabuleiros sejam tão parecidos (similaridade de 98%).

Apesar disso, e para comprovar que a RN não obteve sucesso no agrupamento da BD de final de jogos, uma versão do MP-Draughts baseada nessa RN foi gerada. Os resultados apresentados na Seção 3.5, comprovam a baixa performance da SONDE em agrupar o conhecimento da BD de maneira a contribuir para a performance do MP-Draughts. O baixo desempenho da SONDE para tratar a BD de final de jogo está relacionado ao fato desta RN ter sido originalmente projetada para agrupar dados de fluxo contínuo [65], enquanto que a BD de final de jogo é finita e estável. Dessa maneira, este trabalho propôs uma adaptação na estrutura da SONDE de modo a torná-la capaz de agrupar BD finitas e estáveis.

Adaptação da SONDE - ASONDE

A característica finita e estável da BD de final de jogo (i.e, que não se modifica ao longo do tempo) permite que os estados de tabuleiros sejam apresentados às RNs por vários ciclos de aprendizagem, de modo que os parâmetros de aprendizagem e os pesos sinápticos que representam o conhecido da RN possam ser ajustados ao longo desses ciclos. Essa característica da BD contribui para o refinamento e aprimoramento das habilidades de agrupamento das RNs.

Na SONDE os dados são apresentados uma única vez e os parâmetros de aprendizagem Ω e γ são constantes. Para adaptar a SONDE às características da BD de final de jogo, foi implementado ciclos de aprendizagem com reajuste dos parâmetros Ω e γ. O Algoritmo 3 apresenta as modificações inseridas no algoritmo da SONDE. Observe que foram incluídas as seguintes etapas: na linha 3 a quantidade de N épocas, correspondente ao ciclo de aprendizagem da RN, deve ser pré-definida pelo usuário; na linha 4, os parâmetros γ

e Ω têm seus valores *iniciais* definidos pelo usuário; na linha 6 a taxa *lr*, referente ao reajuste de Ω e γ, também deve ser pré-definida pelo usuário; na linha 9 se inicia o ciclo de aprendizagem da ASONDE (laço de repetição de N épocas), o qual finaliza na linha 39; nas linhas 37 e 38 os parâmetros Ω e γ são reajustados por *lr*. Finalizado os N ciclos de treinamento, os dados são reapresentados a ASONDE, agora sem reajuste de pesos e parâmetros, a qual agrupa-os nos *clusters* definidos durante os ciclos de treinamento.

Algoritmo 3 ASONDE

1: {A dimensão do padrão de entrada $\vec{I}_t$ é n}
2: {K representa o conjunto dos centróides}
3: {A quantidade de N épocas é pré-definida pelo usuário}
4: {Os parâmetros γ e Ω são inicialmente definidos pelo usuário}
5: {O parâmetro α_0 é constante e pré-definido pelo usuário}
6: {A taxa de reajuste lr é pré-definida pelo usuário}
7: {Um neurônio y_i é composto por: α_i, rad_i e $\vec{w}_i$}
8: {A ativação do neurônio y_i é dada por a_i}
9: **while** quantidade de épocas $< N$ **do**
10: **for all** $\vec{I}_t$,do instante de tempo $t \in \mathbb{N}$ **do**
11: {Fase de normalização do vetor de entrada. Executada quando n > 1}
12: $\vec{I}_t = \frac{\vec{I}_t}{\|\vec{I}_t\|}$
13: {Cálculo da ativação de cada neurônio e busca pelo melhor neurônio BMU}
14: **for all** $\vec{w}_i \in K$ **do**
15: {Cálculo da distância entre $\vec{w}_i$ e $\vec{I}_t$}
16: $dist_i = \|\vec{w}_i - \vec{I}_t\|$
17: {Cálculo da ativação do neurônio corrente y_i}
18: $a_i = exp(-dist_i)$
19: **if** $a_i > a_{BMU}$ **and** $a_i > \alpha_i$ **then**
20: {Encontra o neurônio com maior ativação}
21: $y_{BMU} = y_i$
22: **end if**
23: **end for**
24: **if** BMU foi encontrado **then**
25: {Adaptação do melhor neurônio y_{BMU}}
26: $\vec{w}_{BMU_t} = (1-\gamma) * \vec{w}_{BMU_{t-1}} + \gamma * \vec{I}_t$
27: $rad_{BMU_t} = (1-\Omega) * rad_{BMU_{t-1}} + \Omega * \|\vec{I}_t - \vec{w}_{BMU_{t-1}}\|$
28: $p = \|\frac{rad_{BMU_t} - rad_{BMU_{t-1}}}{max(rad_{BMU_t}, rad_{BMU_{t-1}})}\|$
29: $\alpha_{BMU_t} = min((1+p) * \alpha_{BMU_{t-1}}, exp^{(} - rad_{BMU_t} * (1+p)))$
30: **else**
31: {Cria novo neurônio $y_{new} \in K$}
32: $\vec{w}_{new} = \vec{I}_t$
33: $\alpha_{new} = \alpha_0$
34: $rad_{new} = -ln(\alpha_0)$
35: **end if**
36: **end for**
37: $\gamma = \gamma * lr$
38: $\Omega = \Omega * lr$
39: **end while**

A ASONDE também foi treinada por 1000 épocas, com taxas de aprendizagem Ω e γ igual a 0,1 e sendo decrementadas em 5% a cada 10 épocas ($lr = 0,95$), e $\alpha_0 = 0,88$. Tais valores foram definidos de modo a manter conformidade com os definidos para a ART 2A e para a KSOM-Cos. Valores de Ω e γ variando entre 0,9 e 0,1 foram testados. Porém, lembre-se que os valores de Ω e γ definem a influência de padrões do passado na situação corrente da RN e, quando maior os valores desses parâmetros, maior é o grau de esquecimento da RN. Considerando que os ciclos de treinamento inseridos nessa RN tiveram a intenção de refinar, e não de "sobrescrever" seu conhecimento, não faria sentido que os valores de Ω e γ fossem altos.

Para os parâmetros acima citados, a ASONDE criou 14 *clusters* para agrupar a BD. A distribuição dos estados de tabuleiros nos *clusters* foi melhor do que na SONDE, se assemelhando a uniformidade de distribuição obtida pela ART 2A e pela KSOM-Cos. A versão do MP-Draughts cujos EGAs foram treinados nos *clusters* obtidos pela ASONDE, possui 14 agentes especialistas na fase de final de jogo.

3.5 Resultados Experimentais

Os resultados apresentados nesta seção avaliam, sequencialmente: a coerência do processo de agrupamento executado pelas RNs, a adequação das medidas de similaridade, a contribuição das RNs adaptativas para a representar a fase de final de jogo, além do desempenho geral do MP-Draughts contra outros oponentes. Os resultados foram medidos considerando o desempenho do MP-Draughts nas fases de final de jogo.

Para avaliação desses pontos, foram criados 5 cenários de testes: o primeiro cenário compara os *clusters* obtidos por cada RN; o segundo avalia qual medida de similaridade, dentre as usadas pelas RNs aqui investigadas, é a mais adequada para agrupar os estados de tabuleiros de final de jogo; no terceiro, baseado no resultado obtido do segundo, define-se qual é a melhor versão do MP-Draughts. Por consequência, este cenário define a RN mais adequada para definir os perfis de final de jogo presentes num jogo de Damas. O quarto cenário, avalia a performance da melhor versão do MP-Draughts contra outros jogadores de Damas não supervisionados. Por fim, o quinto cenário compara a média de coincidência entre os movimentos executados pelos jogadores não supervisionados aqui envolvidos, em relação aos movimentos que seriam executados pelo jogador fortemente supervisionado, Cake [36], na mesma situação. As avaliações dos cenários 2, 3, 4 e 5 são feitas por meio de jogos competitivos entre os jogadores envolvidos nesses cenários.

Considerando que todas as versões do MP-Draughts apresentariam o mesmo comportamento na fase de início e meio de jogo, visto que possuem o mesmo IIGA, os testes foram executados exclusivamente em situações de final de jogo. Os testes de validação, cenários 2 e 3, foram executados usando as 5 BD de testes obtidas do particionamento 5-CV na Seção 3.4.1. Já os testes de avaliação, cenários 4 e 5, foram executados em

estados de tabuleiros obtidos da BD OCA 2.0, também disponível em [91]. Durante os testes, cada jogador joga alternadamente com peças pretas e vermelhas.

Os jogos terminados em empate foram desconsiderados para a análise dos resultados, uma vez que nestes jogos nenhum jogador se mostra mais ou menos eficiente.

3.5.1 Cenário 1: comparação dos *clusters*

Este cenário avalia o quão compatíveis são os *clusters* obtidos pelas técnicas de agrupamento aqui investigadas. Esta comparação mostra o quão similares (ou dispares) são os resultados produzidos pelas diferentes RNs de agrupamento. Este resultado, num primeiro momento, mostra apenas o quão semelhante uma técnica é em relação as outras com que foram comparadas, porém, num segundo momento quando forem analisados os ganhos de desempenho obtidos pelo MP-Draughts, treinado nos *clusters* obtidos por essas RNs, será possível avaliar o quanto essa similaridade (ou dissimilaridade) contribuiu para o aprendizado de cada EGA.

Os *clusters* obtidos pelas RNs foram comparados utilizando o método *Rand Index* Ajustado (do inglês *Adjusted Rand Index* (ARI)) [94]. Os resultados de comparação obtidos pelo ARI podem assumir valores entre [-1, 1], sendo que 1 indica a máxima concordância entre os *clusters* obtidos pelas técnicas comparadas, e valores próximos de 0 ou negativos indicam que as concordâncias foram obtidas ao acaso.

A Tabela 4 ilustra os resultados obtidos das comparações. Considerando que a parte inferior da tabela é o espelho da parte superior, a mesma foi desconsiderada para avaliação. O ARI obtido da comparação entre os *clusters* da *ART 2A* e as demais técnicas foram em média [1]: 0,40 com desvio padrão $\sigma = 0,04$ quando comparado com a *KSOM-Cos*; 0,01 com desvio $\sigma = 0,003$ quando comparado com a *SONDE* e 0,03 com desvio $\sigma = 0,01$ quando comparado com a *ASONDE*. Isso quer dizer que os *clusters* obtidos pela KSOM-Cos são mais similares aos obtidos pela ART 2A dos que os obtidos pelas variações SONDE quando comparados com a ART 2A. Comparando *KSOM-Cos* com a *SONDE* e a *SONDE*, o ARI foi menor que 0,01 com desvio $\sigma = 0,003$ nos dois casos, ou seja, a KSOM-Cos produziu *clusters* muito diferentes dos produzidos pelas SONDE e ASONDE. Quando comparado os *clusters* obtidos pela *SONDE* e pela *ASONDE* o valor médio do ARI sobe para 0,6 com desvio $\sigma = 0,02$, sendo os *clusters* mais concordantes entre todos os comparados.

Por enquanto, os resultados permitem concluir apenas que a ART 2A e a KSOM-Cos produziram *clusters* mais similares entre si, enquanto que a SONDE e a ASONDE também o fizeram. O quanto essa disparidade entre os *clusters* foi interessante para a aprendizagem o MP-Draughts será confirmada nos próximos cenários. O ganho de desempenho do MP-Draughts treinado nos *clusters* obtidos por essas RNs servirá como indicador da qualidade dos agrupamentos obtido por essas RNs.

[1] média obtida nos 5-CV

ARI				
	ART 2A	KSOM-Cos	SONDE	ASONDE
ART 2A	-	0,4 (σ=0,04)	0,01 (σ=0,003)	0,03 (σ=0,01)
KSOM-Cos	-	-	0,007 (σ=0,003)	0,006 (σ=0,003)
SONDE	-	-	-	0,6 (σ=0,02)
ASONDE	-	-	-	-

Tabela 4 – Comparação da similaridade entre os *clusters* obtidos pelas RNs utilizando o método ARI.

3.5.2 Cenário 2: avaliação da medida de similaridade

Este cenário verifica qual medida de similaridade, entre as investigadas aqui, é a mais apropriada para agrupar os estados de tabuleiros da BD de final de jogo. As medidas avaliadas são distância Euclidiana e similaridade Cosseno.

A fim de verificar qual medida é a mais apropriada, foram realizados jogos competitivos entre a versão do MP-Draughts baseada em KSOM-Cos e a versão preliminar no mesmo, a qual é baseada em KSOM-DE. A única diferença entre as duas versões é a medida de similaridade utilizada para obter os *clusters* de treinamento dos EGAs. Ambas as versões foram treinadas pelo mesmo período de tempo, possuem a mesma quantidade de EGAs (no caso, 17) e os *clusters* foram obtidos da mesma BD de final de jogo.

A Tabela 5 mostra as porcentagens dos resultados obtidos de 1378 jogos, os quais foram executados em 689 estados de tabuleiros diferentes. Em cada tabuleiro foram executados 2 jogos, nos quais os jogadores jogaram alternadamente com peças pretas e vermelhas para manter as vantagens relacionadas as características *NetFeatureMap* para ambos os jogadores. Como pode ser observado, a versão do MP-Draughts baseada em KSOM-Cos obteve em média 16% de vitórias, com desvio padrão de 8,6, enquanto a versão preliminar baseada em KSOM-DE obteve 11%, com desvio padrão de 4,3. A superioridade (vitórias) da versão baseada em KSOM-Cos foi em média 5%. Tais resultados permitem concluir que a similaridade cosseno agrupou os estados de tabuleiros de maneira a contribuir mais para o aprendizado dos EGAs do que a distância Euclidiana. Tal resultado já era esperado, devido ás características dos estados de tabuleiro da BD. Tais estados possuem 15 atributos (um para representar cada característica do mapeamento *NetFeatureMap*), sendo que muitos deles têm valor 0 (dados esparsos).

Para confirmar se a média de vitórias da versão KSOM-Cos é superior a da versão KSOM-DE, foi aplicado o *teste t* (do inglês *t-test*), disponível em [95]. Considerando os valores das médias e desvio padrão apresentados na tabela 5, foi possível confirmar com um intervalo de confiança 99% que a média da versão KSOM-Cos é superior.

A similaridade cosseno é, conhecidamente, mais adequada para medir a similaridade entre dados esparsos e de alta dimensionalidade, uma vez que favorece comparações do

Tabela 5 – Vitórias, empates e derrotas da versão do MP-Draughts baseada em KSOM-Cos, e desvio padrão considerando as 5 BDs de testes (5-CV).

$KSOM - Cos \times KSOM - DE$							
	1-CV	2-CV	3-CV	4-CV	5-CV	Média	σ
Vitórias	6%	25%	22%	20%	8%	**16%**	8,6
Empates	85%	67%	61%	72%	78%	72%	9,3
Derrotas	9%	8%	18%	9%	14%	11%	4,3

tipo zero-zero. Na verdade, a similaridade cosseno desconsidera esses valores no momento de calcular a similaridade, enquanto a distância Euclidiana não. A distância Euclidiana permite que 2 dados sejam agrupados como similares devido aos valores que não possuem (valores iguais a 0). Por exemplo, considere a seguinte situação: dois estados de tabuleiros não possuem 8 das 15 características, porém as outras 7 possuem valores diferentes de zero e diferentes quando comparados entre si. Nesta situação, a similaridade cosseno tende a encontrar um valor de similaridade pequeno entre eles (pois considera apenas 7 atributos), enquanto a distância Euclidiana tende a identificar uma similaridade maior (pois considera os 15 atributos). Devido a esta característica, a distância Euclidiana não conseguiu obter *clusters* melhores que os obtidos pela similaridade cosseno, de modo a contribuírem para o ganho de desempenho do multiagente.

Devido a performance inferior do MP-Draughts baseado em KSOM-DE, esta versão foi desconsiderada para os próximos cenários. Note que a versão do MP-Draughts baseada em KSOM-Cos avaliada neste cenário já supera sua versão preliminar, originalmente proposta em [23].

3.5.3 Cenário 3: obtenção da melhor versão do MP-Draughts

Os jogos competitivos deste cenário foram executados entre as versões do MP-Draughts baseadas em KSOM-Cos, ART 2A, SONDE e ASONDE. O objetivo deste cenário é detectar qual é a melhor versão do MP-Draughts para a fase de final de jogo. Em cada torneio foram executados 1378 jogos de validação em 689 estados de tabuleiros, os mesmos do cenário anterior.

A Tabela 6 apresenta os resultados dos jogos e o desvio padrão desses, considerando as 5 BDs de testes (5-CV). A versão do MP-Draughts baseada em KSOM-Cos se mostrou mais eficiente que as versões baseadas na ART 2A e na SONDE, sendo que o fator de ganho foi de 5% e 18%, respectivamente. Até este ponto, ainda não foi possível encontrar uma RN adaptativa que fosse capaz de substituir com êxito a KSOM-Cos. A versão do MP-Draughts baseada na ASONDE obteve 11% de ganho sobre a versão baseada na ART 2A e 19% sobre a versão SONDE, sendo esta a melhor entre as versões adaptativas de RNs. Dos jogos realizados entre a versão do MP-Draughts baseada em KSOM-Cos e em ASONDE, a versão ASONDE foi a melhor, apresentando uma superioridade de

desempenho em torno de 5%. A versão do MP-Draughts baseado na ART 2A foi 11% superior que a versão baseada na SONDE, porém ambas não obtiveram bom desempenho quando comparada com as demais versões.

Tabela 6 – Resultado dos jogos de validação entre as versões do MP-Draughts. Os resultados estão apresentados destacando as vitórias, empates e derrotas para o primeiro jogador, além do desvio padrão considerando todos as 5 BDs de testes (5-CV).

$KSOM - Cos \times ART2A$							
	1-CV	2-CV	3-CV	4-CV	5-CV	Média	σ
Vitórias	17%	19%	16%	16%	18%	**17%**	1,3
Empates	73%	66%	66%	75%	72%	70%	3,7
Derrotas	10%	16%	18%	9%	10%	12%	3,7
$KSOM - Cos \times SONDE$							
Vitórias	20%	7%	32%	34%	28%	**25%**	11
Empates	74%	83%	61%	60%	63%	68%	10
Derrotas	6%	10%	7%	6%	9%	7%	1,8
ASONDE $\times ART2A$							
Vitórias	12%	24%	22%	20%	21%	**20%**	4,6
Empates	74%	71%	70%	75%	67%	71%	3,2
Derrotas	14%	5%	9%	5%	12%	9%	4
ASONDE $\times SONDE$							
Vitórias	15%	10%	34%	28%	30%	**23%**	10,3
Empates	76%	86%	64%	70%	65%	72%	9
Derrotas	8%	4%	2%	2%	5%	4%	2,5
ASONDE $\times KSOM - Cos$							
Vitórias	11%	13%	18%	15%	13%	**14%**	2,6
Empates	76%	82%	73%	78%	77%	77%	3,3
Derrotas	13%	5%	8%	7%	11%	9%	3,2
$ART2A \times SONDE$							
Vitórias	22%	7%	31%	36%	25%	**24%**	11
Empates	68%	72%	60%	56%	57%	63%	7
Derrotas	9%	21%	9%	8%	18%	13%	6

Para confirmar se a média de vitórias da versão baseada na ASONDE é superior as das demais versões, também foi aplicado o *teste t* nos jogos envolvendo esta versão. Considerando os valores das médias e desvio padrão apresentados na tabela 6, foi possível confirmar com um intervalo de confiança de 99% que a média da versão ASONDE é superior.

A superioridade de jogo obtida pela versão do MP-Draughts baseado na ASONDE em relação a todas as outras versões, indica que a arquitetura adaptativa desta RN de fato foi a que mais contribuiu para o aumento da performance do multiagente. Além disso, comprova a hipótese de que é possível detectar, utilizando RNs adaptativas, uma arquitetura de SMA mais eficiente do que se utilizando RN de arquitetura fixa. Tais resultados permitem concluir também que o agrupamento obtido pela ASONDE, o qual

foi bastante diferente dos agrupamentos obtidos pela ART 2A e pela KSOM-Cos (ARI inferior a 0,03), agrupou os conhecimentos inerentes a fase de final de jogo de maneira mais apropriada e que pudesse contribuir mais para o aprendizado dos EGAs.

Assim, a versão do MP-Draughts baseada na RN adaptativa ASONDE é a versão que deve compor, juntamente com o IIGA, a arquitetura final do MP-Draughts.

3.5.4 Cenário 4: avaliação da performance do MP-Draughts contra outros jogadores automáticos não supervisionados

Este cenário apresenta a performance da arquitetura final do MP-Draughts em jogos competitivos contra os jogadores monoagentes não supervisionados NeuroDraughts e VisionDraughts. Os estados de tabuleiros utilizados neste cenário foram filtrados da BD OCA-2.0 [91], os quais não foram utilizados no processo de treinamento do MP-Draughts. Foram executados 80 jogos em 40 estados de tabuleiros diferentes, mantendo a alternância das cores de peças dos jogadores em cada tabuleiro.

A Tabela 7 apresenta os resultados dos jogos entre eles em todas as 5 BDs de testes. Note que o MP-Draughts se mostrou mais eficiente que seus oponentes na maioria das BDs. Nos jogos contra o NeuroDraughts, o multiagente obteve 11% de vitórias (com σ = 4,6) contra apenas 4% (com σ = 4) do oponente. Nos jogos contra o VisionDraughts, o multiagentes venceu 15% (com σ = 3,8) contra 8% (com σ = 2,7) do oponente. Observe que em ambos os torneios o multiagente obteve uma média de 7% de superioridade (vitórias) em relação aos oponentes. Tais resultados reforçam a eficiência da ASONDE em detectar os perfis existentes na fase de final de jogo, uma vez que os vários EGAs obtidos por essa RN mostraram conhecer melhor a fase de final de jogo do que um agente simples que conhece o jogo como um todo. Tais resultados comprovam a hipótese de que jogadores de Damas multiagentes são mais eficientes que os jogadores monoagentes, o que se deve a visão especialista que cada EGA possui sobre uma determinada "região" de conhecimento.

Tabela 7 – Resultado dos testes avaliativos contra outros jogadores não supervisionados. Os resultados destacam as vitórias, empates e derrotas do MP-Draughts.

$MP-Draughts \times NeuroDraughts$							
	1-CV	2-CV	3-CV	4-CV	5-CV	Média	σ
Vitórias	10%	8%	18%	6%	11%	**11%**	4,6
Empates	79%	89%	79%	93%	88%	85%	6,3
Derrotas	11%	4%	4%	1%	1%	4%	4
$MP-Draughts \times VisionDraughts$							
Vitórias	20%	16%	16%	11%	11%	**15%**	3,8
Empates	76%	75%	76%	78%	79%	77%	1,6
Derrotas	4%	9%	8%	11%	10%	8%	2,7

Tabela 8 – Média de coincidência de movimentos executados pelos jogadores não supervisionados comparado com os que seriam executados pelo Cake na mesma situação.

Jogos	MP-Draughts	Oponente
$MP-Draughts \times NeuroDraughts$	**62,3%**	50,2%
$MP-Draughts \times VisionDraughts$	**65,4%**	44,5%

Não foram realizados jogos entre o MP-Draughts e os outros jogadores automáticos não supervisionados citados nos trabalhos relacionados a esta pesquisa porque, de acordo com os autores desses jogadores [26], [25], não existe uma interface disponível desses agentes.

3.5.5 Cenário 5: avaliação das escolhas de movimentos sobre a perspectiva do Cake

Este cenário compara os movimentos coincidentes executados pelos agentes não supervisionados MP-Draughts, NeuroDraughts e VisionDraughts, os quais seriam executados pelo jogador fortemente supervisionado Cake na mesma situação. Esta comparação mostra o quão próximo é o raciocínio dos agentes não supervisionados em relação ao agente supervisionado na fase de final de jogo. Para tanto, foram comparados apenas os movimentos executados nesta fase do jogo. Tais agentes não foram avaliados em jogos diretos contra o Cake, pois os mesmos ainda não são competitivos contra agentes que contam com forte supervisão no processo de aprendizagem.

Para a execução deste cenário foram avaliados 6 dos 80 jogos executados no cenário 4 entre o MP-Draughts e cada um de seus oponentes, mantendo a alternância das cores das peças (ora preta ora vermelha) dos jogadores. Foram analisados todos os movimentos executados até a finalização do jogo. Em média, foram analisados 20 movimentos de cada jogador em cada jogo. A Tabela 8 apresenta os resultados desta comparação. Nos jogos contra o NeuroDraughts, o MP-Draughts obteve 62,3% de movimentos coincidentes com os movimentos que seriam executados pelo Cake, enquanto que o NeuroDraughts obteve 50,2% de coincidência. Nos jogos contra o VisionDraughts esta taxa subiu para 65,4%, contra 44,5% do VisionDraughts. Os resultados novamente confirmam a eficiência do MP-Draughts, visto que suas escolhas de movimento coincidem, na maioria dos casos, com as escolhas de um agente comprovadamente eficiente e supervisionado. Além disso, confirma a superioridade do MP-Draughts em relação aos seus oponentes não supervisionados, uma vez que a coincidência de raciocínio do multiagente é maior que a de seus oponentes.

3.6 Considerações Finais

Este capítulo apresentou o multiagente não supervisionado jogador de Damas MP-Draughts, cuja nova proposta de arquitetura é baseada em RN adaptativas e MLP. O

objetivo deste capítulo foi otimizar a versão preliminar do multiagente, cuja arquitetura era baseada em KSOM-DE e MLP. A substituição da KSOM-DE por uma RN adaptativa, possibilitou a automatização do processo que define a quantidade adequada de *clusters* para representar a BD de final de jogo utilizada no treinamento dos agentes especialistas nessa fase (EGAs), além encontrar a quantidade mais apropriada destes para compor a arquitetura de final do jogo do multiagente. No intuito de encontrar a melhor RN adaptativa pra tal tarefa, investigou-se a ART 2A, a SONDE, a ASONDE e a KSOM-Cos. A ASONDE corresponde a uma nova proposta de RN adaptativa, baseada na SONDE, para lidar com BD finitas e estáveis. Investigou-se também qual das medida de similaridade, distância Euclidiana e similaridade cosseno, é a mais apropriada para medir a semelhança entre os estados de tabuleiros de final de jogo. A KSOM-Cos estava entre as RNs analisadas porque esta é a versão da KSOM que utiliza similaridade cosseno e sua investigação fez-se necessária para que fosse possível distinguir o ganho obtido pela mudança da medida de similaridade e o ganho obtido pela adaptabilidade das RNs.

Os experimentos realizados demostraram que: a medida de similaridade cosseno é mais indicada para medir a semelhança entre os estados de tabuleiros da BD de final de jogo; que a ASONDE foi a RN que mais contribuiu para a otimização do multiagente, sendo que os EGAs treinados nos *clusters* obtidos por esta RN são melhores que os EGAs treinados nos *clusters* produzidos pelas outras RNs aqui investigadas; que a versão do multiagente baseada nesta RN também se mostrou mais eficiente que os agentes jogadores de Damas não supervisionados NeuroDraughts e VisionDraughts e, finalmente; que o raciocínio do multiagente na fase de final de jogo é, pelo menos, 62% coincidente com o raciocínio do excelente jogador de Damas supervisionado Cake. Sendo assim, os resultados obtidos neste capítulo atenderam algumas das contribuições pretendidas por este trabalho, que são: encontrar uma arquitetura de SMA cuja estrutura fosse adequada para o domínio de jogos de Damas e ainda, propor uma RN adaptativa adequada para reconhecer os diferentes perfis inerentes a fase de final de jogo um jogo de Damas.

Contudo, apesar do MP-Draughts ter conseguido superar sua versão anterior e ter se mostrado melhor que vários agentes não supervisionados, ainda existem limitações na sua arquitetura que, se resolvidas, tendem a melhorar ainda mais seu desempenho. Tais limitações são: a maioria das características da representação *NetFeatureMap*, as quais foram escolhidas manualmente para representar os tabuleiros do jogo, não estão presentes em grande parte dos tabuleiros (valores dos atributos referentes a essas características possuem valor zero), de modo que a representação das informações relevantes desses ambientes podem estar sendo "subestimadas" (ou mal representadas) pelas características escolhidas e; as alocações dos EGAs para atuar na fase de final de jogo ainda estão sendo inadequadas em algumas situações, sendo que nem mesmo a substituição da RN foi capaz de melhorar essas situações.

Capítulo 4

Aprimorando a Representação dos Ambientes de Atuação de Agentes Inteligentes baseado na Mineração de Itens Frequentes

A representação adequada dos estados sobre os quais agentes inteligentes atuam é fundamental para sua boa performance, particularmente quando eles atuam em ambientes competitivos que possuem elevado espaço de estados. Um tipo particular de representação, adequada para este tipo de ambiente, é a representação *NetFeatureMap* [14], a qual representa o ambiente baseando-se em um conjunto de funções, denominadas características, que capturam conhecimentos relevantes atribuídos ao domínio do problema. Diferentemente da representação vetorial, que representa todas as informações contidas no ambiente, a *NetFeatureMap* provê uma percepção otimizada do ambiente de modo a representar apenas informações que, de fato, contribuem para a atuação (tomada de decisão) dos agentes. Tal percepção permite que tais agentes sejam mais ágeis na exploração do espaço de estados, visto que aumenta sua visão (*look-ahead*) sobre o ambiente e assim, melhora suas habilidades em escolher adequadamente as ações a serem executadas. Vários agentes automáticos propostos na literatura usam a *NetFeatureMap* para representar os estados dos ambientes sobre os quais atuam. Em alguns desses agentes, as características utilizadas para a representação foram selecionadas manualmente, o que leva esses agentes a efetuar escolhas inadequadas de ações, comprometendo assim sua performance [21], [6]. Em outros, tal seleção foi feita automaticamente utilizando AGs, o que propiciou a escolha adequadas das características e consequentemente, contribuiu para o aumento da performance desses agentes [96], [29]. Motivados pelos bons resultados obtidos pela seleção automática de características, este capítulo propõe uma nova abordagem para a seleção automática dessas características, a qual é baseada na mineração de padrões frequentes de BDs. Tal abordagem representa a frequência com que essas

características ocorrem nos estados explorados pelos agentes ao longo do tempo. Assim como no capítulo anterior, o domínio do jogo de Damas foi utilizado como ambiente de desenvolvimento e investigação da qualidade da abordagem aqui proposta, uma vez que se trata de um ambiente competitivo e com alta complexidade de espaço de estados [2].

O primeiro jogador automático de Damas a utilizar a *NetFeatureMap* para representar o ambiente do jogo foi o jogador de Arthur Samuel [14], o qual a utiliza para representar o estado de tabuleiro corrente do jogo no momento da tomada de decisão (escolha de um movimento) de seu agente. Tal representação também foi utilizada pelos vários agentes jogadores propostos pela equipe de pesquisa na qual este trabalho se insere, tais como o VisionDraughts [21], D-VisionDraughts [24], LS-VisionDraughts [29] e MP-Draughts [23], [6]. Particularmente, nos agentes VisionDraughts, D-VisionDraughts e MP-Draughts, as características utilizadas para representar os estados de tabuleiros foram selecionadas manualmente, enquanto que no LS-VisionDraughts tal seleção foi feita automaticamente por um AG. De acordo com os autores do LS-VisionDraughts, apesar da melhora significativa da performance desse agente, tal abordagem demandou de muito tempo de processamento para produzir bons resultados (aproximadamente 4 meses de execução contínua). Os autores argumentam que seria muito interessante investigar outras alternativas mais eficientes para a seleção automática de características, principalmente quando a seleção envolver problemas que possuem elevado espaço de estados, tal como o jogo de Damas. Quanto maior o espaço de estados passíveis de exploração num ambiente, maior deve ser o tempo de processamento para selecionar as características adequadas para representar tal espaço. Outro fato que deve ser considerado quanto ao uso do AG é que sua performance está diretamente relacionada a qualidade da sua função de avaliação. No caso do LS-VisionDraughts, a função de avaliação depende da profundidade da árvore de busca (montada durante a busca pelo melhor movimento) e do nível de experiência do agente, uma vez que tal função é baseada nos resultados obtidos pelo agente em torneios competitivos. De fato, se a árvore de busca é muito rasa, ela compromete a visão de jogo (*look ahead*) do agente e, por consequência, a escolha dos movimentos. Por outro lado, se a árvore é muito profunda, o tempo de evolução do AG se torna impraticável. Além disso, nos casos em que o agente não tem muita experiência, as características consideradas por ele no momento de analisar o tabuleiro e escolher um movimento, podem não ser as mesmas que seriam consideradas por grandes mestres do jogo. Outro fator que motiva tal investigação no domínio do jogo de Damas é que, como identificado no capítulo anterior, as características *NetFeatureMap*, selecionadas manualmente para representarem os tabuleiros dos jogos para multiagente MP-Draughts, não estavam presentes na maioria dos estados de tabuleiros apresentados às RNs de agrupamento (valores esparsos que justificaram a melhor performance da medida de similaridade cosseno), o que significa que muitas delas não foram relevantes para o problema. Além disso, as características relevantes para uma determinada fase do jogo não necessariamente são relevantes para

todo o jogo e, por se tratar de um sistema composto por agentes especialistas em distintas fases do jogo, é importante para o MP-Draughts contar com características que de fato sejam relevantes para cada momento do jogo, as quais possam melhor contribuir para sua tomada de decisão.

Nesse sentido, a abordagem aqui proposta tem o objetivo de selecionar automaticamente as características, de maneira que os resultados obtidos sejam menos dependentes das variáveis relacionadas a técnica de seleção. Tal abordagem foi inicialmente avaliada para selecionar as características adequadas para representar os estados de tabuleiros que ocorrem durante todo o jogo. Comprovada a eficiência desta abordagem, cujos resultados foram comparados com os obtidos pelo AG de [29], a mesma foi aplicada a BDs de estados de tabuleiros de final de jogo para descobrir quais são as características adequadas para esta fase do jogo.

As próximas seções desse capítulo apresentam a aplicação da abordagem para todo o jogo e também, para a fase de final de jogo. Tais seções estão organizadas como segue: a Seção 4.1 explica a aplicação da abordagem para obter as características relevantes para todo o jogo, assim como a obtenção dos conjuntos de características mais frequentes e os resultados obtidos por tal abordagem quando comparado com os resultados obtidos pelo AG; a Seção 4.2 apresenta sua aplicação para a fase de final de jogo, os conjuntos de características mais frequentes nesta fase e os resultados obtidos pela abordagem. Considerando os resultados aqui obtidos, este capítulo cumpre mais uma parte do objetivo geral e o objetivo específico 2 propostos no presente trabalho (ver Seção 1.2).

4.1 Mineração de Padrões Frequentes e Seleção Automática de Características

Como citado anteriormente, vários jogadores automáticos propostos na literatura utilizam conjuntos de características da *NetFeatureMap* para representar os estados de tabuleiros, cujas características foram obtidas manual ou automaticamente. Nesta seção é apresentada uma abordagem capaz de selecionar automática e eficientemente as melhores características para representar o ambiente de Damas durante um jogo completo. Tal abordagem é baseada na mineração de padrões frequentes, os quais são extraídos de uma BD especializada que contém o histórico (estados de tabuleiros) de milhares de jogos disputados por grandes mestres do domínio ao longo do último século [91]. Os padrões frequentes obtidos dessa BD correspondem aos conjuntos de características que frequentemente ocorrem nos estados de tabuleiros, os quais foram considerados pelos mestres jogadores no momento em que deveriam escolher um movimento.

A estratégia aqui adotada pode ser resumida assim: inicialmente, é gerada uma BD contendo vários estados de tabuleiros, representados vetorialmente, que ocorreram ao longo dos jogos disputados pelos grandes mestres no domínio. Tais estados de tabulei-

ros são convertidos para a representação *NetFeatureMap* e, na sequência, os estados com maior frequência relativa são extraídos da BD. Como cada um desses tabuleiros estão representados por um determinado conjunto de características, os mesmos correspondem aos conjuntos de características que mais ocorreram nos jogos avaliados. Esses conjuntos são então considerados como possíveis candidatos para representar o ambiente do jogo de Damas, os quais devem ser avaliados em cenários de jogos competitivos envolvendo jogadores automáticos que utilizam a *NetFeatureMap*, para que seja possível averiguar a qualidade dessas possíveis representações. Observe que os conjuntos de características obtidos são independentes da qualidade de uma função de avaliação, ao contrário, representam o mais refinado conhecimento considerado por grandes jogadores.

As próximas subseções apresentam todas as atividades envolvidas no processo de seleção dos melhores conjuntos de características para representar todo o ambiente do jogo de Damas.

4.1.1 Geração da Base de Dados especializada

A BD especializada utilizada aqui foi gerada a partir de jogos que ocorreram em campeonatos mundiais de Damas envolvendo grandes mestres do jogo, os quais estão disponíveis na plataforma CheckerBoard em [91]. Tais jogos são interessantes para a obtenção da BD pois correspondem às (boas) situações que tendem a ocorrer em jogos, além de representarem o valioso conhecimento considerado pelos mestres jogadores no momento de suas tomadas de decisão. Dentre os vários jogos disponibilizados na plataforma, nesta abordagem foram utilizados os jogos disponíveis na BD OCA2.0 e não apenas os jogos disponíveis na BD do Tinsley como utilizado no capítulo anterior. Tal alteração foi feita por dois motivos: aumentar a quantidade de estados de tabuleiros representados na BD, uma vez que a BD OCA2.0 é consideravelmente maior, e aumentar a variabilidade desses tabuleiros, o que é possível porque na OCA2.0 existem jogos envolvendo vários jogadores e não apenas jogos do Tinsley. Apenas os jogos cujo resultado final era diferente de empate foram considerados, totalizando 8.429 jogos. Os jogos de empate foram desconsiderados porque o interesse aqui é representar na BD o conhecimento existente nos tabuleiros que levaram os agentes a obter bom desempenho de jogo, o que num caso de empate não é claro. Para a geração da BD, todos os 8.429 jogos foram re-executados e, durante a re-execução de cada jogo apenas os estados de tabuleiros apresentados ao agente *vencedor* do referido jogo para a execução de um movimento foram armazenados na BD. O motivo de se considerar apenas os estados de tabuleiros apresentados ao vencedor, é para garantir que a BD conterá apenas os conhecimentos relacionados a situações de sucesso no jogo. Finalizado o processo de obtenção da BD, todos os estados de tabuleiros nela contidos foram convertidos para a representação *NetFeatureMap*.

A Figura 17 mostra um exemplo de um estado de tabuleiro na representação vetorial. O mapeamento C apresentado na sequência, ilustra como é feito a conversão de um

tabuleiro que está na representação vetorial para sua representação *NetFeatureMap*.
C: $I_i \longrightarrow \{0,1\}^n$,
$C(I_i) =< f_1(I_i), ..., f_n(I_i) >$, e
$f_j(I_i) = 1$, se a característica f_j está presente no estado de tabuleiro I_i,
$f_j(I_i) = 0$, caso contrário.
O valor de n corresponde a quantidade de características f_j $(1 \leq j \leq n)$ utilizadas para representar o estado de tabuleiro na *NetFeatureMap*. Ou seja, cada estado de tabuleiro I_i é representado por uma *n-tupla* composta de n atributos, os quais representam a presença ou a ausência das características f_1,..., f_n, respectivamente. Para representar os estados de tabuleiros na *NetFeatureMap* foram utilizadas as 15 características apresentadas na Tabela 2.

1	1	1	0	0	1	1	0	0	0	0	2	1	0	0	0	0	0	2	1	2	0	0	0	2	0	2	0	0	2	0	2

Figura 17 – Exemplo de um estado de tabuleiro na representação vetorial

A Figura 18 mostra um exemplo de estado de tabuleiro na representação *NetFeatureMap*, o qual indica que as características f_2, f_6, f_{13}, f_{14} e f_{15} estavam presentes no estado de tabuleiro quando este foi analisado pelo jogador no momento de decidir qual movimento executar. É importante observar que o mapeamento C é do tipo "muitos pra um", ou seja, vários estados de tabuleiros vetoriais podem ser mapeados para uma única representação *NetFeatureMap*.

0	1	0	0	0	1	0	0	0	0	0	0	1	1	1
f1	*f2*	*f3*	*f4*	*f5*	*f6*	*f7*	*f8*	*f9*	*f10*	*f11*	*f12*	*f13*	*f14*	*f15*

Figura 18 – Exemplo de um estado de tabuleiro na representação *NetFeatureMap*

Diferente do que foi apresentado no capítulo de fundamentação teórica (Capítulo 2), em que as características são representadas qualitativa e quantitativamente, neste capítulo, para a mineração de padrões frequentes foi considerada apenas a representação qualitativa (binária) dessas características. Tal representação é suficiente, uma vez que é necessário saber apenas quais características são relevantes para representar os tabuleiros do jogo. Sabendo isso, a utilização destas para representar o tabuleiro do jogo no momento das tomadas de decisão dos agentes serão feitas qualitativa e quantitativamente (conforme Capítulo 2). Outro motivo em se considerar a representação binária dessas características no momento da mineração de padrões é para garantir a confiabilidade dos resultados obtidos desta abordagem quando comparado com os resultados obtidos pelo AG. Da mesma forma, as características utilizadas aqui são as mesmas utilizadas pelo AG.

Finalizado o processo de re-execução dos 8.429 jogos, a BD especializada contém 92.636 estados de tabuleiros, os quais estão representados seguindo a *NetFeatureMap*.

4.1.2 Mineração dos Padrões Frequentes e Seleção dos Melhores Conjuntos de Características

No intuito de selecionar os melhores conjuntos de características para representar o jogo, foi feita uma análise estatística da BD obtida na seção anterior. O objetivo dessa análise foi detectar e selecionar os estados de tabuleiros, representados por características, mais frequentes que ocorreram durante os jogos representados nessa BD. Tais estados de tabuleiros representam os conjuntos de características frequentemente analisados pelos jogadores nos momentos de tomadas de decisão. A seleção desses conjuntos foi feita considerando a frequência relativa de sua ocorrência na BD. Todos os conjuntos cuja frequência ficaram acima de 10% foram selecionados, ou seja, o suporte mínimo s considerado para a extração dos padrões frequentes foi de $0,1$.

Como o mapeamento C (que converte os estados de tabuleiros para a representação *NetFeatureMap*) é do tipo "muitos pra um", a frequência de ocorrência de um determinado conjunto de características na BD não corresponde a frequência de ocorrência de um determinado estado de tabuleiro nos jogos analisados. Isso que dizer que o conjunto de atributos considerados pelo jogador no momento de uma tomada de decisão pode ser o mesmo, ainda que em situações distintas do jogo, fato que reforça a importância da representação do ambiente em todas as tomadas de decisões do agente jogador.

Foram selecionados 26 conjuntos de características distintos, os quais foram considerados como os possíveis melhores conjuntos para representar o ambiente do jogo. Para avaliar a qualidade de representação desses 26 conjuntos, foi usado como ambiente de investigação o jogador automático VisionDraughts. Para tanto, foram criadas 26 novas versões desse jogador, as quais foram treinadas, cada uma, num conjunto de características distinto. O treinamento dessas versões foram executados de maneira análoga ao apresentado no Capítulo 3, Seção 3.2. Cada versão do VisionDraughts foi treinada por 4 ciclos de 400 jogos cada, totalizando 1.600 jogos de treinamento. A única diferença no treinamento de cada versão está no conjunto de características utilizado por elas para representar o estado de tabuleiro na entrada da MLP para avaliação dos possíveis movimentos. Finalizado o processo de treinamento, existiam 26 versões distintas do VisionDraughts, cada uma com diferentes habilidades de jogos, as quais foram modeladas de acordo com a visão que cada uma tinha no jogo nos momentos de tomadas de decisão.

No intuito de avaliar a qualidade de jogo agregado a essas versões em relação a outros jogadores cujas características foram selecionadas por outros métodos, foram executados torneios competitivos entre essas versões contra os seguintes agentes: LS-VisionDraughts, cujas características foram selecionadas automaticamente por um AG, e a versão original do VisionDraughts, cujas características foram manualmente selecionadas. A eficiência obtida pela abordagem foi medida a partir do desempenho de jogo dos agentes e do tempo de processamento de cada abordagem.

4.1.3 Resultados Experimentais

Esta seção avalia a eficiência da abordagem proposta em 4 cenários de testes distintos, os quais avaliam o ganho de performance do jogador VisionDraughts proporcionado pela seleção automática das características e a superioridade desta abordagem quando comparada com a abordagem baseada em AG. Nesse sentido, os cenários de testes foram divididos como segue: o primeiro avalia a performance de jogo das 26 novas versões do VisionDraughts, as quais representam as diferentes propostas de características para representar o jogo do Damas; o segundo avalia o quão eficiente a seleção automática é quando comparada com a seleção manual; o terceiro avalia a performance das melhores versões do VisionDraughts obtidas no cenário 1, cujas características foram selecionadas pela presente abordagem, em torneios contra o LS-VisionDraughts cujas características foram selecionadas por um AG e; finalmente, o quarto cenário compara o tempo de execução requerido por ambas as abordagens para obter os conjuntos de características apropriados para representar os diferentes ambientes que ocorrem no domínio do jogo de Damas.

Para garantir a confiabilidade dos resultados obtidos, todos os agentes envolvidos nos cenários de testes foram treinados nas mesmas condições, ou seja, foram treinados por 4 ciclos de 400 jogos, usando o algoritmo de busca Alfa-Beta, combinado com TT e AI com profundidade máxima de busca igual a 8.

4.1.4 Cenário 1: definindo os melhores conjuntos de características

Este cenário determina quais são os melhores conjuntos de características, dentre os 26 extraídos da BD, para representar o ambiente do jogo. Para isso, foram executados torneios competitivos entre todas as 26 versões do VisionDraughts, cada uma treinada num distinto conjunto de características. Considerando que a única diferença no treinamento dessas versões são as características usadas por elas para representar os tabuleiros do jogo, qualquer vantagem de jogo obtida por uma ou outra versão está relacionada a esta distinção. Dentre as 26 versões, 10 obtiveram a taxa de vitórias superior a 50%, quando comparado com o somatório das taxas de empates e derrotas. Esse resultado mostra que os conjuntos de características utilizados por essas 10 versões foram mais eficientes em representar o ambiente de atuação dos jogadores, ou seja, tais conjuntos representam características mais relevantes para o aprendizado e desempenho dos seus respectivos agentes do que os outros 16 conjuntos.

Contudo, este resultado não é suficiente para garantir que esses 10 melhores conjuntos sejam de fato os mais adequados para representar os tabuleiros, ele garante apenas que dentre os verificados, tais conjuntos são os melhores. Sendo assim, para verificar se estes conjuntos são melhores que os utilizados por outros jogadores que utilizam a *NetFeatureMap*, os próximos cenários investigam essa situação.

4.1.5 Cenário 2: avaliando a eficiência da seleção automática de características

Este cenário avalia, por meio de torneios competitivos, se os 10 melhores conjuntos de características, vinculados as 10 melhores versões do VisionDraughts obtidos no cenário anterior, são mais eficientes para representar o ambiente de atuação dos jogadores do que o conjunto completo contendo as 15 características. Essas 15 características foram selecionadas manualmente, pelos autores do LS-VisionDraughts, de um conjunto maior contendo 26 características [14], as quais foram mantidas para estabelecer condições iguais de comparação dos resultados. Para avaliar este cenário, a versão original do VisionDraughts foi treinada utilizando essas 15 características. Note que os estados de tabuleiros a serem avaliados pela versão original do VisionDraughts durante os torneios, são representados considerando as 15 características, enquanto os tabuleiros apresentados as 10 melhores versões utilizam apenas subconjuntos dessas características.

Tabela 9 – Resultados dos jogos entre a versão original do VisionDraughts e suas 10 melhores versões treinadas nos conjuntos de características selecionadas automaticamente. Os resultados ilustram as vitórias, empates e derrotas para as diferentes versões do VisionDraughts.

Versões VisionDraughts x VisionDraughts						
	Vitórias		Empates		Derrotas	
V_1	23	77%	6	20%	1	3%
V_2	11	37%	12	40%	7	23%
V_3	13	43%	5	17%	12	40%
V_4	9	30%	13	43%	18	27%
V_5	10	33%	14	47%	6	20%
V_6	17	57%	10	33%	3	10%
V_7	9	30%	10	33%	11	37%
V_8	7	23%	14	47%	9	30%
V_9	13	44%	16	53%	1	3%
V_{10}	25	84%	4	13%	1	3%

Foram executados 30 jogos entre a versão original do VisionDraughts e cada uma de suas 10 melhores novas versões. A Tabela 9 apresenta os resultados desses jogos, na qual as 10 melhores versões são identificadas como $V_1, V_2, ..., V_{10}$. Os resultados mostram a quantidade e a porcentagem de vitórias, empates e derrotas obtidas por essas versões. Como destacado em cinza na tabela, 8 das 10 versões foram melhores que a versão original do jogador, ou seja, 8 dos 10 conjuntos automaticamente selecionados pela abordagem proposta foram mais eficientes em representar o ambiente do jogo do que a seleção manual. Em outras palavras, a abordagem de seleção automática foi 80% mais eficiente que a seleção manual. Tal resultado comprova que a seleção automática de característica aqui proposta em geral é mais eficiente que a seleção manual. Outros estudos, baseados em diferentes abordagens de seleção, também já comprovaram que a seleção automática de

características, seja *NetFeatureMap* ou outras, é de fato mais eficiente que a seleção manual [29], [96], [97].

4.1.6 Cenário 3: avaliando a performance dos melhores jogadores

Este cenário avalia a performance de jogo das 10 melhores versões do VisionDraughts em competições contra o jogador LS-VisionDraughts. Esta avaliação é muito pertinente visto que o LS-VisionDraughts é o jogador, dentre todos os que utilizam a *NetFeatureMap* para representar o ambiente do jogo de Damas que possui o melhor conjunto de características [29].

Foram executados 30 jogos entre o LS-VisionDraughts e cada uma das versões do VisionDraughts ($V_1, V_2, ..., V_{10}$). A Tabela 10 apresenta os resultados desses jogos, os quais ilustram a quantidade e a porcentagem de vitórias, empates e derrotas, obtidas pelas versões do VisionDraughts. Como destacado em cinza na tabela, as versões V_1, V_2, V_3, V_6, V_9 e V_{10} obtiveram maior porcentagem de vitórias que o LS-VisionDraughts. Observe que 4 dessas versões ($V_1, V_3, V_6 e V_{10}$) foram pelo menos 50% mais eficientes que o LS-VisionDraughts e que as versões V_6 e V_{10} obtiveram pelo menos 71% de superioridade.

Tabela 10 – Resultados dos jogos entre o LS-VisionDraughts e as 10 melhores versões do VisionDraughts. Os resultados ilustram as vitórias, empates e derrotas para as diferentes versões do VisionDraughts.

Versões VisionDraughts x LS-VisionDraughts						
	Vitórias		Empates		Derrotas	
V_1	23	77%	3	10%	4	13%
V_2	12	40%	9	30%	9	30%
V_3	21	70%	4	13%	5	17%
V_4	6	20%	13	43%	11	37%
V_5	11	37%	7	23%	12	40%
V_6	22	74%	7	23%	1	3%
V_7	9	30%	7	23%	14	47%
V_8	7	23%	6	20%	17	57%
V_9	9	30%	16	53%	5	17%
V_{10}	23	77%	6	20%	1	3%

Os conjuntos de características utilizados pelas 6 melhores versões do VisionDraughts são:

- V_1: *PieceAdvantage, XcentreControl, TotalMobility, Exposure, DoubleDiagonal, DiagonalMoment, Threat, Taken;*
- V_2: *BackrowBridge, CentreControl, XcentreControl, TotalMobility, DoubleDiagonal, DiagonalMoment, Taken;*

- V_3: *PieceAdvantage, CentreControl, XcentreControl, TotalMobility, Exposure, DoubleDiagonal, DiagonalMoment, Threat, Taken;*
- V_6: *PieceAdvantage, XcentreControl, TotalMobility, Exposure, DoubleDiagonal, DiagonalMoment, Taken;*
- V_9: *BackrowBridge, XcentreControl, TotalMobility, Exposure, DoubleDiagonal, DiagonalMoment, Threat;*
- V_{10}: *PieceAdvantage, PieceThreat, BackrowBridge, CentreControl, XcentreControl, TotalMobility, Exposure, DoubleDiagonal, DiagonalMoment, Threat, Taken;*

Comparando esses conjuntos com o obtido pelo AG do LS-VisionDraughts - *PieceAdvantage*, *PieceDisadvantage*, *PieceThreat*, *PieceTake*, *DoubleDiagonal*, *BackrowBridge*, *CentreControl*, *KingCentreControl*, *Threat* - é possível identificar que os conjuntos obtidos aqui possuem entre 7 e 11 características, contra 9 do LS-VisionDraughts. Vale ressaltar que, quanto menor a quantidade de características utilizadas para representar os estados de tabuleiros, menor é o tempo de treinamento do jogador. Note que os conjuntos de características vinculados as versões V_1, V_2, V_6, e V_9 possuem menos de 9 características, ou seja, o treinamento desses agentes é computacionalmente mais eficiente que o treinamento do LS-VisionDraughts. As versões V_2, V_6 e V_9 obtiveram boa performance de jogo usando apenas 7 características para representar o ambiente, o que representa uma redução de aproximadamente 53% das características da *NetFeatureMap* sem prejudicar o desempenho desses jogadores.

Considerando o desempenho em jogo e a quantidade de características utilizadas, a versão V_6 do VisionDraughts foi a que mais se destacou. Esta versão foi muito superior ao LS-VisionDraughts e utiliza apenas 47% das características aqui investigadas.

Outro ponto interessante a ser destacado é que nem todos os conjuntos de características mais frequentes obtidos da análise estatística da BD foram avaliados. Isso significa que ainda podem existir outros conjuntos, dentre os que ocorreram em menos de 10% da BD, que sejam mais eficientes que o obtido pelo AG do LS-VisionDraughts.

4.1.7 Cenário 4: comparando o tempo de execução de ambas as abordagens

Este cenário compara o tempo de execução requerido pela abordagem proposta e pelo AG para obterem suas melhores propostas de conjuntos de características suficientemente adequados para representar o domínio do jogo de Damas. O tempo de execução requerido pela abordagem de mineração de padrões corresponde a soma dos tempos para: geração da BD, identificação dos itens frequentes, extração dos conjuntos de características frequentes e, treinamento das 26 versões do VisionDraughts. O tempo de execução requerido pelo AG

corresponde ao processo evolutivo de 30 gerações com 40 indivíduos (MLP), cuja função de avaliação corresponde ao treinamento e a torneios competitivos executados entre todos os indivíduos de uma mesma geração.

O tempo de execução da abordagem aqui proposta foi de aproximadamente 90 horas, sendo que 78 horas corresponde ao tempo de treinamento das 26 versões do VisionDraughts e apenas 12 horas para a geração e mineração da BD especializada. Por outro lado, o tempo de execução do AG foi aproximadamente 3.572 horas. O tempo estimado para o AG foi calculado considerando o tempo de treinamento de cada indivíduo apresentado em [29], que é de 178,6 minutos por indivíduo. O tempo de execução da abordagem proposta representa apenas 2% do tempo gasto pelo AG.

Além de ter obtido 6 conjuntos de características superiores ao obtido pelo AG, a abordagem proposta obteve os resultados em apenas 2% do tempo gasto por ele. Isso é devido a 3 vantagens dessa abordagem: primeiro, os conjuntos de características foram obtidos a partir de informações reais que ocorrem em excelentes jogos de Damas; segundo, o treinamento dos jogadores ocorre um única vez, apenas quando as características frequentes são encontradas e; terceiro, o tempo de treinamento da maioria das versões do VisionDraughts foi menor, visto que o conjunto de características vinculadas a elas também o são.

4.2 Mineração de Padrões Frequentes e Seleção Automática de Características aplicada à Fase de Final de Jogo

Comprovada a eficiência e a superioridade da abordagem baseada em mineração de itens frequentes para selecionar automaticamente as características adequadas para representar o ambiente de atuação dos agentes durante todo o jogo de Damas, esta seção a utiliza para selecionar as características adequadas para representar os estados de tabuleiros que frequentemente ocorrem nas fases de final de jogo. Relembrando, neste trabalho "fase de final de jogo" remete a tabuleiros que contenham no máximo 14 peças.

Como pode ser observado na Tabela 2 (Capítulo 2, seção 2.3.5.2), existem características que são passíveis de ocorrerem com mais frequencia em determinadas fases do jogo, tais como *CentreControl*, *XcentreControl*, *Exposure* e *Kingcentrecontrol*. Tais características estão relacionadas a evolução do jogo em direção ao território do oponente e a formação de Damas, o que normalmente ocorrem nas fases finais de um jogo. Diante disso, acredita-se que assim como existem características que são frequentes durante todo o jogo, existem aquelas que são ainda mais frequentes em uma e/ou outra fase. Como o ambiente de desenvolvimento e avaliação das técnicas propostas neste trabalho é a fase final do jogo de Damas, não faria sentido que tal abordagem não fosse aplicada a esta

fase. Afinal, quanto mais refinada puder ser a visão de um agente sobre o ambiente, mais chances ele terá de efetuar boas ações.

Sabendo que o MP-Draughts possui uma arquitetura moldada de acordo com os diferentes perfis (*clusters*) de conhecimento que podem ocorrer na fase de final de jogo, é coerente que sejam encontradas, para cada perfil, as características que ocorrem com maior frequência entre estados de tabuleiros que os representam. A BD utilizada para representar esses perfis foi alterada para a OCA2.0, também com o intuito de aumentar a quantidade e a representatividade dos estados de tabuleiros que caracterizam cada perfil do jogo. Dessa maneira, todo o processo de agrupamento e definição da arquitetura do multiagente para a fase de final de jogo utilizando a ASONDE teve que ser re-executado (para relembrar tal processo veja Capítulo 3). Como o intuito era aumentar a representatividade de cada perfil, foram gerados *clusters* contendo uma quantidade maior de estados de tabuleiro de final de jogo, de modo que foram gerados 4 deles. Em cada um desses *clusters* foi aplicada a abordagem de mineração de padrões frequentes de maneira a identificar os conjuntos de características mais frequentes entre os tabuleiros contidos em cada um.

A estratégia adotada para aplicar a abordagem em cada um dos *clusters* de final de jogo foi a mesma adotada para todo o jogo. As Subseções 4.2.1 e 4.2.2 descrevem o processo de obtenção das BDs especializadas (uma para cada *cluster*) e dos conjuntos de características frequentes em cada *cluster*. A seção de resultados discute o resultado em termos do ganho de performance de jogo obtido da utilização desses conjuntos para representar o ambiente de atuação dos agentes de final de jogo (EGAs) do MP-Draughts.

4.2.1 Obtenção das Bases de Dados Especializadas para a Fase de Final de Jogo

As BDs especializadas utilizadas aqui foram obtidas a partir dos 4 *clusters* gerados do agrupamento da OCA2.0, os quais contém apenas estados de tabuleiros que ocorreram na fase de final de jogo (tabuleiros com 14 peças). Para que fossem obtidos todos os estados de tabuleiros que ocorreram no jogo, subsequentes à identificação desta fase, os jogos da OCA2.0 foram reexecutados considerando apenas os tabuleiros de final de jogo, de modo que foram extraídos apenas os estados que continham 14 peças ou menos. Do mesmo jeito que na seção anterior, os estados armazenados nas BDs correspondem àqueles apresentados ao jogador *vencedor* para a tomada de decisão. Finalizado o processo de geração das 4 BDs especializadas, uma para cada *cluster*, as mesmas devem conter todos os tabuleiros que ocorreram nos jogos a partir do momento que a fase final foi atingida até que o jogo finalizasse. Todos os estados de tabuleiros de cada BD foram convertidos para a representação *NetFeatureMap*. Tais BDs contém respectivamente 417, 634, 863 e 717 estados de tabuleiros, totalizando 2.631 estados que representam essa fase.

4.2.2 Mineração dos Padrões Frequentes e Seleção dos Melhores Conjuntos de Características para cada Perfil de Final de Jogo

Para selecionar os melhores conjuntos de características para representar cada *cluster* de final de jogo, foram feitas análises estatísticas em cada uma das 4 BDs especializadas (que representam tais *clusters*), de modo que os conjuntos cuja frequência relativa fossem maior que 10% foram selecionados, mantendo o suporte mínimo s em $0,1$. Relembrando, como o mapeamento C que converte os estados de tabuleiros para a representação *NetFeatureMap* é do tipo "muitos pra um", a frequência de ocorrência de um determinado conjunto numa determinada BD não corresponde a frequência de ocorrência um determinado estado de tabuleiro.

Foram selecionados 3 conjuntos de características distintos para cada BD, os quais foram considerados como os possíveis melhores conjuntos para representar os ambientes sobre os quais os EGAs atuam. Tais conjuntos quando comparados entre as BDs, correspondem a exatamente aos mesmos conjuntos, ou seja, os conjuntos obtidos para uma BD são os mesmos obtidos para todas as outras. As características que os compõem são:

- Conjunto 1: *CentreControl, XcentreControl, TotalMobility, Exposure, DoubleDiagonal, DiagonalMoment, Threat e Taken*;
- Conjunto 2: *XcentreControl, TotalMobility, Exposure, DoubleDiagonal, DiagonalMoment, Threat e Taken*;
- Conjunto 3: *XcentreControl, TotalMobility, Exposure, DoubleDiagonal, DiagonalMoment, Threat, Taken e PieceAdvantage*;

Para avaliar a qualidade desses 3 conjuntos, foram criadas novas versões do MP-Draughts cujos EGAs foram treinados a partir dos estados de tabuleiros contidos nos *clusters* obtidos da OCA2.0, cada um treinado em um dos 3 conjuntos de características identificados como frequentes em cada *cluster*. Como foram obtidos 4 *clusters* e 3 conjuntos de características para cada um, foram treinadas 12 versões de EGAs, sendo 3 para cada *cluster* (um por conjunto de características). O ganho obtido da seleção automática das características foi medido considerando a performance de jogo desses agentes na fase de final de jogo. Para facilitar o entendimento e contextualizar a criação de cada versão dos EGAs, as mesmas serão explicadas dentro de cada cenário de avaliação em que foram criadas, na próxima seção.

4.2.3 Resultados Experimentais

Esta seção avalia a eficiência da seleção automática das características adequadas para representar os ambientes que ocorrem na fase de final de jogo em 2 cenários de testes

distintos. Esses cenários avaliam o ganho de performance dos EGAs proporcionado pela seleção automática das características e também a performance geral do MP-Draughts em jogos contra os jogadores VisionDraughts e o LS-VisionDraughts. Para tanto, o primeiro cenário avalia dentre os 3 possíveis melhores conjuntos de características para representar um *cluster*, qual de fato é o melhor. Por fim, o segundo cenário avalia se o melhor conjunto de cada perfil quando acoplados a arquitetura do MP-Draughts de fato contribuem para sua performance em jogos contra outros oponentes.

Os resultados obtidos nessa seção, em conjunto com os obtidos na Seção 4.1.3, confirmam a hipótese de que é possível definir quais são as características mais relevantes que um jogador automático de Damas deve considerar durante sua atuação no jogo e ainda, que é possível defini-las de acordo com fase do jogo em que o jogador atua. Além disso, tais resultados mostram que tal definição contribui para o ganho de desempenho do jogador, um vez que especializa sua visão sobre o ambiente.

4.2.4 Cenário 1: definindo o melhor conjunto de características para representar cada perfil de final de jogo

Para avaliar qual dos 3 conjuntos de características obtidos para cada *cluster* é o mais adequado para representá-lo, foi adotada a seguinte estratégia: para cada *cluster* foram treinadas 3 versões do EGA, uma para cada conjunto de característica, sendo que esses foram treinados nos mesmos estados de tabuleiros contidos no *cluster* e sobre as mesmas condições de treinamento, de modo que a única diferença entre eles está no conjunto de características utilizadas no treinamento. Cada EGA foi treinado por 4 ciclos de 10 jogos em cada estado de tabuleiro do *cluster*, com profundidade máxima de busca do Alfa-Beta igual a 8. Seguindo essa estratégia, foram geradas 3 versões de EGAs para um mesmo *cluster*, de modo que foram efetuados torneios competitivos entre elas para definir qual é a melhor para representar o *cluster*. Ou seja, qual possui o conjunto de características mais adequado para representar os conhecimentos contidos no referido *cluster*. Os estados de tabuleiros utilizados nos jogos de teste foram obtidos da BD do Tinsley, os quais são desconhecidos pelos agentes.

Os resultados obtidos de cada torneio estão ilustrados nas Tabelas 11, 12, 13, 14. A nomenclatura utilizada para cada EGA de cada *cluster* é a seguinte: EGANúmerodo*Cluster* - número do conjunto. Por exemplo, o EGA1-1, corresponde ao EGA representante do *cluster* 1, treinado no conjunto de características 1.

A Tabela 11 apresenta o resultado de 30 jogos entre as 3 versões dos EGAs que representam o *cluster* 1. Os resultados ilustram a quantidade e a porcentagem de vitórias, empates e derrotas para o primeiro jogador listado na primeira coluna. Nos jogos efetuados entre o $EGA1-1 \times EGA1-2$, o EGA1-1 obteve 20% de vitórias contra 7% do EGA1-2, ou seja, o EGA1-1 foi 13% mais eficiente. Seguindo a mesma analogia para entendimento

dos demais jogos no *cluster* 1, é possível observar que o EGA1-3 foi 40% e 20% mais eficiente que o EGA1-1 e o EGA 1-2, respectivamente. Isso quer dizer que o conjunto de características vinculados ao EGA1-3 foi mais eficiente em representar os estados de tabuleiros de final de jogo do *cluster* 1, uma vez que o EGA que visualizou os ambientes de jogo sob a perspectiva representada por este conjunto obteve o melhor desempenho de jogo.

Tabela 11 – Resultado dos jogos entre as 3 versões dos EGAs que representam o *cluster* 1.

Jogos entre os EGAs do *cluster* 1						
	Vitórias		Empates		Derrotas	
EGA1-1 $\times$ $EGA1-2$	6	**20%**	22	73%	2	7%
$EGA1-1 \times$ **EGA1-3**	2	7%	14	46%	14	**47%**
$EGA1-2 \times$ **EGA1-3**	4	13%	17	57%	9	**30%**

A Tabela 12 apresenta o resultado de 30 jogos entre as 3 versões dos EGAs que representam o *cluster* 2. Como ilustrado, o EGA2-3 foi o que obteve melhor desempenho de jogo, sendo 35% e 17% mais eficiente que o EGA2-1 e o EGA2-2, respectivamente. Do mesmo que no *cluster* 1, no *cluster* 2 o conjunto de características vinculados ao EGA2-3 foi o mais eficiente em representar os estados de tabuleiros de final de jogo.

Tabela 12 – Resultado dos jogos entre as 3 versões dos EGAs que representam o *cluster* 2.

Jogos entre os EGAs do *cluster* 2						
	Vitórias		Empates		Derrotas	
$EGA2-1 \times EGA2-2$	6	20%	18	60%	6	20%
$EGA2-1 \times$ **EGA2-3**	4	13%	12	39%	**15**	**48%**
$EGA2-2 \times$ **EGA2-3**	7	23%	11	37%	**12**	**40%**

A Tabela 13 ilustra o resultado dos jogos entre as 3 versões dos EGAs que representam o *cluster* 3. Observe que o EGA3-3 foi o que obteve melhor desempenho de jogo, sendo 30% e 71% mais eficiente que o EGA3-1 e o EGA3-2, respectivamente. Novamente, o conjunto-3 de características, vinculado ao EGA3-3, foi o mais eficiente em representar os estados de tabuleiros de final de jogo contidos nesse *cluster*.

Tabela 13 – Resultado dos jogos entre as 3 versões dos EGAs que representam o *cluster* 3.

Jogos entre os EGAs do *cluster* 3						
	Vitórias		Empates		Derrotas	
EGA3-1 $\times$ $EGA3-2$	**20**	**67%**	9	30%	1	3%
$EGA3-1 \times$ **EGA3-3**	3	10%	15	50%	**12**	**40%**
$EGA3-2 \times$ **EGA3-3**	1	3%	7	23%	**22**	**74%**

Por fim, a Tabela 14 apresenta o resultado dos jogos entre as 3 versões dos EGAs que representam o *cluster* 4. Do mesmo modo que nos outros *clusters*, o conjunto de características vinculado ao EGA4-3 também foi o mais eficiente em representar os estados de tabuleiros de final de jogo, o qual foi 34% e 14% mais eficiente que o EGA 4-1 e o EGA4-2, respectivamente.

Tabela 14 – Resultado dos jogos entre as 3 versões dos EGAs que representam o *cluster* 4.

Jogos entre os EGAs do *cluster* 4						
	Vitórias		Empates		Derrotas	
$EGA4-1 \times$ **EGA4-2**	2	7%	17	56%	**11**	**37%**
$EGA4-1 \times$ **EGA4-3**	1	3%	18	60%	**11**	**37%**
$EGA4-2 \times$ **EGA4-3**	4	13%	18	60%	**8**	**27%**

Resumindo, o conjunto 3 foi o mais eficiente em representar a fase de final de jogo em todos os perfis desta fase representados na arquitetura do MP-Draughts. Tal conjunto é composto por 8 características, que são: *XcentreControl, TotalMobility, Exposure, DoubleDiagonal, DiagonalMoment, Threat, Taken e PieceAdvantage*. Comparando com o melhor conjunto obtido para representar o ambiente do jogo de Damas durante todo o jogo, o V_6 (definido na Seção 4.1.6), apenas a característica *Threat* foi adicionada, a qual representa as possibilidades de movimentos que o agente pode executar e assim, ameaçar o oponente. Essa característica mostra ao agente as possibilidades de ações ofensivas em direção ao final do tabuleiro, o que aumenta a possibilidade de formação de Damas e a redução de força (peças) do oponente. Sem dúvidas, essa é também uma característica importante que deve ser considerada na fase final do jogo. Contudo, as características adequadas para representar a fase de final de jogo aqui identificadas não inclui nenhuma das relacionadas a presença de damas no tabuleiro, o que faz sentido porque a abordagem utilizada filtra as mesmas com base na frequência em que ocorrem no jogo. Peças do tipo Damas não ocorrem com tanta frequência no jogo quando comparada com as peças simples, porém quando ocorrem são muito importantes. Desse modo, uma deficiência desta abordagem é não identificar características pouco frequentes porém não menos importantes para o jogo.

Diante das características consideradas adequadas para representar os diferentes ambiente de atuação do MP-Draughts, este jogador utilizará o conjunto V_6 para representar seus ambientes de atuação desde o início até a fase de final de jogo e, desse ponto em diante, incluirá a característica *Threat* na representação.

4.2.5 Cenário 2: avaliando a performance da nova versão do MP-Draughts contra outros agentes não supervisionados

Este cenário avalia a performance da melhor versão do MP-Draughts para a fase de final de final de jogo em competições contra os jogadores VisionDraughts Original treinado a partir das 15 características, a melhor versão do VisionDraughts (VisionDraughts-V_6) cujas características foram automaticamente selecionadas e o LS-VisionDraughts. Foram executados 30 jogos competitivos entre o MP-Draughts e cada um desses jogadores, cujos estados de tabuleiros também foram extraídos da BD de jogos do Tinsley. A Tabela 15 ilustra o resultado desses jogos. Os resultados mostram as vitórias, empates e derrotas para o MP-Draughts.

Nos jogos competitivos do MP-Draughts contra o VisionDraughts Original, o multiagente obteve 57% de vitórias contra apenas 3% do seu oponente. Este resultado reforça que a seleção automática de características é melhor que a manual. Já nos jogos contra o VisionDraughts-V_6, cujas características foram selecionadas para representar todo o jogo, a superioridade de jogo obtida pelo MP-Draughts que foi de 17% a mais de vitórias, comprova que a seleção automática das características para representar a fase de final de jogo aos EGAs foi uma boa solução. Nos jogos contra o LS-VisionDraughts, o MP-Draughts obteve 10% a mais de vitórias, o que mostra que este jogador se configura o melhor entre jogadores automáticos não supervisionados aqui avaliados. Finalmente, estes resultados confirmam que, dentre os jogadores automáticos de Damas que utilizam a *NetFeatureMap* para representar o ambiente de atuação dos jogadores, o MP-Draughts é o que possui os melhores conjuntos de características para esta representação. Tais conjuntos foram obtidos de forma simples e eficiente utilizando a técnica de mineração de padrões frequentes de BDs.

Tabela 15 – Resultado dos jogos competitivos entre o MP-Draughts (composto pelos melhores EGAs identificados no cenário 1) contra os jogadores VisionDraughts Original, VisionDraughts-V_6 e LS-VisionDraughts.

Resultados dos jogos						
	Vitórias		Empates		Derrotas	
$MP-Draughts \times VisionOriginal$	**17**	**57%**	12	40%	1	3%
$MP-Draughts \times VisionDraughts-V_6$	**6**	**20%**	23	77%	1	3%
$MP-Draughts \times LS-VisionDraughts$	**6**	**20%**	21	70%	3	10%

4.3 Considerações Finais

Este capítulo apresentou uma nova abordagem que seleciona automaticamente conjuntos de características suficientemente adequadas para representar os ambientes de atuação de agentes inteligentes, a qual é baseada na mineração dos padrões que frequentemente

ocorrem no ambiente enquanto esses agentes atuam. Tal abordagem foi utilizada para selecionar as melhores características da *NetFeatureMap* para representar o domínio do jogo de Damas. Essas características puderam ser identificadas a partir de análises estatísticas, baseadas nas suas frequências relativas, realizadas em BDs que continham vários ambientes (estados de tabuleiros) sobre os quais grandes mestres do jogo atuaram. Os resultados obtidos pela abordagem proposta foram comparados com os resultados obtidos por um AG no mesmo domínio e se mostraram muito mais eficientes na obtenção dos conjuntos, além de ser mais viável computacionalmente.

Devido à agilidade computacional desta técnica, foi possível identificar diferentes conjuntos de características adequados para representar o jogo de Damas em diversas situações, que vão desde a mesma representação para todo o jogo até representações distintas para fases específicas do jogo (começo/meio e fim de jogo). Tal feito era considerado impraticável até então, pois o tempo gasto pelo AG para encontrar uma única solução (conjunto de características) para o jogo todo foi de aproximadamente 4 meses, de modo que ficaria difícil prever quanto tempo seria necessário para encontrar mais de uma solução para o jogo todo. Mais difícil ainda seria prever o tempo gasto pelo AG para encontrar soluções para as diversas fases do jogo.

A principal contribuição deste capítulo foi propor uma técnica simples e eficiente que seja capaz de selecionar características adequadas para representar qualquer ambiente de atuação de agentes inteligentes, desde que existam várias amostras desses ambientes. Além disso, tal técnica também é aplicável a ambientes com alta complexidade de estados, como foi o caso do jogo de Damas. Apesar da boa contribuição obtida desta técnica, a mesma não foi capaz de identificar características peculiares e não menos importantes para o ambiente, o que ocorreu devido a essência da técnica que é encontrar padrões que frequentemente ocorrem no ambiente.

Capítulo 5

Aprimorando o Processo de Alocação de Agentes em Sistemas Multiagentes utilizando Regras de Exceção

Um dos requisitos fundamentais para que um sistema multiagente atinja seus objetivos é que os agentes que o compõem apresentem habilidades específicas e complementares. Para tanto, esses agentes devem contar com conhecimentos distintos sobre o ambiente em que atuam, de modo a estarem aptos a agir como especialistas nas situações para as quais foram treinados. Consequentemente, durante a atuação do sistema multiagente, torna-se crucial a tarefa de alocar adequadamente o(s) agente(s) responsável(is) pelas tomadas de decisão com base no estado corrente do ambiente. Obviamente, uma alocação inadequada desses agentes prejudicaria o desempenho global do sistema e tenderia a afastá-lo de seus objetivos. O risco de ocorrência de tal inconveniente aumenta significativamente quando o sistema multiagente atua em ambientes com elevado espaço de estados. Considerando tais argumentos, o presente capítulo propõe um método de alocação de agentes que combina RNs de agrupamento com regras de exceção, as quais em conjunto definem os agentes adequados para atuarem em cada situação do ambiente. Em tal método, cada vez que o multiagente precisa definir um agente para atuar no ambiente, a RN é responsável por abstrair as informações relativas ao estado corrente do ambiente e, com base nessas informações, indicar o agente cujas as habilidades sejam as mais adequadas para a situação. Por outro lado, as regras de exceção são responsáveis por refinar a indicação da RN em situações excepcionais em que esta se mostra pouco apta a fazê-la.

Assim como nos capítulos anteriores, o multiagente MP-Draughts foi utilizado como ambiente de desenvolvimento do método aqui proposto. Conforme apresentado no Capítulo 3, o MP-Draughts utiliza uma RN adaptativa para efetuar as alocações dos EGAs que devem atuar nas distintas situações que ocorrem na fase de final de jogo. Tal RN é a mesma utilizada para obter os *clusters* que representam esta fase do jogo, lembrando que os tabuleiros desses *clusters* foram usados como base de treinamento para os respectivos

EGAs. Durante tais alocações, a RN atua como uma rede de classificação, indicando o *cluster* que melhor representa a situação corrente do jogo. No caso, o EGA que representa o *cluster* indicado é o mais adequado para a tomada de decisão naquela situação. Os mesmos conhecimentos adquiridos pela RN durante o processo de agrupamento são utilizados por ela para alocar (classificar) os EGAs que devem atuar nas distintas situações de final de jogo. Todavia, a partir de análises feitas sobre o comportamento dos EGAs na fase de final de jogo, foi possível observar que em algumas situações excepcionais o EGA indicado pela RN para atuar no jogo não desempenhava adequadamente seu papel, conduzindo o jogo para uma derrota. Foi observado também que outros EGAs, diferente daqueles definido pela RN, eram capazes de obter melhores resultados (vitórias) nessas mesmas situações. Tal observação permitiu concluir que o baixo desempenho dos EGAs não estava relacionado à falta de habilidade dos mesmos e sim, a falta de conhecimento da RN para adequadamente alocar o melhor EGA para atuar nessas situações. Como tais situações representam exceções aos casos gerais e, na maioria das vezes, a RN é adequada para efetuar as alocações, é conveniente propor alguma alternativa de alocação que trate apenas as situações excepcionais em que a RN se mostrar inadequada, complementando assim sua capacidade de alocação.

As regras de exceção são úteis em situações em que o conhecimento geral abstraído de um contexto não é suficiente para representar todo o conhecimento nele contido. Uma exceção pode ser definida como algo diferente da maioria, que contradiz o senso comum e geralmente representa algum conhecimento interessante [72]. Enquanto o conhecimento geral representa o senso comum de um determinado contexto, as regras de exceção representam conhecimentos locais que contradizem esse senso comum. O conceito de localidade das regras de exceção está relacionado ao fato delas serem obtidas a partir de casos especiais que o conhecimento geral não foi capaz de representar, sendo usadas para complementar esse conhecimento. Isso significa dizer que os conhecimentos gerais são obtidos primeiro e as exceções a esses conhecimentos são obtidas como um posterior refinamento dos mesmos. De volta ao problema de alocação do multiagente MP-Draughts em que a RN não foi capaz de abstrair o conhecimento necessário para alocar adequadamente os EGAs em todas as situações de final de jogo, as regras de exceção podem ser usadas para representar o conhecimento embutido nessas situações, de modo a refinar o conhecimento da RN. Desse modo, a combinação entre a RN para representar o conhecimento geral e as regras de exceção para representar conhecimentos locais se mostra uma alternativa interessante e conveniente para tratar o problema de alocação de agentes identificado nesse sistema.

No intuito de facilitar o entendimento do leitor sobre como o método aqui proposto pode ser utilizado pelo multiagente para efetuar a alocação dos EGAs e de como as regras de exceções podem ser obtidas de modo a complementar o conhecimento da RN, as próximas seções deste capítulo estão organizadas da seguinte maneira: a Seção 5.1

apresenta a nova arquitetura do multiagente a qual foi acoplado o módulo de regras de exceção para atuar conjuntamente com a RN adaptativa na tarefa de alocação dos EGAs; a Seção 5.2 explica o método proposto, o qual combina RN e regras de exceção para otimizar o processo de alocação de agentes do MP-Draughts. Finalmente, a Seção 5.3 apresenta os resultados obtidos pelo método quando acoplado a arquitetura do MP-Draughts. Tais resultados completam o objetivo geral perseguido no presente trabalho.

5.1 Nova Arquitetura do MP-Draughts

A nova arquitetura do MP-Draughts, apresentada na Figura 19, é composta por 5 módulos, sendo que o módulo *Regras de Exceção* representa a novidade desta arquitetura:

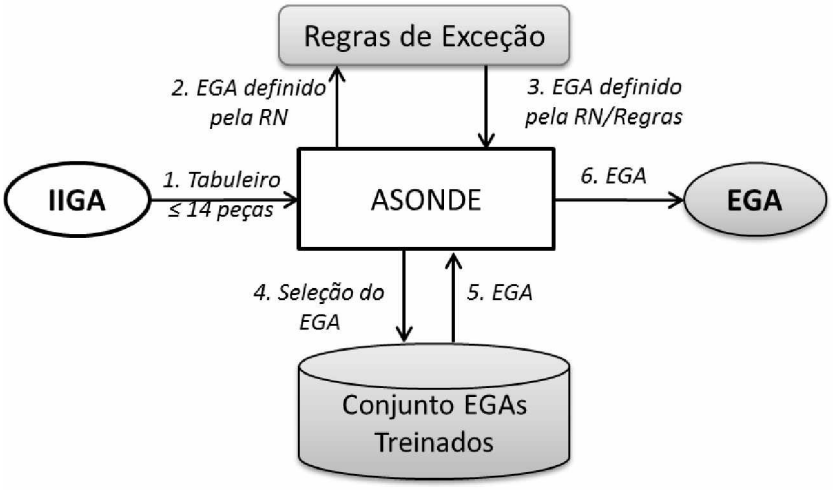

Figura 19 – Nova proposta de Arquitetura do MP-Draughts: integração entre a RN ASONDE e o conjunto de regras de exceção na tarefa de alocação dos EGAs.

- **IIGA - *Initial/ Intermediate Game Agent***: este módulo representa o agente especialista que atua nas fases iniciais e intermediárias do jogo de Damas;
- **Conjunto de EGAs Treinados - *EndGame Agents***: representa o conjunto dos agentes especialistas nas fases de final de jogo. Cada agente deste módulo é treinado para se tornar especialista em perfis de tabuleiros de final de jogo similares aos contidos no *cluster* em que o mesmo foi treinado;
- **ASONDE**: a RN adaptativa que compõe este módulo possui duas responsabilidades distintas na arquitetura do multiagente: primeiro, ela é responsável por minerar uma BD de estados de tabuleiros de final de jogo e obter os *clusters* de treinamento dos EGAs; segundo, ela é responsável por, dado o tabuleiro corrente do jogo, indicar dentre os EGAs treinados, aquele que está mais apto a executar a tomada de decisão;
- **Regras de Exceção**: módulo responsável por, dada a indicação da RN e o tabuleiro corrente do jogo, verificar se existe alguma restrição quanto à citada indicação. Caso

exista, o módulo deve substituir a indicação da RN pela indicação das regras de exceção. O fato de existir alguma restrição quanto à indicação da RN significa que o estado corrente do jogo representa uma situação excepcional em que a RN não foi capaz de indicar o EGA mais adequado;

- **EGA**: corresponde ao agente de final de jogo escolhido, ora pelo módulo da RN adaptativa, ora pelo módulo das regras de exceção para atuar na fase de final de jogo. Esse agente, por ter sido treinado num *cluster* cujos tabuleiros se assemelham ao estado corrente do jogo, é o que tem maiores habilidades para atuar neste cenário.

Diante da nova arquitetura apresentada, a dinâmica de atuação do MP-Draughts em jogos contra outros jogadores deve ser a seguinte: o IIGA é o agente responsável por atuar no jogo desde o início (tabuleiro 8×8 completo) até que a fase de final de jogo seja alcançada. Neste momento, o tabuleiro corrente do jogo é apresentado à RN adaptativa, que abstrai as informações contidas no mesmo e, diante dessas informações, indica o EGA mais adequado para atuar no jogo. A partir da indicação da RN, o módulo de regras de exceção é acionado para verificar se existe alguma restrição à indicação da RN. Caso exista, a indicação da RN é substituída pela indicação das regras de exceção. Caso contrário, prevalece a indicação da RN. Definido o EGA mais adequado para atuar na fase de final de jogo (ou pela RN ou pelas regras), este assume o jogo e o conduz até o final.

Na dinâmica descrita acima, as regras de exceção atuam após a atuação da RN, refinando, quando necessário, a indicação da mesma. Tal fato conserva o conceito de localidade das regras de exceção.

5.2 Combinando Redes Neurais e Regras de Exceção para Tarefas de Alocação de Agentes

Vários métodos foram propostos na literatura para extrair exceções de situações especiais que ocorrem em contextos onde técnicas tradicionais de regras de classificação são aplicadas [71], [73], [72], [76], [74], [75], [98], [77]. Tais situações especiais correspondem a exemplos de dados que não foram corretamente cobertos pelas regras de classificação, os quais podem ser cobertos por regras de exceção. Entretanto, nenhum desses métodos propõe tratar situações especiais que ocorrem em contextos que utilizam RN de agrupamento. Uma vez que essa RN teve seus pesos ajustados para representar os *clusters* por ela obtidos, após concluído seu treinamento ela também pode ser utilizada para classificar novos dados a ela apresentados em tempo de execução. A estratégia usada pelo método aqui proposto para melhorar a capacidade de alocação da RN pode ser resumido da seguinte forma: inicialmente, devem ser identificadas as situações especiais em que a RN não aloca adequadamente o EGA para atuar na fase de final de jogo, as quais devem ser

separadas em BDs distintas que representem as alocações inadequadas para cada EGA. Na sequência, para cada BD devem ser extraídas as regras de exceção capazes de alocar adequadamente os EGAs nessas situações. Finalmente, as regras de exceção devem ser combinadas com a RN para, juntas, desempenharem a tarefa de alocação. Os detalhes executados em cada etapa deste processo, são explicados nas próximas subseções.

5.2.1 Identificação e Obtenção das Bases de Dados de Situações Especiais

No intuito de identificar as situações especiais em que a RN não é capaz de alocar adequadamente os EGAs, foram executados vários jogos entre todos EGAs do MP-Draughts. A versão do MP-Draughts utilizada nesses jogos foi a melhor versão obtida no Capítulo 4 para a fase de final de jogo, ou seja, a versão que possui 4 EGAs treinados nos estados de tabuleiros contidos nos *clusters* obtidos pela ASONDE da BD OCA2.0. Tais estados de tabuleiros foram representados utilizando as 8 melhores características para a fase de final de jogo identificadas na Seção 4.2.4.

Os jogos para a obtenção das BDs de exceção foram executados a partir dos estados de tabuleiro de final de jogo contidos numa segunda versão da BD OCA 2.0, a qual contém 2.400 tabuleiros de final de jogo. Tal versão contém várias situações de tabuleiros de final de jogo (jogos finalizados em vitória, empate e derrota) e não apenas tabuleiros de vitória como os contidos na BD OCA utilizada no Capítulo 4. A reutilização desta BD é justificável devido à grande quantidade de jogos que ela contém, sendo que nenhuma outra BD de jogos de Damas possui tantos jogos. Além disso, os jogos dessa BD possuem as mais variadas situações de final de jogo, visto que foram jogados por diferentes especialistas do domínio, ao longo de várias décadas.

A expressão "situações de final de jogo" corresponde a um tabuleiro de final de jogo, do mesmo jeito que a expressão "situações especiais" corresponde aos estados de tabuleiros de final de jogo em que a ASONDE não foi capaz de alocar o melhor EGA.

O processo de identificação e obtenção das BDs de situações especiais relacionadas ao baixo desempenho da ASONDE pode ser observado na Figura 20. A parte (a) da Figura 20 apresenta os passos executados para a identificação desses casos: para cada estado de tabuleiro de final de jogo contido na BD OCA 2.0, o mesmo foi traduzido para a representação *NetFeatureMap* e apresentado a ASONDE, a qual abstraiu as informações desse tabuleiro e indicou o EGA mais adequado para a situação. Depois disso, foi executada uma série 2 jogos (peças vermelhas e peças pretas) entre o EGA indicado pela ASONDE (EGA-ASONDE) e cada um dos demais EGAs. Como o MP-Draughts possui 4 EGAs, foram executadas 3 séries de 2 jogos. Em cada série, caso o EGA-ASONDE perdesse pelo menos 1 jogo para o EGA oponente, o tabuleiro era considerado um caso especial e era, então, armazenado numa BD auxiliar. Além dele, também eram armazenados o número

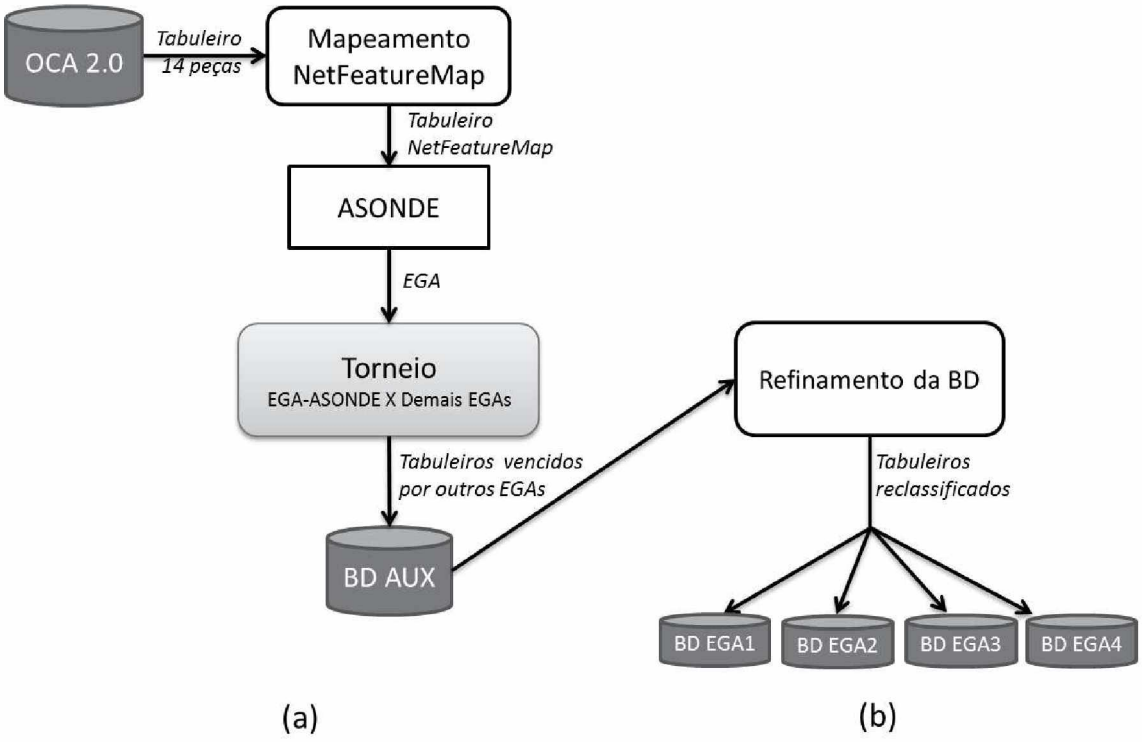

Figura 20 – Processo de identificação e obtenção das BDs de situações especiais.

do EGA que obteve o pior desempenho (no caso, o EGA-ASONDE) e o número do EGA que obteve o melhor desempenho (no caso, o oponente), conforme ilustrado na Figura 21. Note que ao final das 3 séries de jogos no tabuleiro, o mesmo poderia ter sido armazenado na BD auxiliar até 3 vezes, caso o EGA-ASONDE tivesse obtido pior desempenho nas 3 séries. Porém, cada vez que o tabuleiro foi armazenado na BD, o número do EGA que venceu o jogo (campo 3 da Figura 21) era diferente. Finalizadas as 3 séries de jogos no tabuleiro, um novo estado de tabuleiro era recuperado da BD OCA 2.0 e o ciclo de jogos se reiniciava. Depois que todos os tabuleiros da BD OCA 2.0 tivessem sido jogados por um EGA indicado pela ASONDE contra os demais EGAs, a BD auxiliar conteria todas as situações em que a ASONDE se mostrou inadequada na alocação dos EGAs. Além disso, a BD também conteria a informação de qual é o EGA inadequado e o EGA adequado para cada situação (campos 2 e 3 da Figura 21). Foram identificadas 300 situações especiais, o que corresponde a aproximadamente 12,5% dos tabuleiros contidos na OCA 2.0.

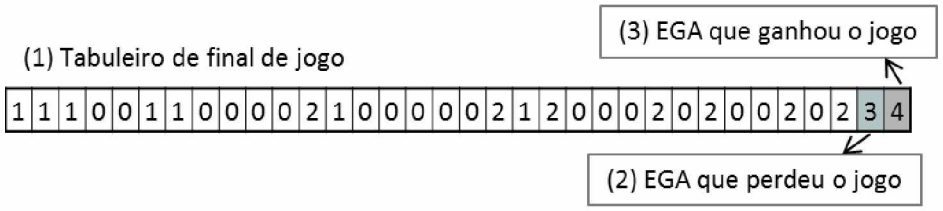

Figura 21 – Exemplo de um tabuleiro de final de jogo armazenado na BD auxiliar.

Uma vez que as situações especiais tenham sido identificadas, o próximo passo foi

separá-las em 4 BDs distintas. Esta separação foi feita para que cada BD contivesse apenas as situações de alocação inadequada feitas pela RN para um determinado EGA. Desta forma, para cada BD foi obtido um conjunto de regras de exceção distinto, as quais serão verificadas em momentos distintos, de acordo com a indicação da RN. Isso permite que, cada vez que a RN indicar um EGA, apenas as regras obtidas a partir da BD que representa tal EGA sejam verificadas, o que garante o conceito de localidade da regra de exceção.

O processo de obtenção dessas BDs (ilustrado na parte (b) da Figura 20) pode ser resumido da seguinte forma: inicialmente, os tabuleiros foram separados em 4 grupos distintos, de modo que cada grupo contivesse apenas os tabuleiros correspondentes às situações de alocação inadequada da RN para um determinado EGA. Isso foi feito a partir da informação contida no campo (2) da Figura 21. Na sequência, cada grupo foi refinado de modo a eliminar as eventuais repetições de um mesmo tabuleiro neste grupo. Uma repetição ocorre sempre que o EGA alocado pela RN para esse tabuleiro tiver sido derrotado por mais de um EGA oponente. Dentre os tabuleiros repetidos, permaneceu no grupo unicamente aquele cujo campo 3 (correspondente ao EGA vencedor do jogo, conforme Figura 21) estivesse preenchido com o valor que apareceu com maior frequência no mesmo campo 3 dos demais tabuleiros do grupo. Dessa forma, o refinamento privilegiou o EGA que obteve melhor desempenho no grupo. Por último, a título de performance, o campo (2), referente ao EGA que perdeu o jogo, de cada tabuleiro foi retirado. Tal campo pôde ser suprimido porque o agrupamento dos tabuleiros (representado por cada BD), realizado no início deste processo, já contempla esta informação.

Resumindo, cada BD contém as situações alocadas inadequadamente pela RN para um determinado EGA (representado pelo número da BD), reclassificadas para o EGA mais adequado. A reclassificação foi feita com base no desempenho dos EGAs oponentes nessas situações.

5.2.2 Obtenção das Regras de Exceção

Cada uma das 4 BDs obtidas na seção anterior contém as situações especiais de jogo em que a RN não indicou corretamente um determinado EGA. Desse modo, cada BD deve ser tratada separadamente no processo de obtenção de regras de exceção, garantindo que as regras obtidas a partir de cada uma estejam aptas a serem utilizadas para refinar a indicação da RN que recaia sobre o EGA que a BD representa, sempre que necessário.

O processo de obtenção das regras de exceção para cada uma das BDs foi o seguinte: as classes as quais as regras podem pertencer foram representadas no campo 3 da Figura 21, as regras de exceção foram extraídas da BD utilizando o algoritmo de mineração de árvores de decisão C4.5 [79], considerando o fator de confiança de 0,5 e validação cruzada com 10 partições. A versão do algoritmo utilizada foi a J48 implementada na plataforma Weka 3.6 [99]. Para verificar a possível ocorrência de um superajustamento das regras

de exceção, o algoritmo foi re-executado seguidas vezes, variando os valores do fator de confiança no intervalo [0,2, 0,8], assim como a quantidade de partições da validação cruzada no intervalo [3, 10]. Também foram testadas algumas variações de amostragem da BD de treinamento e teste. Os valores fixados para a execução do algoritmo foram os que obtiveram as regras com melhores capacidade de cobertura dos dados da BD (no caso, das 4 BDs). A escolha do C4.5 para gerar as regras de exceção foi tomada devido aos seguintes fatores: o algoritmo lida bem com problemas cujos valores dos atributos podem variar bastante, o que ocorre na representação dos tabuleiros utilizando as características quantificadas; a facilidade de interpretação das regras obtidas para as BDs do problema; e o fato de ser um dos algoritmos mais utilizado na literatura, o qual geralmente obtém ótimos resultados para os problemas de classificação em que é aplicado. O algoritmo PART [100] também foi utilizado durante as primeiras etapas de geração das regras, porém as regras obtidas por este algoritmo, além de não possuírem os melhores valores de cobertura, eram significativamente maiores e menos legíveis que as obtidas pelo C4.5.

Todas as regras obtidas para as 4 BDs cuja confiança foi superior a 50% foram consideradas para uma avaliação preliminar. A decisão de avaliar todas as regras com confiança superior a 50% foi baseada nos resultados apresentados nos trabalhos de Prati [98], o qual sugere que regras com confiança acima de 50% podem ser boas. Tal avaliação verificou a capacidade de classificação dessas regras (cobertura das regras) quando aplicadas à BD auxiliar que contém todas as situações especiais identificadas. O intuito dessa avaliação foi garantir que, considerando todas as situações especiais de jogo conhecidas pelas regras, elas seriam capazes de classificá-las apropriadamente. Porém, nem todas foram eficientes, sendo que várias classificaram mais exemplos incorreta do que corretamente. Novamente, apenas as regras cuja confiança de classificação da BD auxiliar permaneceu superior a 50% foram consideradas capazes de representar as situações de exceção. Sendo assim, para cada uma das 4 BDs foram consideradas duas regras de exceção. Tais regras foram então avaliadas no domínio do jogo de Damas, atuando como exceção à alocação da ASONDE. Os resultados obtidos dessas avaliações são apresentados na seção de resultados.

5.2.3 Combinação das Regras de Exceção com a Rede Neural para Alocação de Agentes

Esta subseção apresenta a dinâmica de alocação do EGA apropriado no sistema MP-Draughts. Cada vez que o multiagente precisa definir um EGA para atuar no ambiente, a RN é quem o define, visto que é ela quem tem o conhecimento geral do problema. Na sequência, as regras de exceção validam a escolha da RN. As regras de exceção utilizadas para validar a escolha da RN são aquelas obtidas a partir da BD correspondente ao EGA indicado pela RN. Por exemplo, se a RN indicou o EGA-1, as regras obtidas da BD-1

é que validam a escolha da RN. Caso haja alguma exceção à indicação da RN, fato que aciona alguma regra, o EGA indicado pela regra acionada é o que, de fato, deve atuar no ambiente (tomar a decisão). Caso não haja exceção (nenhuma regra acionada) prevalece a indicação da RN. Note que, quando nenhuma regra é acionada, isso significa que a situação é favorável à RN ou, então, que é uma situação nova, tanto para a RN, quanto para as regras. Neste último caso, é mais conveniente considerar a indicação da RN, visto que esta possui um conhecimento mais generalizado sobre o ambiente e tem maiores condições de efetuar uma escolha mais assertiva.

Analisando o método aqui proposto de uma maneira mais genérica, é possível observar que ele pode ser aplicado a outros problemas de classificação de dados envolvendo RN de agrupamento. Desde que nesses problemas seja possível identificar, além das situações mal classificadas pela RN, qual seria a melhor classificação para essas situações.

5.3 Resultados Experimentais

Esta seção avalia a eficiência do método aqui proposto em 4 cenários de testes distintos. Tais cenários estão divididos entre: validação e avaliação das regras de exceção, apresentados nos cenários 1 e 2; avaliação de desempenho do multiagente em jogos contra outros jogadores automáticos de Damas não supervisionados, cenário 3; e, por fim, avaliação da performance de jogo do multiagente em relação a seus oponentes não supervisionados, quando avaliados sob a perspectiva do jogador supervisionado Cake, cenário 4. Todos os cenários foram avaliados em jogos competitivos realizados na fase de final de jogo, envolvendo as duas versões do MP-Draughts: a que utiliza apenas a RN para alocar o EGA (correspondente à versão original do multiagente (MP-RN)) e a que utiliza a combinação da RN com as regras de exceção para a alocação (que corresponde à nova arquitetura do mesmo (MP-RN-RE)).

Os resultados obtidos nesses cenários comprovam a hipótese de que é possível identificar e tratar localmente as situações de baixo desempenho de uma determinada técnica em alocar os agentes adequados para determinadas situações, utilizando, para tanto, uma técnica que complemente seu conhecimento. Mais especificamente, as situações de baixo desempenho da ASONDE na tarefa de alocação foi tratada localmente e, eficientemente, por regras de exceção.

5.3.1 Cenário 1: validação das regras de exceção

Para validar a eficiência das regras de exceção em situações de baixo desempenho da RN, foram executados jogos competitivos entre as duas versões do MP-Draughts. Os jogos foram executados nos tabuleiros de final de jogo contidos na BD OCA 2.0, a mesma utilizada para obter as situações especiais de final de jogo. Como as situações de exceção ocorridas nesta BD ficou em torno de 12,5% (correspondente a BD auxiliar obtida na

Seção 5.2.1), é esperado que as regras de exceção sejam ativadas nesta mesma proporção durante os jogos. Os resultados dos jogos em que as regras não foram ativadas, não foram considerados para a avaliação deste cenário, uma vez que nesses, os jogos foram executados entre um mesmo EGA. Em outras palavras, como as regras não foram acionadas, os dois EGAs que atuaram nos jogos foram alocados pela RN e como tal RN efetua a alocação considerando o mesmo tabuleiro, sua indicação é a mesma para ambos os agentes.

Os resultados dos jogos em que as regras de exceção foram ativadas, assim como a porcentagem de ativação dessas regras, são apresentados na Tabela 16. As regras R1 e R2 compreendidas nas linhas das BD-1, BD-2, BD-3 e BD-4 representam as regras de exceção à indicação da RN para os EGAs 1, 2, 3 e 4, respectivamente.

Começando pela análise da porcentagem de ativação das regras, as colunas *Total*, *%Regras* e *% BD* mostram, respectivamente: o total de vezes que cada regra foi ativada, a porcentagem de ativação de cada regra em relação a ativação das demais e a porcentagem de ativação de cada regra em relação a todas as situações da BD OCA. Com exceção da R2 da BD-1, cuja ativação foi bem superior, as demais regras foram ativadas relativamente na mesma proporção (colunas Total e %Regras), o que significa que estas não foram superajustadas para a BD. Tal fato é reforçado pela porcentagem de ativação de cada regra quando considerada todas as situações especiais da BD (coluna % BD), cujo somatório (destacado em cinza no final da tabela) se mantém em torno de 12,4%, coerente com a quantidade de situações especiais existentes na BD.

Analisando os resultados dos jogos, as colunas *Vitórias*, *Derrotas* e *Empates* ilustram a quantidade e a porcentagem de vitórias, derrotas e empates obtidos pela nova arquitetura do MP-Draughts contra a arquitetura original. Como pode ser observado, a utilização das regras contribuiu mais positiva do que negativamente para o desempenho da nova arquitetura do multiagente. Em outras palavras, os EGAs alocados pelas regras na situações identificadas como especiais converteram mais jogos para vitória do que para derrota. Em relação aos casos de empates, não se pode dizer nem que a regra contribuiu, nem que prejudicou o desempenho do multiagente. É possível observar que, apesar de a R2 da BD-1 ter sido consideravelmente mais acionada que as demais, essa regra realizou boas alocações.

5.3.2 Cenário 2: avaliação das regras de exceção

Assim como no cenário 1, as regras de exceção foram avaliadas em jogos competitivos envolvendo as duas versões do MP-Draughts (com e sem regras). Para avaliar a eficiência dessas regras foi utilizada a BD de Tinsley, cujos tabuleiros são desconhecidos pelos EGAs, pela RN e pelas regras. Tal BD contém 343 tabuleiros de final de jogo, os quais foram obtidos de maneira análoga ao descrito na Seção 3.4.1. Os resultados dos jogos em que as regras não foram ativadas também não foram considerados neste cenário.

Tabela 16 – Resultado dos jogos de validação entre a nova versão do MP-Draughts (MP-RN-RE) contra sua versão original (MP-RN), nos quais as regras de exceção foram ativadas.

$MP-RN-RE \times MP-RN$										
		Vitórias		Derrotas		Empates		TOTAL	% Regras	% BD
BD-1	R1	3	30%	2	20%	5	50%	10	3%	0,4%
	R2	37	33%	32	28%	44	39%	113	38%	5%
BD-2	R1	5	25%	3	15%	12	60%	20	7%	1%
	R2	6	19%	3	9%	23	72%	32	11%	1%
BD-3	R1	10	67%	2	13%	3	20%	15	5%	1%
	R2	11	32%	6	18%	17	50%	34	11%	1%
BD-4	R1	11	28%	6	15%	22	56%	39	13%	2%
	R2	12	34%	8	23%	15	43%	35	12%	1%
% de ativação das regras em relação a BD OCA 2.0										12,4%

Os resultados obtidos da avaliação das regras são apresentados na Tabela 17, a qual contém os resultados dos jogos em que as regras foram ativadas e a porcentagem de ativação dessas regras. Os campos dessa tabela são os mesmos da tabela apresentada no cenário anterior. No que tange a porcentagem de ativação das regras, essa se manteve em torno de 12% em relação a todas as situações da BD, do mesmo modo que a ativação de cada regra também se manteve equilibrada, exceto para a R2 da BD-1. Tal regra continua sendo mais ativada que as demais, porém ela continua se mantendo como uma boa regra, visto que obteve mais resultados de vitória do que de derrota. Até então, tal regra tem representado uma situação de superajustamento.

Em relação ao desempenho nos jogos, também apresentados na Tabela 17 e cujos resultados são apresentados em relação ao desempenho da nova arquitetura do MP-Draughts, é possível perceber que o refinamento da alocação dos EGAs proporcionado pela ativação das regras foi bastante satisfatório para a maioria dos jogos. No caso menos satisfatório, para a R2 da BD-2 e R2 da BD-4, houve um equilíbrio entre os ganhos e as perdas de performance da nova arquitetura. Nos demais casos, os ganhos superaram em pelo menos 38% as perdas. Novamente, sobre os casos de empates não se pode dizer, nem que as regras contribuíram, nem que prejudicaram o desempenho da nova arquitetura do multiagente.

Para confirmar que as regras de exceção foram uma alternativa viável para tratar situações excepcionais relacionadas a indicação da RN de modo a melhorar a performance de jogo do multiagente, os próximos cenários mostram o desempenho da nova arquitetura do MP-Draughts em jogos contra outros jogadores não supervisionados de Damas. Os mesmos jogos também foram executados entre a versão original do multiagente contra esses oponentes, no intuito de comparar o desempenho de ambas as versões nesses confrontos.

Tabela 17 – Resultado dos jogos de avaliação entre as duas versões do MP-Draughts em que as regras de exceção foram ativadas.

$MP-RN-RE \times MP-RN$										
		Vitórias		Derrotas		Empates		TOTAL	% Regras	% BD
BD-1	R1	2	100%	0	0%	0	0%	2	5%	1%
	R2	10	59%	3	18%	4	24%	17	40%	5%
BD-2	R1	3	60%	1	20%	1	20%	5	12%	1%
	R2	1	33%	1	33%	1	33%	3	7%	1%
BD-3	R1	1	50%	0	0%	1	50%	2	5%	1%
	R2	0	0%	0	0%	3	100%	3	7%	1%
BD-4	R1	3	38%	0	0%	5	63%	8	19%	2%
	R2	1	50%	1	50%	0	0%	2	5%	1%
% de ativação das regras em relação a BD do Tinsley										12%

5.3.3 Cenário 3: avaliação do desempenho do MP-Draughts contra outros jogadores automáticos não supervisionados

Este cenário apresenta a performance de ambas as arquitetura do MP-Draughts em jogos competitivos contra os jogadores não supervisionados VisionDraughts e LS-VisionDraughts. Como citado no Capítulo 3, não foi possível realizar competições contra outros jogadores automáticos não supervisionados recentemente propostos na literatura [25], [26], uma vez que não existe interface disponível desses agentes.

Os jogos foram executados na mesma BD de Tinsley utilizada no cenário de avaliação das regras. Nesse sentido, as situações de ativação das regras de exceção são as mesmas, de modo que as porcentagens de ativação dessas regras se mantém equivalentes às apresentadas na Tabela 17.

A avaliação de desempenho de ambas as arquiteturas contra cada um de seus oponentes foi dividida em três etapas: primeiro, foram comparados o resultado final de todos os jogos entre as duas arquiteturas contra o oponente. Esse resultado mostra o quão eficiente é o multiagente em relação ao seu oponente e o quanto a nova arquitetura é mais eficiente do que a original. Segundo, foram avaliados os resultados dos jogos entre a nova arquitetura e o oponente, nos quais as regras de exceção foram ativadas. Esta avaliação teve o objetivo de verificar se as regras foram eficientes na alocação dos EGAs. Por último, ainda nos jogos em que as regras foram ativadas, foram comparados os resultados obtidos pelas duas arquiteturas. Esta comparação é muito importante, pois mostra em quantos desses jogos a indicação das regras foi capaz de melhorar o resultado desses jogos.

As Tabelas 18, 19, 20 apresentam os resultados das três etapas de avaliação de desempenho das arquiteturas do MP-Draughts em jogos contra o VisionDraughts. Na Tabela 18 são apresentados os resultados dos jogos executados em todos os 343 tabuleiros da BD de Tinsley entre as duas versões do MP-Draughts contra o VisionDraughts. Como em cada tabuleiro são executados 2 jogos, ora com peças pretas, ora com peças vermelhas,

foram executados um total de 686 jogos. Os resultados mostram que ambas as versões do MP-Draughts são mais eficientes do que o VisionDraughts, sendo que a versão original apresenta uma superioridade de vitórias em torno de 27% e, a nova versão, em torno de 31% quando comparado com as vitórias obtidas pelo oponente. Focando na quantidade de vitórias, derrotas em empates obtidos por ambas as versões do multiagente, é possível observar que a nova versão obteve mais vitórias, menos derrotas e empates que a versão original. A maior quantidade de vitórias obtidas pela nova versão está vinculada a alocação mais adequada dos EGAs que atuaram nesses jogos. O fato de a nova versão não ter diminuído os casos de derrotas ocorridos nos jogos em relação a versão original deve estar relacionado à falta de conhecimento do multiagente sobre essas situações de jogo e não à alocação inadequada dos EGAs, visto que nenhumas dessas versões conseguiu bons resultados.

Tabela 18 – Resultado dos jogos competitivos entre as duas arquiteturas do MP-Draughts contra o VisionDraughts.

	Vitórias		Derrotas		Empates	
MP-RN-RE x VisionDraughts	334	49%	129	18%	223	33%
MP-RN x VisionDraughts	321	46%	131	19%	234	34%

A Tabela 19 mostra os resultados dos jogos envolvendo a nova arquitetura do MP-Draughts contra o VisionDraughts em que as regras de exceção foram ativadas. Observe que, quando ativadas, as regras obtiveram pelo menos 67% de sucesso na tarefa de alocação dos EGAs, com exceção da R2 da BD-4, que obteve 50%. Considerando o resultado desses jogos, apenas em 2 dentre as 41 vezes em que as regras foram ativadas (somatório das vitórias, empates e derrotas) o resultado do jogo foi insatisfatório para o multiagente; em outras 32 situações, o resultado foi satisfatório; nas 7 remanescentes, foi neutro. Contudo, tais resultados não são suficientes para garantir que as regras melhoraram significativamente a capacidade de alocação da RN, visto que a alocação da RN para essas mesmas situações de jogos também pode ter sido eficiente. Nesse sentido, a Tabela 20, mostra a quantidade desses jogos cujos resultados foram alterados devido à ativação das regras de exceção. Esta tabela foi obtida a partir da comparação do resultado de cada um dos 41 jogos entre cada versão do MP-Draughts contra o VisionDraughts. Como pode ser observado, 16 dos 41 jogos tiveram seus resultados alterados devido a ativação das regras. Dentre esses, positivamente, 13 casos de empate foram convertidos em vitórias e 1 caso de derrota foi convertido em vitória. Em contrapartida, 2 casos de vitórias foram convertidos em um empate e uma derrota. Ou seja, em 87% das vezes em que as regras foram ativadas, elas conseguiram converter positivamente o resultado do jogo.

As Tabelas 21, 22 e 23 apresentam os resultados das três etapas de avaliação de desempenho de ambas as arquiteturas em jogos contra o LS-VisionDraughts. Na Tabela 21 é apresentado o resultado dos jogos executados entre as duas versões contra o LS-

Tabela 19 – Resultado dos jogos entre a nova arquitetura do MP-Draughts contra o VisionDraughts, em que as regras de exceção foram ativadas.

$MP-RN-RE \times VisionDraughts$							
		Vitórias		Derrotas		Empates	
BD-1	R1	2	100%	0	0%	0	0%
	R2	12	71%	1	6%	4	24%
BD-2	R1	4	80%	1	20%	0	0%
	R2	2	67%	0	0%	1	33%
BD-3	R1	2	100%	0	0%	0	0%
	R2	3	100%	0	0%	0	0%
BD-4	R1	6	86%	0	0%	1	14%
	R2	1	50%	0	0%	1	50%

Tabela 20 – Quantidade de jogos cujos resultados foram alterados devido a ativação das regras de exceção nos jogos contra o jogador VisionDraughts.

Quantidade de jogos	
Empate -> Vitória	13
Empate -> Derrota	0
Derrota -> Empate	0
Derrota -> Vitória	1
Vitória -> Empate	1
Vitória -> Derrota	1

VisionDraughts. Como pode ser observado, ambas as versões do MP-Draughts são mais eficientes que o LS-VisionDraughts. Considerando que os casos empates, que representa 85% dos casos, são neutros no que diz respeito ao desempenho de jogo, a quantidade de vitórias obtidas pelas versões do MP-Draughts em relação as derrotas (que representa vitórias do oponente) é suficiente para comprovar a superioridade do multiagente em relação ao seu oponente. Os resultados mostram que a versão original obteve 11 vitórias a mais que o LS-VisionDraughts enquanto a nova versão obteve 19. Todavia, não pode ser ignorado que a grande quantidade de empates obtida nesses jogos indica que o LS-VisionDraughts é um forte oponente do MP-Draughts. Focando nas vitórias e derrotas obtidas por ambas as versões do MP-Draughts, é possível perceber que a nova versão é pelo menos 3% mais eficiente que a versão original.

Tabela 21 – Resultado geral dos jogos competitivos entre as duas versões do MP-Draughts contra o LS-VisionDraughts.

	Vitórias		Derrotas		Empates	
MP-RN-RE x LS-VisionDraughts	59	9%	40	6%	587	85%
MP-RN x LS-VisionDraughts	56	8%	45	7%	585	85%

A Tabela 22 mostra os resultados dos jogos envolvendo a nova arquitetura do MP-Draughts contra o LS-VisionDraughts em que as regras de exceção foram ativadas. Observa-

se que as situações de empates representam a maioria desses resultados e que as situações de vitórias são superiores as de derrota, assim como no desempenho geral do multiagente apresentado na Tabela 21. As situações de empate representam 31 dos 41 jogos, as situações de vitória 7 e as de derrota apenas 3. Tais resultados, no entanto, não são suficientes para comprovar a eficiência da utilização da combinação das regras e da RN comparadas com a utilização apenas da RN. Sendo assim, a Tabela 23 apresenta quantos desses jogos tiveram os resultados alterados devido à ativação das regras, quando comparados com os mesmos jogos executados entre a versão original (sem regras) e o LS-VisionDraughts. Como pode ser observado, 10 dos 41 jogos tiveram seus resultados alterados. Dentre esses, 4 casos de empate foram convertidos em vitórias e 5 casos de derrota foram convertidos em empate enquanto, apenas 1 caso de vitória foi convertido empate. Ou seja, em 90% das vezes em que as regras foram ativadas, estas conseguiram converter positivamente o resultado do jogo. Observe que, mesmo em jogos nos quais a vantagem do multiagente é pequena em relação ao oponente, no caso do LS-VisionDraughts, as regras conseguem obter melhores resultados.

Tabela 22 – Resultado dos jogos entre a nova arquitetura do MP-Draughts contra o LS-VisionDraughts, em que as regras de exceção foram ativadas.

$MP - RN - RE \times LS - VisionDraughts$							
		Vitórias		Derrotas		Empates	
BD-1	R1	1	50%	0	0%	1	50%
	R2	2	12%	1	6%	14	82%
BD-2	R1	0	0%	0	0%	5	100%
	R2	1	33%	0	0%	2	67%
BD-3	R1	0	0%	0	0%	2	100%
	R2	0	0%	0	0%	3	100%
BD-4	R1	3	43%	2	29%	2	29%
	R2	0	0%	0	0%	2	100%

Tabela 23 – Quantidade de jogos cujos resultados foram alterados devido a ativação das regras de exceção nos jogos contra o LS-VisionDraughts.

Quantidade de jogos	
Empate -> Vitória	4
Empate -> Derrota	0
Derrota -> Empate	5
Derrota -> Vitória	0
Vitória -> Empate	1
Vitória -> Derrota	0

De modo geral, os resultados obtidos nos cenários de teste até aqui avaliados foram bastante coerentes e satisfatórios. Coerentes no que diz respeito a cobertura das regras em relação as situações especiais identificadas nas BDs de jogos (aproximadamente 12%)

Tabela 24 – Média de coincidência de movimentos executados por ambas as versões do MP-Draughts e pelo VisionDraughts quando comparado com os movimentos que seriam executados pelo Cake na mesma situação.

Jogos	Média	
$MP - RN - RE \times VisionDraughts$	**67%**	41%
$MP - RN \times VisionDraughts$	**62%**	31%

e, satisfatórios no que diz respeito a habilidade dessas regras em identificar as situações de má alocação da RN, refinando adequadamente essas alocações de modo a aumentar o desempenho geral do multiagente.

5.3.4 Cenário 4: avaliação das escolhas de movimentos das versões do MP-Draughts contra seus oponentes sob a perspectiva do Cake

Este cenário compara os movimentos executados pelos agentes não supervisionados MP-Draughts (ambas as versões), VisionDraughts e LS-VisionDraughts com os que seriam executados pelo jogador supervisionado Cake na mesma situação. Esta comparação mostra a coincidência de raciocínio, indicada pelos movimentos coincidentes, de cada um dos agentes não supervisionados aqui avaliados em relação ao Cake na fase de final de jogo. Tais agentes não foram avaliados em jogos direto contra o Cake, pois os mesmos ainda não são competitivos contra agentes que contam com forte supervisão no processo de aprendizagem.

Para a execução deste cenário, foram considerados 6 dos 41 jogos executados no cenário de avaliação de performance desses jogadores. A comparação foi feita em 6 jogos entre cada umas das versões do MP-Draughts contra o VisionDraughts e contra o LS-VisionDraughts. Não foi possível avaliar todos os 41 jogos, visto que esta é uma tarefa manual e onerosa. Foram analisados todos os movimentos executados até a finalização do jogo.

A Tabela 24 apresenta os resultados da comparação feita entre ambas as versões do MP-Draughts em jogos contra o VisionDraughts. Ambas as versões obtiveram mais coincidência de movimentos do que o VisionDraughts. A nova arquitetura do multiagente obteve 67% de movimentos coincidentes contra 41% de coincidência obtida pelo VisionDraughts. Já a versão original do multiagente obteve 62% de movimentos coincidentes contra 31% do VisionDraughts. A Tabela 25 apresenta os resultados da mesma comparação feita entre ambas as versões do MP-Draughts contra o LS-VisionDraughts. Novamente, ambas as versões do MP-Draughts obtiveram mais coincidência de movimentos quando comparados com o Cake do que o oponente. Enquanto a nova arquitetura do MP-Draughts obteve 57% de movimentos coincidentes, o LS-VisionDraughts obteve apenas 36%. Já a versão original obteve 52% contra 31% do LS-VisionDraughts.

Tabela 25 – Média de coincidência de movimentos executados por ambas as versões do MP-Draughts e pelo LS-VisionDraughts quando comparado com os movimentos que seriam executados pelo Cake na mesma situação.

Jogos	Média	
$MP - RN - RE \times LS - VisionDraughts$	**57%**	36%
$MP - RN \times LS - VisionDraughts$	**52%**	31%

Os resultados obtidos neste cenário confirmam o seguinte: a boa capacidade de jogo do MP-Draughts, visto que suas escolhas de movimento coincidem, na maioria das vezes, com as escolhas de um agente eficiente e fortemente supervisionado; a superioridade do MP-Draughts em relação aos seus oponentes não supervisionados, uma vez que a coincidência de raciocínio do multiagente em relação ao raciocínio do Cake é maior do que a de seus oponentes e, por último, reafirmam que as regras de exceção são uma boa proposta para tratar as situações excepcionais de alocação da RN, uma vez que os agentes alocados por elas são mais eficientes do que os alocados pela RN para atuar nessas situações.

5.4 Considerações Finais

As duas principais contribuições apresentadas neste capítulo são: a utilização de regras de exceção para tratar situações especiais em que uma RN de agrupamento, quando utilizada como RN de classificação, não é capaz de desempenhar adequadamente seu papel e; a utilização dessas regras de exceção para refinar a capacidade de alocação de agentes em sistemas multiagentes, os quais utilizam RN para tal tarefa.

A primeira contribuição apresenta um método que extrai regras de exceção a partir de situações especiais que ocorrem na fase final do jogo de Damas, nas quais a RN utilizada para alocação dos agentes não desempenha adequadamente seu papel. A partir dos resultados obtidos por este método foi possível verificar que as regras de exceção abstraíram os conhecimentos inerentes a essas situações, representando-as adequadamente. Analisando o método de extração de regras de uma maneira mais genérica, é possível observar que ele pode ser aplicado a outros problemas de classificação de dados envolvendo outras técnica de RNs, desde que nesses seja possível identificar, além das situações mal classificadas pela RN, qual seria a melhor classificação nessas situações. Note que o mesmo pode ser aplicado tanto para RN de classificação como de agrupamento de dados.

A segunda contribuição deste capítulo baseia-se no uso das regras de exceção para auxiliar na tarefa de alocação de agentes de um sistema multiagente, em situações em que a técnica de alocação utilizada não é suficientemente adequada para a tarefa. Nesse sentido, e aplicado ao domínio do jogo de Damas, o método proposto combina RNs de agrupamento com regras de exceção, as quais em conjunto definem os agentes adequados para atuarem nas diferentes situações que ocorrem na fase de final de jogo. Em tal

método, cada vez que o multiagente precisa definir um agente para atuar no ambiente, a RN é responsável por abstrair as informações do ambiente e, com base nessas informações, indicar o agente mais adequado para a situação. Em complemento a indicação da RN, as regras de exceção são responsáveis por identificar se o ambiente em questão corresponde a uma situação de má atuação da RN e, caso afirmativo, é responsável por refinar a indicação da RN.

Diante dos resultados obtidos neste capítulo, o qual combina tais métodos para aprimorar a capacidade de alocação de agentes do multiagente MP-Draughts, foi possível comprovar que, de fato, esses cumprem eficientemente a proposta, além de contribuírem para a performance geral do multiagente.

Capítulo 6

Conclusão

Este trabalho apresentou três novas propostas para aprimorar o processo de aprendizagem e alocação de agentes inteligentes em plataformas multiagentes, que atuam em ambientes com alta complexidade de resolução. Tais propostas foram baseadas em técnicas de aprendizagem de máquina não supervisionadas, as quais foram utilizadas para: otimizar o processo de aprendizagem dos agentes que compõem o sistema multiagente, delineando e delimitando as habilidades que devem ser desenvolvidas por cada um de acordo com o ambiente em que atuam; refinar a representação desses ambientes, priorizando as informações que, de fato, são pertinentes para a tomada de decisão desses agentes e; definir um processo de alocação eficiente, que permita que os agentes alocados para atuarem nos ambientes sejam os mais adequados, independente das excepcionalidades que possam ocorrer nesses ambientes. Devido a sua complexidade técnica e o alto espaço de estados inerentes à sua resolução, o domínio do jogo Damas foi utilizado como ambiente de desenvolvimento e avaliação dessas propostas. Mais especificamente, tais propostas foram implementadas sobre a arquitetura do multiagente não supervisionado MP-Draughts.

Para o cumprimento da primeira proposta foi implementada a RN adaptativa ASONDE, a qual aprimora sua versão original tornando-a apta para minerar bases de dados finitas e estáveis. Tal RN foi utilizada na arquitetura do MP-Draughts para identificar os diferentes perfis de conhecimentos inerentes à fase de final de jogo (foco de atuação do multiagente) e, a partir de então, definir os *clusters* necessários para agrupar tais conhecimentos. Esses *clusters* foram então utilizados para definir a quantidade de agentes necessários para atuar nessa fase do jogo, além de servirem como fonte de conhecimento para o treinamento desses agentes.

Sabendo quais são os diferentes perfis de conhecimentos inerentes à fase final de um jogo de Damas, a segunda proposta deste trabalho teve por objetivo selecionar automaticamente as características mais adequadas para representar as diferentes situações que ocorrem no ambiente do jogo. As características utilizadas para a representação correspondem a um subconjunto das características *NetFeatureMap* propostas por Samuel [14]. Para tanto, foi implementada uma abordagem baseada na mineração de padrões frequen-

tes que ocorrem em base de dados de jogos de grandes especialistas. Inicialmente, foram obtidos os conjuntos de características adequados para representar todo o ambiente do jogo, independente de sua fase. Isso foi feito para comprovar a qualidade da abordagem proposta quando comparada com outra já existente para o mesmo propósito, que inclusive, é computacionalmente inviável de ser utilizada no sistema multiagente aqui investigado. Comprovada sua qualidade, a abordagem proposta foi aplicada à fase de final de jogo para selecionar as características mais adequadas para representar os diferentes perfis que esta fase possui. Tal abordagem se mostrou muito apropriada para a situação.

Finalmente, a terceira proposta implementou um método de alocação de agentes para sistemas multiagentes que combina RNs de agrupamento, no caso a ASONDE, e regras de exceção. Tal método foi implementado para resolver os problemas de má atuação da ASONDE durante o processo de alocação dos agentes. Até então, essa RN era responsável por abstrair as informações do ambiente e, com base nessas informações, indicar o melhor agente do MP-Draughts para atuar no mesmo. Porém, nem todas as indicações da RN eram adequadas, de modo que o agente indicado nem sempre era o melhor. Para essas situações, foram obtidas regras de exceções, as quais complementam o conhecimento sobre o domínio e refinam a indicação da RN. Dessa maneira, utilizando o método proposto, cada vez que é necessário definir um agente para atuar no jogo, a ASONDE é responsável por abstrair as informações relativas ao estado corrente do ambiente e indicar o agente mais adequado para atuar em tal estado. Por sua vez, as regras de exceção são responsáveis por refinar e otimizar a indicação da RN nas situações excepcionais em que esta não se mostrar suficientemente capacitada para fazê-la. Os resultados obtidos dessa proposta, comprovaram que as regras de exceção são uma excelente solução para tratar situações excepcionais relacionadas a má atuação da RN na tarefa de alocação de agentes.

Como pôde ser observado no decorrer dos capítulos deste trabalho, os resultados parciais obtidos da implementação de cada proposta, assim como o resultado final que implementa todas elas na arquitetura do MP-Draughts, confirmaram que elas foram eficientes para tratar os problemas para os quais foram projetadas. Além disso, tais propostas podem ser adaptadas e aplicadas a outros tipos de problemas relacionados a aprendizagem de máquina não supervisionada.

6.1 Principais Limitações do Trabalho

As principais limitações encontradas durante o desenvolvimento deste trabalho foram:

1. Restrição na quantidade de *hardwares* disponíveis para executar os experimentos do trabalho. Por se tratar de um sistema multiagente, cuja arquitetura é composta por várias MLPs que aprendem por reforço ao longo de vários ciclos de treinamento, qualquer alteração, para ser validada, envolvia vários ciclos de treinamento e torneio

evolvendo os vários agentes que compõem o multiagente. O tempo de processamento gasto para validar qualquer simples alteração era muito alto;

2. A falta de conhecimento profundo sobre o domínio do jogo de Damas por parte dos autores atrasou algumas decisões relacionadas a escolha das técnicas mais adequadas para resolver determinados problemas. Por outro lado, esta falta de informação impulsionou a busca por técnicas de dependessem o mínimo possível do conhecimento humano sobre o domínio e que fossem adequadas para o mesmo;

3. Os testes de desempenho do MP-Draughts ficaram limitados a jogos contra poucos oponentes não supervisionados. Isso ocorreu porque, em contato com alguns autores de jogadores automáticos de Damas, os mesmos argumentaram não possuir interface de jogo para seus jogadores, ou ainda, que os códigos referentes aos mesmos não estão disponíveis para a comunidade científica. Uma forma de diminuir o impacto dessa limitação foi avaliar as ações do multiagente em comparação as ações do jogador supervisionado Cake [36].

6.2 Trabalhos Futuros

Durante o desenvolvimento deste trabalho várias possibilidades de pesquisas foram identificadas, as quais podem tratadas em propostas de trabalhos futuros, a saber:

1. Aplicar a ASONDE em outros domínios de problemas que possuam bases de dados finitas e estáveis, por exemplo as BDs da UCI [101]. Tal aplicação terá o intuito de reforçar a capacidade desta RN em identificar as várias regiões de agrupamento dos dados representados nessas BDs, além de comprovar que esta RN é adequada para estes tipos de problemas (em que os dados não são de fluxo contínuo);

2. Adaptar a abordagem de mineração de características frequentes, de modo que ela seja capaz de identificar as características que ocorrem com menos frequência no ambiente, porém são muito importantes para a representação adequada deste ambiente. Uma possibilidade seria ponderar a ocorrência das características baseando-se no momento em que elas tendem a ocorrer no ambiente. Por exemplo, na fase inicial do jogo as características que representam posições estratégicas são mais importantes, do mesmo modo que no final do jogo são mais importantes aquelas que avançam no tabuleiro em direção a formação de damas;

3. Investigar, utilizando a abordagem de mineração de padrões frequentes, as melhores representações do ambiente do jogo de Damas considerando todas as 26 características propostas por Samuel [14], além de outras, propostas por outros pesquisadores do domínio [19], [90];

4. Aplicar a técnica de alocação de agentes a outros domínios que envolvem sistemas multiagentes. Além disso, aplicar tal técnica em outros tipos de problemas relacionados a classificação de dados, uma vez que esta funciona também como uma técnica de classificação;

5. Aplicar a técnica de geração de regras de exceção a outros problemas de classificação de dados envolvendo redes neurais de agrupamento de dados, nos quais seja possível identificar, além das situações mal classificadas pela RN, quais seriam as melhores classificações para essas situações.

6.3 Contribuições em Produção Bibliográfica

A seguir são apresentadas as contribuições bibliográficas obtidas durante a realização deste trabalho, as quais estão divididas em artigos publicados em conferências internacionais e artigos submetidos a periódicos internacionais.

Artigos publicados em eventos internacionais:

1. Duarte, V. A. R. and Julia, R. M. S.: *MP-Draughts: Ordering the Search Tree and Refining the Game Board Representation to Improve a Multi-agent System for Draughts.* In: IEEE 24th International Conference on Tools with Artificial Intelligence (ICTAI'12), Athens, Greece, 2012. (Ciência da Computação: Qualis A2).

2. Duarte,V. A. R. and Julia, R. M. S. and Albertini, M. K. and Neto, H. C.: *MP-Draughts: Unsupervised Learning Multi-agent System Based on MLP and Adaptive Neural Networks.* In: IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI'15), Vietri sul Mare, Italy, 2015. (Ciência da Computação: Qualis A2).

3. Duarte, V. A. R. and Julia, R. M. S.: *Improving NetFeatureMap-based Representation through Frequent Pattern Mining in a Specialized Database.* In: IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI'16), San Jose, California, USA, 2016. (Ciência da Computação: Qualis A2).

4. Duarte, V. A. R. and Julia, R. M. S.: *Improving the State Space Representation through Association Rules.* In: 15th IEEE International Conference on Machine Learning and Applications (ICMLA'16), Anaheim, California, USA, 2016. (Ciência da Computação: Qualis B2).

Artigos submetidos a periódicos internacionais:

1. Duarte, V. A. R. and Julia, R. M. S.: *Improving the Agent Allocation Process on Multi-agent Systems by Combining Clustering Neural Networks and Exception Rules.* Journal of Knowledge and Information Systems (KAIS). (Ciência da Computação: Qualis B1).

2. Tomaz, L. B. P. and Julia, R. M. S. and Duarte, V. A. R.: *A Multiagent Player System Composed by Expert Agents in Specific Game Stages Operating in High Performance Environment.* Journal of Applied Intelligence (Appl Intell). (Ciência da Computação: Qualis B1).

Referências

1 MCCULLOCH, W.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **Bulletin of Mathematical Biophysics**, v. 5, p. 115–133, 1943.

2 HERIK, H. J. V.; UITERWIJK, J. W. H. M.; RIJSWIJCK, J. V. Games solved: Now and in the future. **Artificial Intelligence**, v. 134, p. 277–311, 2002.

3 CAMPOS, P.; LANGLOIS, T. Abalearn: Efficient self-play learning of the game abalone. In: **INESC-ID, Neural Networks and Signal Processing Group**. [S.l.: s.n.], 2003.

4 NETO, H. C. **Uma Nova Abordagem de Aprendizagem de Máquina Combinando Elicitação Automática de Casos, Aprendizagem por Reforço e Mineração de Padrões Sequenciais Para Agentes Jogadores de Damas**. Tese (Doutorado) — Univeridade Federal de uberlândia, Novembro 2016.

5 MITCHELL, T. M. **Machine learning**. [S.l.]: McGraw-Hill, 1997. (McGraw Hill series in computer science). ISBN 978-0-07-042807-2.

6 DUARTE, V. A. R. et al. Mp-draughts: Unsupervised learning multi-agent system based on mlp and adaptive neural networks. In: **IEEE 27th International Conference on Tools with Artificial Intelligence,ICTAI 2015, Vietri sul Mare, Italy, November 9-11, 2015**. [S.l.: s.n.], 2015.

7 WOOLDRIDGE, M. **An Introduction to Multiagent Systems**. 2. ed. New York, NY, USA: John Wiley & Sons, Inc., 2009.

8 JENNINGS, N. R.; WOOLDRIDGE, M. J. **Applications of intelligent agents - Agent technology: foundations, applications, and markets**. [S.l.]: Springer-Verlag, 1998.

9 MOULIN, B.; CHAIB-DRAA, B. An overview of distributed artificial intelligence. **Foundations of Distributed Artificial Intelligence**, p. 3–56, 1996.

10 G., Z.; C., C. **Gamification by Design: Implementing Game Mechanics in Web and Mobile Apps**. 1st edition. ed. [S.l.]: O' Reilly Media, 2011. v. 1.

11 KAPP, K. M. **The Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education**. 1st. ed. [S.l.]: Pfeiffer & Company, 2012.

12 FADEL, L. M. et al. **Gamificação na educação**. [S.l.]: Pimenta Cultural, 2014.

13 NEUMANN, J. V.; MORGENSTERN, O. **Theory of Games and Economic Behavior**. [S.l.]: Princeton University Press, 1944.

14 SAMUEL, A. L. Some studies in machine learning using the game of checkers. **IBM Journal of Research and Development**, v. 3, n. 3, p. 210–229, 1959.

15 _____. Some studies in machine learning using the game of checkers ii. **IBM Journal of Research and Development**, v. 11, n. 6, p. 601–617, 1967.

16 FÜRNKRANZ, J. Machine learning in games: A survey. In: **Machines that Learn to Play Games, chapter 2**. [S.l.]: Nova Science Publishers, 2000. p. 11–59.

17 MCCARTHY, J.; FEIGENBAUM, E. A. In memoriam: Arthur samuel - pioneer in machine learning. **AI Magazine**, v. 11, n. 3, p. 10–11, 1990.

18 SHANNON, C. E. XXII. Programming a computer for playing chess. **Philosophical Magazine (Series 7)**, Taylor & Francis, v. 41, n. 314, p. 256–275, 1950.

19 SCHAEFFER, J. et al. Chinook: The world man-machine checkers champion. **AI Magazine**, v. 17, n. 1, p. 21–30, 1996.

20 _____. Checkers is solved. **Science Express**, v. 328, n. 5844, p. 1518, 2007.

21 CAIXETA, G. S.; JULIA, R. M. da S. A draughts learning system based on neural networks and temporal differences: The impact of an efficient tree-search algorithm. In: **Advances in Artificial Intelligence - SBIA 2008, 19th Brazilian Symposium on Artificial Intelligence, Savador, Brazil, October 26-30, 2008. Proceedings**. [S.l.: s.n.], 2008. p. 73–82.

22 DUARTE, V. A. R. et al. Mp-draughts: a multiagent reinforcement learning system based on MLP and kohonen-som neural networks. In: **Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, San Antonio, TX, USA, 11-14 October 2009**. [S.l.: s.n.], 2009. p. 2270–2275.

23 DUARTE, V. A. R.; JULIA, R. M. da S. Mp-draughts: Ordering the search tree and refining the game board representation to improve a multi-agent system for draughts. In: **IEEE 24th International Conference on Tools with Artificial Intelligence,ICTAI 2012, Athens, Greece, November 7-9, 2012**. [S.l.: s.n.], 2012. p. 1120–1125.

24 BARCELOS, A. R. A.; JULIA, R. M. da S.; JUNIOR, R. M. D-visiondraughts: a draughts player neural network that learns by reinforcement in a high performance environment. In: **ESANN 2011, 19th European Symposium on Artificial Neural Networks, Bruges, Belgium**. [S.l.: s.n.], 2011.

25 CHEHELTANI, S. H.; EBADZADEH, M. M. Immune based fuzzy agent plays checkers game. **Applied Soft Computing**, v. 12, n. 8, p. 2227 – 2236, 2012. ISSN 1568-4946.

26 AL-KHATEEB, B.; KENDALL, G. Introducing individual and social learning into evolutionary checkers. **Computational Intelligence and AI in Games, IEEE Transactions on**, v. 4, n. 4, p. 258–269, Dec 2012.

27 TOMAZ, L. B. P.; JULIA, R. M. S.; BARCELOS, A. B. A. Improving the accomplishment of a neural network based agent for draughts that operates in a distributed learning environment. In: **IEEE 14th International Conference on Information Reuse & Integration, IRI 2013, San Francisco, CA, USA, August 14-16, 2013**. [S.l.: s.n.], 2013. p. 262–269.

28 ELNAGGAR, A. A. et al. Autonomous checkers robot using enhanced massive parallel game tree search. In: **2014 9th International Conference on Informatics and Systems (INFOS)**. [S.l.: s.n.], 2014. p. 35–44.

29 NETO, H. C. et al. Ls-visiondraughts: improving the performance of an agent for checkers by integrating computational intelligence, reinforcement learning and a powerful search method. **Applied Intelligence**, Springer US, p. 1–26, 2014. ISSN 0924-669X.

30 NETO, H. C.; JULIA, R. M. S. Ace-rl-checkers: Improving automatic case elicitation through knowledge obtained by reinforcement learning in player agents. In: **2015 IEEE Conference on Computational Intelligence and Games, CIG 2015, Tainan, Taiwan, August 31 - September 2, 2015**. [S.l.: s.n.], 2015. p. 328–335.

31 NETO, H. C.; JULIA, R. M. S.; DUARTE, V. A. R. Improving the accuracy of the cases in the automatic case elicitation-based hybrid agents for checkers. In: **Tools with Artificial Intelligence (ICTAI), 2015 IEEE 27th International Conference on**. [S.l.: s.n.], 2015. p. 912–919. ISSN 1082-3409.

32 FRANKLAND, C.; PILLAY, N. Evolving heuristic based game playing strategies for checkers incorporating reinforcement learning. In: _____. **Advances in Nature and Biologically Inspired Computing: Proceedings of the 7th World Congress on Nature and Biologically Inspired Computing (NaBIC2015) in Pietermaritzburg, South Africa, held December 01-03, 2015**. [S.l.]: Springer International Publishing, 2016. p. 165–178.

33 SINCLAIR, D. Using example-based reasoning for selective move generation in two player adversarial games. In: _____. **Advances in Case-Based Reasoning: 4th European Workshop, EWCBR-98 Dublin, Ireland, September 23–25, 1998 Proceedings**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998.

34 FLINTER, S.; KEANE, M. T. On the automatic generation of case libraries by chunking chess games. In: **In M. Veloso and A. Aamodt (Eds.), Proceedings of the 1st International Conference on Case Based Reasoning (ICCBR-95**. [S.l.]: Springer Verlag, 1995. p. 421–430.

35 SILVER, D. et al. Mastering the game of go with deep neural networks and tree search. **Nature**, v. 529, p. 484–503, 2016.

36 FIERZ, M. C. **Cake**. 2015. Disponível em: <http://www.fierz.ch/cake.php>.

37 SCHAEFFER, J. et al. A world championship caliber checkers program. **Artificial Intelligence**, v. 53, p. 53–2, 1992.

38 ALBERTA, D. o. C. S. G. G. University of. **Chinook**. 2016. Disponível em: <http://webdocs.cs.ualberta.ca/~chinook/index.php>.

39 LYNCH, M.; GRIFFITH, N. Neurodraughts: The role of representation, search, training regime and architecture in a td draughts player. **Eighth Ireland Conference on Artificial Intelligence**, Ireland, p. 67–72, 1997.

40 FELDMAN, R. et al. Parallel algorithms for machine intelligence and vision. In: KUMAR, V.; GOPALAKRISHNAN, P. S.; KANAL, L. N. (Ed.). New York, NY, USA: Springer-Verlag New York, Inc., 1990. cap. Distributed game tree search, p. 66–101.

41 CHELLAPILLA, K.; FOGEL, D. B. Anaconda defeats hoyle 6-0: A case study competing an evolved checkers program against commercially available software. In: **Proceedings of the 2000 Congress on Evolutionary Computation CEC00**. [S.l.]: IEEE Press, 2000. p. 857–863.

42 FOGEL, D. B.; CHELLAPILLA, K. Verifying anaconda's expert rating by competing against chinook: Experiments in co-evolving a neural checkers player. **Neurocomputing**, v. 42, n. 1-4, p. 69–86, 2001.

43 ELNAGGAR, A. A. et al. Enhanced parallel negamax tree search algorithm on gpu. In: **Progress in Informatics and Computing (PIC), 2014 International Conference on**. [S.l.: s.n.], 2014. p. 546–550.

44 WANG, L.; WANG, Y.; LI, Y. Mining experiential patterns from game-logs of board game. **Int. J. Comput. Games Technol.**, Hindawi Publishing Corp., New York, NY, United States, v. 2015, 2015.

45 WEBER, B. G.; MATEAS, M. A data mining approach to strategy prediction. In: **Proceedings of the 5th International Conference on Computational Intelligence and Games (CIG 09)**. [S.l.]: IEEE Press, 2009. p. 140–147.

46 TAKEUCHI, S.; KANEKO, T.; YAMAGUCHI, K. Evaluation of game tree search methods by game records. **IEEE Transactions on Computational Intelligence and AI in Games**, v. 2, p. 288–302, 2010.

47 ZAMBONELLI, F. et al. Developing pervasive multi-agent systems with nature-inspired coordination. **Pervasive and Mobile Computing**, v. 17, Part B, p. 236 – 252, 2015.

48 PALÁCIOS, R. H. C. et al. A novel multi-agent approach to identify faults in line connected three-phase induction motors. **Applied Soft Computing**, v. 45, p. 1 – 10, 2016.

49 HSIEH, F.-S.; LIN, J.-B. A self-adaptation scheme for workflow management in multi-agent systems. **Journal of Intelligent Manufacturing**, v. 27, n. 1, p. 131–148, 2016.

50 MALIALIS, K.; DEVLIN, S.; KUDENKO, D. Resource abstraction for reinforcement learning in multiagent congestion problems. In: **Proceedings of the 2016 International Conference on Autonomous Agents; Multiagent Systems**. [S.l.]: International Foundation for Autonomous Agents and Multiagent Systems, 2016. (AAMAS '16), p. 503–511.

51 RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 3. ed. [S.l.]: Prentice Hall, 2009.

52 SILVA, I. N.; SPATTI, D. H.; FLAUZINO, R. A. **Redes Neurais Artificiais: para engenharia e ciência aplicada**. [S.l.]: ArtLiber, São Paulo, 2010.

53 HAYKIN, S. **Redes Neurais: Princípios e Prática (2º edição)**. Porto Alegre, RS: Bookman Editora, 2001.

54 SUTTON, R.; BARTO, A. **Reinforcement Learning: An Introduction**. [S.l.]: MIT Press, 1998.

55 SCHAEFFER, J.; HLYNKA, M.; JUSSILA, V. Temporal difference learning applied to a high performance game-playing program. **Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)**, p. 529–534, 2001.

56 PLAAT, A. et al. A new paradigm for minimax search. 1995.

57 PLAAT, A. **Research Re: search & Re-search**. Tese (Doutorado), Rotterdam, Netherlands, 1996. Disponível em: <citeseer.ist.psu.edu/plaat96research.html>.

58 KOHONEN, T. The self-organizing map. **Proc. IEEE**, p. 1464–1480, 1990.

59 _____. **Self-Organizing Maps**. [S.l.]: Springer, 2001.

60 CARPENTER, G. A.; GROSSBERG, S.; ROSEN, D. B. Art 2-a: An adaptive resonance algorithm for rapid category learning and recognition. **Neural Networks**, v. 4, n. 4, p. 493–504, 1991.

61 GROSSBERG, S. Adaptive pattern classification and universal recoding: I. parallel development and coding of neural feature detectors. **Biological Cybernetics**, v. 23, n. 3, p. 121–134, 1976.

62 _____. Adaptive pattern classification and universal recording: II. Feedback, expectation, olfaction, illusions. **Biological Cybernetics**, v. 23, p. 187–202, 1976.

63 CARPENTER, G. A.; GROSSBERG, S. The art of adaptive pattern recognition by a self-organizing neural network. **Computer**, IEEE Computer Society Press, v. 21, n. 3, p. 77–88, 1988.

64 FRANK, T.; KRAISS, K.-F.; KUHLEN, T. Comparative analysis of fuzzy art and art-2a network clustering performance. **IEEE Transactions on Neural Networks**, v. 9, n. 3, p. 544–559, 1998.

65 ALBERTINI, M. K.; MELLO, R. F. de. A self-organizing neural network for detecting novelties. In: **Proceedings of the 2007 ACM Symposium on Applied Computing**. [S.l.]: ACM, 2007. p. 462–466.

66 _____. A self-organizing neural network to approach novelty detection. In: CHIONG, R. (Ed.). **Intelligent Systems for Automated Learning and Adaptation**. [S.l.]: IGI Global, 2010. p. 49–71.

67 MONARD, M. C.; BARANAUKAS, J. A. Sistemas inteligentes: fundamentos e aplicações. In: _____. [S.l.]: Editora Manole, 2002. cap. Indução de regras e árvores de decisão, p. 115–139.

68 HAN, J.; KAMBER, M.; PEI, J. **Data Mining: Concepts and Techniques**. 3rd. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011. ISBN 0123814790, 9780123814791.

69 AGGARWAL, C. C.; HAN, J. (Ed.). **Frequent Pattern Mining**. [S.l.]: Springer, 2014.

70 AGRAWAL, R.; SRIKANT, R. Fast algorithms for mining association rules in large databases. In: **Proceedings of the 20th International Conference on Very Large Data Bases**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994. p. 487–499.

71 CARMONA, C. et al. Mefes: An evolutionary proposal for the detection of exceptions in subgroup discovery. an application to concentrating photovoltaic technology. **Knowledge-Based Systems**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 54, p. 73–85, 2013.

72 SUZUKI, E. Data mining methods for discovering interesting exceptions from an unsupervised table. **Journal of Universal Computer Science**, v. 12, n. 6, p. 627–653, 2006.

73 _____. Discovering interesting exception rules with rule pair. In: **In J. Fuernkranz (Ed.), Proceedings of the ECML/PKDD Workshop on Advances in Inductive Rule Learning**. [S.l.: s.n.], 2004. p. 163–178.

74 PRATI, R. C.; MONARD, M. C.; CARVALHO, A. C. P. L. F. d. Looking for exceptions on knowledge rules induced from HIV cleavage data set. **Genetics and Molecular Biology**, scielo, v. 27, p. 637 – 643, 2004.

75 _____. **A Method for Refining Knowledge Rules Using Exceptions**. 2004.

76 HUSSAIN, F. et al. Exception rule mining with a relative interestingness measure. In: **Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications**. London, UK, UK: Springer-Verlag, 2000. (PADKK '00), p. 86–97. ISBN 3-540-67382-2.

77 LIU, H. et al. Efficient search of reliable exceptions. In: _____. **Methodologies for Knowledge Discovery and Data Mining: Third Pacific-Asia Conference, PAKDD-99 Beijing, China, April 26–28, 1999 Proceedings**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999. p. 194–204.

78 PHAM, T. H. et al. Computational discovery of transcriptional regulatory rules. **Bioinformatics**, Oxford University Press, Oxford, UK, v. 21, n. 2, p. 101–107, jan. 2005. ISSN 1367-4803.

79 QUINLAN, J. R. **C4.5: programs for machine learning**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

80 _____. Induction of decision trees. **Machine Learning**, Kluwer Academic Publishers, Hingham, MA, USA, p. 81–106, 1986.

81 MILLINGTON, I. **Artificial Intelligence for Game**. [S.l.]: Morgan Kaufmann, 2006.

82 ZOBRIST, A. L. **A Hashing Method with Applications for Game Playing**. [S.l.], 1969.

83 STEVANOVIC, R. **Quantum Random Bit Generator Service**. 2015. Disponível em: <http://random.irb.hr/>.

84 STIPCEVIC, M.; ROGINA, B. M. Quantum random number generator based on photonic emission in semiconductors. **Review of Scientific Instruments**, v. 78, n. 4, 2007.

85 CAEXETA, G. S. **Visiondraughts - Um Sistema de Aprendizagem de Jogos de Damas Baseado em Redes Neurais, Diferenças Temporais, Algoritmos Eficientes de Busca em Árvores e Informações Perfeitas Contidas em Bases de Dados**. Dissertação — Faculdade de Computação - Universidade Federal de Uberlândia, 2008.

86 DUARTE, V. A. R. **MP-Draughts - Um Sistema Multiagente de Aprendizagem Automática para Damas Baseado em Redes Neurais de Kokohen e Perceptron Multicamadas**. Dissertação — Faculdade de Computação - Universidade Federal de Uberlândia, 2009.

87 BREUKER, D.; UITERWIJK, J.; HERIK, H. J. van den. **Replacement Schemes for Transposition Tables**. 1994. Disponível em: <citeseer.ist.psu.edu/112066.html>.

88 FREY, P. W. **Chess Skill in Man and Machine**. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1979. ISBN 0387079572.

89 PLAAT, A. et al. Best-first fixed-depth game-tree search in practice. In: **Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-95)**. Montreal, Canada: [s.n.], 1995. v. 1, p. 273–279. Disponível em: <citeseer.ist.psu.edu/plaat95bestfirst.html>.

90 SCHAEFFER, J. **One Jump Ahead: Computer Perfection at Checkers**. 2nd. ed. [S.l.]: Springer Publishing Company, Incorporated, 2009. ISBN 0387765751, 9780387765754.

91 FIERZ, M. C. **CheckerBoard**. 2016. Disponível em: <http://www.fierz.ch/checkerboard.php>.

92 GEISSER, S. **Predictive inference: an introduction**. New York: Chapman & Hall, 1993.

93 KOHAVI, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: **Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2**. [S.l.]: Morgan Kaufmann Publishers Inc., 1995. (IJCAI'95), p. 1137–1143. ISBN 1-55860-363-8.

94 HUBERT, L.; ARABIE, P. Comparing partitions. **Journal of Classification**, Springer-Verlag, v. 2, n. 1, p. 193–218, 1985.

95 ANALYSIS, S. I. S. **SISA**. 2016. Disponível em: <http://www.quantitativeskills.com/sisa/index.htm>.

96 NETO, H. C.; JULIA, R. M. S. Ls-draughts - a draughts learning system based on genetic algorithms, neural network and temporal differences. In: **IEEE Congress on Evolutionary Computation (CEC'07)**. [S.l.: s.n.], 2007. p. 2523–2529.

97 BURO, M. Improving heuristic mini-max search by supervised learning. **Artificial Intelligence**, p. 85–99, 2002.

98 PRATI, R. C. **Novas aborgagens em aprendizado de máquina para a geração de regras, classes desbalanceadas e ordenação de casos**. Tese (Doutorado) — USP - São Carlos, 2006.

99 HALL, M. et al. The weka data mining software: An update. **SIGKDD Explor. Newsl.**, ACM, New York, NY, USA, v. 11, n. 1, p. 10–18, 2009. ISSN 1931-0145.

100 FRANK, E.; WITTEN, I. H. Generating accurate rule sets without global optimization. In: SHAVLIK, J. (Ed.). **Fifteenth International Conference on Machine Learning**. [S.l.]: Morgan Kaufmann, 1998. p. 144–151.

101 LICHMAN, M. **UCI Machine Learning Repository**. 2013. Disponível em: <http://archive.ics.uci.edu/ml>.