Uma abordagem para melhorar o desempenho de agentes automáticos que operam em ambientes competitivos por meio de informações semânticas sobre mudanças de comportamento do oponente

Eldane Vieira Júnior

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Uberlândia
2022

Eldane Vieira Júnior

Uma abordagem para melhorar o desempenho de agentes automáticos que operam em ambientes competitivos por meio de informações semânticas sobre mudanças de comportamento do oponente

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: Ciência da Computação

Orientadora: Profª.Dra.Rita Maria da Silva Julia
Coorientadora: Profª.Dra. Elaine Ribeiro de Faria Paiva

Uberlândia
2022

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

V658
2022

Vieira Júnior, Eldane, 1984-
Uma abordagem para melhorar o desempenho de agentes automáticos que operam em ambientes competitivos por meio de informações semânticas sobre mudanças de comportamento do oponente. [recurso eletrônico] / Eldane Vieira Júnior. - 2022.

Orientadora: Rita Maria da Silva Julia.
Coorientadora: Elaine Ribeiro de Faria Paiva.
Tese (Doutorado) - Universidade Federal de Uberlândia, Pós-graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.te.2022.672
Inclui bibliografia.
Inclui ilustrações.

1. Computação. I. Julia, Rita Maria da Silva,1960-, (Orient.). II. Paiva, Elaine Ribeiro de Faria,1980-, (Coorient.). III. Universidade Federal de Uberlândia. Pós-graduação em Ciência da Computação. IV. Título.

CDU: 681.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:
Gizele Cristine Nunes do Couto - CRB6/2091
Nelson Marcos Ferreira - CRB6/3074

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Ciência da Computação
Av. João Naves de Ávila, 2121, Bloco 1A, Sala 243 - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4470 - www.ppgco.facom.ufu.br - cpgfacom@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Ciência da Computação				
Defesa de:	Tese de doutorado, 23/2022, PPGCO				
Data:	30 de Novembro de 2022	Hora de início:	09h20	Hora de encerramento:	12h55
Matrícula do Discente:	11713CCP003				
Nome do Discente:	Eldane Vieira Júnior				
Título do Trabalho:	Uma abordagem para melhorar o desempenho de agentes automáticos que operam em ambientes competitivos por meio de informações semânticas sobre mudanças de comportamento do oponente				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Inteligência Artificial				
Projeto de Pesquisa de vinculação:	-				

Reuniu-se, via videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Ciência da Computação, assim composta: Professores Doutores: Fabíola Souza Fernandes Pereira - FACOM/UFU; Marcelo Zanchetta do Nascimento - FACOM/UFU; Elaine Ribeiro de Faria Paiva - FACOM/UFU; André Carlos Ponce de Leon Ferreira de Carvalho - USP; Luiz Chaimowicz - UFMG e Rita Maria da Silva Julia -FACOM/UFU, orientadora do candidato.

O examinador André Carlos Ponce de Leon Ferreira de Carvalho participou desde a seguinte localidade: São Carlos/SP e o examinador Luiz Chaimowicz participou desde a seguinte localidade: Belo Horizonte/MG. Os demais membros e o discente participaram da cidade de Uberlândia/MG.

Iniciando os trabalhos a presidente da mesa, Profª Drª Rita Maria da Silva Julia, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao Discente a palavra para a exposição do seu trabalho. A duração da apresentação do Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir a senhora presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir o candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

Aprovado.

Esta defesa faz parte dos requisitos necessários à obtenção do título de Doutor.

O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Fabíola Souza Fernandes Pereira**, **Professor(a) do Magistério Superior**, em 01/12/2022, às 10:23, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Elaine Ribeiro de Faria Paiva**, **Professor(a) do Magistério Superior**, em 01/12/2022, às 10:48, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Rita Maria da Silva Julia**, **Professor(a) do Magistério Superior**, em 02/12/2022, às 14:48, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **André Carlos Ponce de Leon Ferreira de Carvalho**, **Usuário Externo**, em 07/12/2022, às 20:18, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Luiz Chaimowicz**, **Usuário Externo**, em 23/12/2022, às 18:05, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Marcelo Zanchetta do Nascimento**, **Professor(a) do Magistério Superior**, em 02/01/2023, às 09:32, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **4102979** e o código CRC **32CBD006**.

Referência: Processo nº 23117.088221/2022-11 SEI nº 4102979

A minha família e amigos.

Agradecimentos

Agradeço aos meus pais por todo o apoio que me oferecem na minha formação tanto profissional quanto como ser humano. Aos meus amigos e familiares que durante todo este percurso de formação acadêmica, tornaram momentos difíceis em momentos mais agradáveis. À Rosane Maria Maffei Vallim, autora do algoritmo M-DBScan, que esclareceu várias dúvidas, facilitando o início do desenvolvimento deste trabalho. Ao estatístico Cássio Alicântara pela ajuda que me foi dada, tornando possível uma melhor análise dos resultados gerados neste trabalho. Principalmente, agradeço à Professora Rita Maria da Silva Julia e à Professora Elaine Ribeiro de Faria Paiva por toda a paciência e confiança a mim destinadas ao longo da orientação deste Doutorado. Compartilhando comigo todo profissionalismo e experiência que transcendem todo o trabalho desenvolvido ao longo destes anos.

“Para ter algo que você nunca teve, é preciso fazer algo que você nunca fez.”
(Chico Xavier)

Resumo

O presente trabalho de Doutorado propõe e implementa uma inédita abordagem para aumentar o desempenho de agentes que operam em cenários competitivos que envolvem fluxo contínuo de dados por meio de informações relativas à dinâmica do comportamento de seus adversários. Para tanto, tais agentes devem ser dotados da habilidade de detectar, em tempo real, eventuais mudanças no comportamento de seus oponentes e, com base nessas informações, adaptar seus processos de tomada de decisão de forma a melhorar suas habilidades para lidar com os problemas para os quais foram concebidos. O jogo de *Real-Time Strategy* (RTS) StarCraft: BroodWar foi usado como estudo de caso. A abordagem proposta foi desenvolvida tendo como base as seguintes ações: 1) Extensão e aprimoramento do *Micro-clustering DBScan* (M-DBScan), que é um bem sucedido algoritmo de detecção de mudança de comportamento em cenários de fluxo contínuo de dados , de forma a permitir que ele aumente sua acurácia no processo de detecção de mudança de comportamento, bem como seja apto a associar, a cada um desses comportamentos, uma semântica que o represente; 2) Implementação de um agente jogador de StarCraft cujo módulo de tomada de decisão opere da seguinte maneira: as informações referentes aos significados do comportamento do adversário providas pelas versões estendidas do M-DBScan, aqui propostas, serão utilizadas como base para nortear o agente na execução de ações específicas que se moldem ao comportamento corrente do adversário. A abordagem proposta foi validada por meio de experimentos conduzidos de forma a avaliar as seguintes métricas: acurácia na detecção de mudanças de comportamento do oponente e na atribuição de significados a tais comportamentos; taxa de vitória do agente aqui proposto em torneios em que ele enfrenta diferentes versões do agente StarCraft. Os experimentos realizados corroboraram para o aumento na taxa de vitória do agente decorrente do uso tanto da informação semântica sobre a mudança de comportamento, quanto de um adequado conjunto de comportamentos, que são relevantes para o contexto do problema.

Palavras-chave: Detecção de novidades. Fluxo contínuo de dados. Semântica de Comportamento. Jogos RTS. StarCraft. Aprendizagem de máquina.

Abstract

This Ph.D. work proposes and implements an unprecedented approach to increase the performance of agents operating in competitive scenarios that involve a data stream of information related to the dynamics of their adversaries' behavior. Therefore, such agents must have the ability to detect, in real-time, eventual changes in the opponents' behavior and, based on this information, adapt their decision-making processes in order to improve their abilities to deal with problems for which were designed. The RTS StarCraft: BroodWar game was used as a case study. The proposed approach was developed based on the following actions: 1) Extension and improvement of M-DBScan, which is a successful behavior change detection algorithm for data stream scenarios, in order to increase its accuracy of detecting behavior change, as well as being able to associate to each of these behaviors, a semantic that represents it; 2) Implementation of a StarCraft player agent whose decision-making module operates as follows: the information regarding the meanings of the opponent's behavior provided by the extended versions of M-DBScan, proposed here, will be used to guide the agent in the execution of adequate actions considering the current opponent behavior. The proposed approach was validated through experiments conducted in order to evaluate the following metrics: accuracy in detecting changes in the opponent's behavior and in assigning meanings to such behaviors; the win rate of the developed agent in tournaments where it faces different opponents in the game Starcraft. With the experiments carried out, it was possible to demonstrate the gain in the agent's winning rate, due to the use of semantic information about the behavior change and the use of an adequate set of behaviors, which are relevant to the context of the problem.

Keywords: Novelty detection. Data Stream. Behavior Semantics. RTS games. StarCraft. Machine learning.

Lista de ilustrações

Lista de tabelas

Lista de siglas

ADWIN *Adaptive Window Algorithm*

ASP *Answer Set Programming*

AMOC *Activity Monitor Operating Characteristic*

BiCNet *Multiagent Bidirectionally-Coordinated Network*

BWAPI *Brood War Aplication Programming Interface*

CM Cadeia de Markov

COEP *Continual Online Evolutionary Planning*

CUSUM *Cumulative Sum approach*

DBN *Deep Belief Network*

DBSCAN *Density-Based Spatial Clustering of Applications with Noise*

DDM *Drift Detection Method*

EDDM *Early Drift Detection Method*

FCA *Formal Concepts Analysis*

HP *Hit points*

IA Inteligência Artificial

KNN *K-Nearest Neighbors*

M-DBScan *Micro-clustering DBScan*

MINAS *Multi-class Learning Algorithm for Data Stream*

MLP *Multilayer Perceptron*

NNge *Non-Nested generalized exemplars*

RTS *Real-Time Strategy*

ROC *Receiver Operating Characteristic*

SAND *Semi-Supervised Adptative Novel Class Detection and Classification over Data Stream*

SDS-MDBScan *Statistically Defined Semantic-MDBScan*

SEA *Streaming Ensemble Algorithm*

Semantic-MDBScan *Semantic Micro-Clustering DBScan*

SOM *Kohonem Self-organized map*

WEKA *Waikato Environment for Knowledge Analysis*

XAI *Explainable Artificial Intelligence*

Lista de algoritmos

Sumário

CAPÍTULO 1

Introdução

A fim de lidar com a crescente complexidade de problemas que envolvem diferentes setores cruciais da sociedade moderna, como a medicina, sistema financeiro, trânsito nas cidades, dentre outros, ficou evidente a necessidade de se contar com sistemas computacionais mais sofisticados, providos de alguma autonomia, e que requeiram cada vez menos a intervenção humana e a dependência de especialistas (FACELI et al., 2011; AMARAL, 2016b). Dessa maneira, o aprendizado de máquina assumiu o importante papel de criação de ferramentas computacionais capazes de resolver problemas de maneira eficiente e autônoma, a partir de um histórico de dados que refletem experiências passadas na resolução de tais problemas (FACELI et al., 2011).

O cotidiano das pessoas está repleto de problemas que podem ser resolvidos por meio de aprendizado de máquina, especialmente pela disponibilidade da matéria-prima necessária para o processo de aprendizagem, que é o dado. Dentre as importantes fontes de dados disponíveis, destacam-se as seguintes: o compartilhamento de informações pessoais, o uso de serviços disponibilizados na Internet, a presença de sensores em equipamentos como celulares, bem como o entretenimento e a comunicação via aplicativos (ATZORI; IERA; MORABITO, 2010).

O uso de técnicas de aprendizado de máquina para a análise de um grande volume de dados tem sido o foco de diversas pesquisas (CHEN; MAO; LIU, 2014), pois podem trazer benefícios, sob o ponto de vista competitivo de uma empresa, possibilitando que decisões de negócios sejam tomadas com base em conhecimentos presentes nestes dados, ainda que implicitamente (KRAWCZYK et al., 2017).

Como exemplos de áreas que se destacam pelo uso de técnicas de análise de dados, podem ser citadas: Marketing, na descoberta de consumidores que podem atrair outros clientes; Finanças, para prever o desempenho financeiro de uma empresa; Recursos humanos, na identificação do perfil do funcionário que pode abandonar o emprego; Jogos digitais, na modelagem do comportamento do jogador, o que possibilita tornar o jogo mais interessante e desafiador (AMARAL, 2016a; VALLIM, 2013).

Dentre os fatores que impactam na definição das características dos agentes que vão lidar com tais problemas, destaca-se a natureza do ambiente no qual vão atuar.

No caso dos ambientes estáticos, os agentes, uma vez treinados, executarão as mesmas

ações sempre que se depararem com um mesmo cenário do ambiente. Para o treinamento desses agentes, o uso de bases de dados estáticas (treinamento em *batch*) é adequado, visto que as bases a serem usadas neste caso não precisam refletir as mudanças do ambiente (RUSSELL; NORVIG, 2016).

Por outro lado, no cenário de problemas que apresentam ambiente dinâmico, ou seja, em que novas situações podem surgir ao longo do tempo, agentes treinados em cenário estático não estariam aptos a efetuar boas escolhas de ações diante de situações inéditas, visto que essas não foram contempladas durante o treinamento desses agentes. É importante destacar que problemas cujos ambientes envolvem oponentes que tentam minimizar a chance de sucesso dos agentes, frequentemente, representam cenários dinâmicos, uma vez que mudanças no *modus operandi* dos adversários podem ocasionar variadas, inéditas e imprevisíveis situações. Exemplo relevante desses problemas são os jogos digitais, em que a capacidade dos agentes de adaptar o seu processo de tomada de decisão ao perfil do oponente é crucial para o sucesso deles (NETO; JULIA, 2018; NETO, 2016).

Um caso relevante de cenário dinâmico é aquele no qual os dados são obtidos na forma de um fluxo contínuo. Esse cenário incorpora características que não são comuns em um cenário estático, tais como: os dados chegam em tempo real; o sistema não tem qualquer controle sobre a ordem de chegada dos dados; desconhecimento da quantidade de dados que chegará pelo fluxo, bem como do instante de chegada dos mesmos; o fluxo é passível de mudanças relativas à distribuição dos dados; alta frequência de chegada de dados via fluxo (GAMA, 2010; KRAWCZYK et al., 2017; BABCOCK et al., 2002). Em função dessas características, os algoritmos que lidam com fluxo contínuo precisam, por exemplo, descartar os dados já tratados tão logo concluído o processamento deles, visto que a memória é um recurso limitado. Além disso, estes algoritmos precisam ser rápidos para lidar com os limites rígidos de tempo inerentes ao processamento em tempo-real (GAMA, 2010; KRAWCZYK et al., 2017; BABCOCK et al., 2002; MUTHUKRISHNAN et al., 2005; KRAWCZYK et al., 2017).

Em virtude das características existentes nos problemas que envolvem fluxo contínuo, as técnicas que lidam com eles devem estar aptas a executar um processo de aprendizagem incremental, ou seja, o aprendizado deve refletir as informações relevantes dos dados que chegam via fluxo, contando com um tempo de processamento limitado para atender a este requisito. Isso define um cenário complexo e dinâmico, em que certas informações relativas aos dados, ao longo do tempo, podem confirmar-se como relevantes, ao passo que, outras, podem tornar-se inócuas.

Dentre as abordagens concebidas para operar com o desafio associado a tais problemas, destaca-se o Micro-Clustering DBScan (M-DBScan) (VALLIM, 2013), que é um algoritmo concebido para detectar mudanças na distribuição dos dados em fluxos contínuo, sem, contudo, estar apto a identificar o significado prático de tais mudanças, o qual representaria uma relevante informação norteadora para o processo de tomada de decisão de agentes criados com o objetivo de lidar com tal desafio.

Os jogos eletrônicos se destacam entre os problemas que envolvem fluxo contínuo de dados. De fato, neles os dados são gerados ao longo das interações (ou disputas) entre o jogador (oponente) e o motor do jogo. Os dados desse fluxo representam, assim, os sucessivos cenários (ou ambientes) de jogo que retratam tais interações. Dessa forma, a análise desses dados permite, por exemplo, mapear as diversas - e frequentemente imprevisíveis - variações no comportamento do oponente em decorrência dos desafios que lhe são impostos pelo motor do jogo ao longo de uma partida. Saliente-se que, nesses jogos, a elevada quantidade de eventos possíveis torna impraticável a predição do volume de dados que será gerado ao longo de uma partida.

Assim sendo, o modelo gerado a partir desses dados deve passar por constantes adaptações, de modo a refletir o cenário corrente do jogo, sendo que isso deve ser feito de modo ágil, de forma a não permitir que o sistema perca informações sobre um determinado dado em virtude de não ter tido tempo de concluir seu processamento antes da chegada do dado subsequente.

É importante ressaltar que essas características dos jogos eletrônicos os tornam muito semelhantes a vários problemas do mundo real que também lidam com desafios como a imprevisibilidade, a existência de eventos contrários aos seus interesses, a necessidade de adaptações em decorrência de mudanças no ambiente, entre outras (SCHAEFFER; HERIK, 2002). Além disso, a indústria de jogos eletrônicos tem-se destacado por apresentar um significante crescimento, vinculado a importantes avanços, tais como o aumento do poder computacional dos hardwares e os avanços gráficos que aprimoram a experiência do jogador (JOHNSON; WILES, 2001).

Todos esses fatos vêm fazendo com que os jogos eletrônicos, de forma acentuadamente crescente, tornem-se um rico "laboratório"de estudos nas pesquisas em Inteligência Artificial (IA), o que vem proporcionando conquistas muito significativas na área, destacando-se, dentre elas, a vitória do agente jogador de Go (AlphaGo) sobre o humano campeão mundial do jogo na época (SILVER et al., 2017).

Considerando os fatos apresentados, o presente trabalho de Doutorado propõe e implementa uma inédita abordagem para aumentar o desempenho de agentes que operam em cenários competitivos, por meio de informações relativas à dinâmica do comportamento de seus adversários. Para tanto, tais agentes devem ser dotados da habilidade de detectar, em tempo real, eventuais mudanças no comportamento de seus oponentes e, com base nessas informações, adaptar seus processos de tomada de decisão de forma a melhorar suas habilidades para lidar com os problemas para os quais foram concebidos. O jogo de *Real-Time Strategy* (RTS) StarCraft: BroodWar foi usado como estudo de caso. Tal jogo foi desenvolvido pela empresa *Blizzard Entertaiment*, possuindo várias características em comum com outros jogos da categoria RTS, tal como o fato de os jogadores precisarem gerenciar ações, extrair recursos do meio, construir bases com os recursos disponíveis, além de produzir e aprimorar os membros de seus exércitos, que são normalmente chamados de unidades.

A abordagem proposta foi desenvolvida tendo como base as seguintes ações principais: 1) Extensão e aprimoramento do M-DBScan, de forma a permitir que ele aumente sua acurácia

no processo de detecção de mudança de comportamento do oponente, bem como seja apto a identificar um significado prático que possa ser adequadamente atribuído a cada um desses comportamentos; 2) Implementação de um agente jogador de StarCraft cujo módulo de tomada de decisão opera da seguinte maneira: as informações referentes aos significados do comportamento do adversário, providas pelas versões estendidas do M-DBScan, aqui propostas, serão utilizadas como base para nortear o agente na execução de ações específicas que se moldem ao comportamento corrente do adversário. Tal agente foi implementado a partir dos recursos disponíveis na API *Brood War Aplication Programming Interface* (BWAPI) (TEAM, 2015). No caso, o módulo de tomada de decisão do agente proposto estende o motor de regras de tomadas de decisão padrão do jogo, por meio da inclusão de novas regras que especificam como o agente deve agir em situações em que as versões estendidas do M-DBScan aqui propostas detectam uma mudança de comportamento do oponente, bem como identificam o significado do novo comportamento.

A validação desta proposta foi efetuada por meio de experimentos conduzidos de forma a avaliar as seguintes métricas: acurácia na detecção de mudanças de comportamento do oponente e na atribuição de significados a tais comportamentos; taxa de vitória do agente aqui proposto em torneios em que ele enfrenta o agente UAlbertaBot (CHURCHILL; BURO, 2011; CHURCHILL; SAFFIDINE; BURO, 2012) e outras versões do agente desenvolvidas ao longo da pesquisa, também construído a partir dos recursos disponíveis na BWAPI, mas que opera com o conjunto de regras de tomada de decisão padrão da biblioteca .

Investigou-se também neste trabalho o impacto da adoção de duas estratégias distintas de representação dos dados (que correspondem aos cenários de jogo): uma primeira, em que um mesmo conjunto de atributos foi usado para representar os dados ao longo de toda uma partida; e uma segunda, em que diferentes conjuntos de atributos foram selecionados para representar os dados em diferentes fases do jogo (no caso, cada conjunto reflete características inerentes a uma dada fase).

O desenvolvimento desta pesquisa foi conduzido em etapas ordenadas voltadas ao cumprimento dos objetivos específicos apresentados na Seção 1.2.

1.1 Motivação

Cenários extremamente dinâmicos, como o observado em um jogo RTS, refletem muito bem uma grande variedade de problemas reais. Dessa forma, desenvolver um sistema com capacidade de identificar novidades no contexto dos jogos e de se adaptar a elas, representa um importante avanço no desenvolvimento de alternativas de soluções que podem ser aplicadas a problemas que atingem o dia-a-dia das pessoas. Tal fato corrobora a decisão da escolha do jogo StarCraft - em cenário de fluxo contínuo - como estudo de caso no presente trabalho, uma vez que esse ambiente remete a situações de problemas do mundo real que envolvem gerenciamento de recursos, ágeis tomadas de decisão em função de eventos momentâneos, formação de grupos

com base na habilidade de seus membros (o que equivale à construção de um exército), etc.

O algoritmo baseado em Cadeia de Markov (CM) chamado M-DBScan foi escolhido como base para esta pesquisa, em virtude de sua capacidade de processar o dado em tempo real e gerar um modelo incrementalmente, servindo para prover informações sobre alguma mudança significativa sobre os dados, que pode ser interpretada como uma mudança de comportamento dependendo do problema em que se aplica o algoritmo. Esse algoritmo tem sido usado em pesquisas relacionadas a detecção de mudanças no cenário de fluxo contínuo de dados (VALLIM et al., 2013), sendo validado com o uso de dados artificiais e dados gerados a partir de alguns jogos em situações controladas artificialmente. Neta pesquisa com M-DBScan utiliza-se somente um conjunto de atributos para representar o elemento sobre o qual deseja-se identificar mudanças, não adequando essa representação a evolução de diferentes situações refletidas sobre os dados. Considerando o cenário dinâmico dos jogos, os atributos podem ter diferentes níveis de relevância em diferentes momentos da partida, por isso, na presente pesquisa utiliza-se diferentes conjuntos de atributos considerando o momento do jogo. Com esta abordagem, utilizando uma melhor e mais relevante representação do jogo via atributos, espera-se um aumento na precisão nas identificações de mudanças de comportamento. Um impacto dessa abordagem será o uso de uma CM para cada momento considerado no jogo.

Em virtude da dinâmica existente no cenário de jogos, o agente jogador pode adaptar sua estratégia de acordo com o que é observado na partida, indicando uma necessidade do agente em perceber os eventos relevantes para uma eventualmente adaptação estratégica. Dentre esses eventos relevantes, destaca-se o estilo de jogo do adversário enfrentado pelo agente, visto que uma mudança no seu comportamento, que reflete em seu estilo de jogo, é um critério suficiente para que o agente adapte as suas ações conforme a nova estratégia do adversário. Dessa maneira, além da necessidade de conhecer o momento da ocorrência de mudança de comportamento, o agente precisa ter a informação sobre o que essa mudança representa semanticamente, de tal modo que as suas ações serão adaptadas em função desse novo cenário de jogo. Diante disso, uma importante motivação para essa pesquisa é a carência de trabalhos que fornecem junto a identificação de mudança de comportamento, uma informação sobre o que essa mudança significa, e como essa informação pode ser usada de modo a proporcionar ganhos. Com o jogo StarCraft será possível desenvolver e testar essa abordagem que fornece um significado para a mudança de comportamento do adversário, e fazer uso dessa informação na tomada de decisão no jogo.

No jogo StarCraft há um desafiador fator de incerteza, que se deve ao fato de o jogador não ter informações sobre aquilo que não faz parte do seu campo de visão no mapa do jogo. Nesse sentido, a presente pesquisa fará uso das informações obtidas durante os confrontos com o adversário, por este ser um momento em que é possível obter muita informação sobre o mesmo. Tentar obter informações sobre o adversário fora do momento de confronto pode gerar lacunas ou informações imprecisas, uma vez que o adversário pode estar produzindo unidades em diferentes pontos do mapa e alguns desses pontos podem estar fora do campo de visão das

unidades do agente. Algo que não pode ser feito, por exemplo, é acessar certos recursos do motor do jogo para obter informações sobre o adversário que não estariam disponíveis para um jogador humano, logo seria uma forma irregular de obter dados.

Pela extração de dados sobre o adversário ocorrer, principalmente, no momento do confronto, as alterações no módulo de decisão do agente também ocorrerão sobre as decisões relacionadas às batalhas.

Como mais uma motivação destaca-se a importância da pesquisa apresentada nesta tese no desenvolvimento de soluções que podem ser aplicadas em problemas de diferentes áreas, contribuindo para integrar a pesquisa científica à resolução de questões práticas presentes no cotidiano das pessoas.

1.2 Objetivos e Desafios da Pesquisa

Os objetivos desta pesquisa concentram-se na área de aprendizado de máquina em cenário de fluxo contínuo de dados, usando como estudo de caso os agentes jogadores. Nesta direção, o objetivo geral aqui perseguido consiste no seguinte:

- ❏ Aprimorar o processo de tomada de decisão em tempo real, por meio do desenvolvimento de um módulo semântico que atribui um significado às mudança de comportamento identificadas, tornando possível tomar decisões contextualizadas. Tal módulo pode ser aplicado a diferentes problemas, contudo, será testado em um agente que atua no cenário do jogo de RTS StarCraft, possibilitando que suas tomadas de decisão funcionem em conformidade com o que a mudança de comportamento representa.

Os objetivos específicos a serem cumpridos com o intuito de atingir este objetivo global são dispostos a seguir:

1. Aplicar o algoritmo M-DBScan a dados obtidos a partir do jogo StarCraft, analisando a viabilidade do uso desse algoritmo na identificação de mudanças no comportamento do adversário com base em informações que retratam os cenários de jogo e, consequentemente, o comportamento de todos os jogadores envolvidos no confronto (a ser cumprido na *Etapa 1* apresentada no Capítulo 4);

2. Utilizar diferentes conjuntos de atributos para representar, adequadamente, diferentes momentos do jogo, buscando um aumento na acurácia do M-DBScan quanto à identificação de mudanças de comportamento (a ser cumprido na *Etapa 2* apresentada no Capítulo 5).

3. Desenvolver duas versões modificadas do M-DBScan, denominadas *Semantic Micro-Clustering DBScan* (Semantic-MDBScan) e *Statistically Defined Semantic-MDBScan* (SDS-MDBScan), capazes de prover uma semântica às mudanças de comportamento detectadas em cenário de fluxo contínuo de dados. No caso, o SDS-MDBScan será um aprimoramento do Semantic-MDBScan, uma vez que, além de estar apto a prover um significado

às mudanças de comportamento, deverá também contar com um processo de detecção de mudanças com maior acurácia do que a do Semantic-MDBScan (o qual usa o mesmo processo do M-DBScan) (a ser cumprido na *Etapa 3* apresentada no Capítulo 6).

4. Implementar um agente StarCraft cuja arquitetura inclua o SDS-MDBScan e um motor de tomada de decisão que corresponde a uma extensão do motor padrão do StarCraft. Nesse motor, novas regras relacionadas ao comportamento do oponente são adicionadas. No caso, o significado atribuído pelo SDS-MDBScan às mudanças detectadas no comportamento do oponente são usadas como informação adicional para escolha do conjunto de ações a serem executadas no cenário corrente de jogo (conjunto, este, definido pelas regras do motor de decisão). Além disso, com base nos resultados observados no cumprimento do segundo objetivo específico apresentado, pretende-se investigar o impacto no desempenho do agente provocado pelo uso de diferentes conjuntos de atributos em diferentes estágios de jogo - onde cada conjunto conterá atributos mais adequados às fases de jogo envolvidas na análise - (a ser cumprido na *Etapa 4* apresentada no Capítulo 7).

1.3 Hipóteses

A possibilidade de um sistema aprender com os dados que chegam em tempo real no cenário de fluxo contínuo de dados, pode contribuir muito para otimizar as tarefas ligadas ao aprendizado de um agente automático que opera em ambiente competitivo. A partir disso, as seguintes hipóteses foram definidas:

- ❑ O M-DBScan, quando usado com atributos adequados, é bastante apropriado para detectar mudanças de comportamento do oponente em um jogo de StarCraft;

- ❑ À medida em que uma partida de um jogo RTS evolui, alguns atributos podem se tornar mais relevantes, ou até mesmo perder a sua relevância. Assim sendo, o uso de diferentes conjuntos de atributos para representar fases distintas de evolução do ambiente, proporcionará um aumento na precisão da identificação de mudanças de comportamento pelo algoritmo M-DBScan, em virtude de uma representação mais adequada.

- ❑ A acurácia do M-DBScan pode ser aumentada por meio da criação de abordagens alternativas de detecção de mudanças que expandam o referido algoritmo nos seguintes aspectos: habilidade para identificar o significado das mudanças detectadas; uso de um novo critério estatístico, baseado na reincidência de um mesmo significado em uma sequência de novidades, como lastro para apontar novas mudanças de comportamento;

- ❑ Um agente cujo motor de tomada de decisão conta com um módulo capaz de identificar mudanças de comportamento, atribuir uma semântica a elas e usar tal significado para norteá-lo na escolha de suas ações, tem um melhor desempenho.

1.4 Contribuições Científicas

As principais contribuições oriundas deste trabalho são listadas a seguir:

- ❏ Realização de análises experimentais que comprovaram a eficácia do M-DBScan como detector de mudanças de comportamento do oponente em jogos reais de StarCraft, bem como a conveniência de se usarem diferentes conjuntos de atributos, adaptados a fases distintas desse jogo, como estratégia para melhorar o desempenho desse algoritmo;
- ❏ Produção do Semantic-MDBScan e do SDS-MDBScan, que correspondem a expansões do M-DBScan. Ambos efetuam tal expansão pela habilidade em atribuir um significado às mudanças encontradas. No caso do SDS-MDBScan, além dessa habilidade, ele conta com um novo critério estatístico de detecção de mudanças, baseado na reincidência de um mesmo significado em uma sequência de novidades, que também contribui para aumentar a acurácia do M-DBScan. Salienta-se que tais algoritmos representam abordagens genéricas que podem ser aplicados a variados cenários de fluxo contínuo de dados diferentes de jogos;
- ❏ Proposta e implementação do agente SDS-StarCraft, cujo motor de tomada de decisão conta com um módulo capaz de identificar mudanças de comportamento, atribuir uma semântica a elas e usar tal significado para norteá-lo na escolha de suas ações, de modo que a reação do agente seja compatível com as mudanças detectadas. É importante destacar que tal estratégia também pode ser aplicada a outros tipos de problemas (ou jogos) em que um oponente tenta minimizar a chance de sucesso do agente.

1.5 Produção Bibliográfica

O desenvolvimento desta pesquisa gerou uma série de produções bibliográficas, as quais estão listadas a seguir:

- ❏ O artigo "*Mining Data Stream to Detect Behavior Change in a Real-Time Strategy Game*" publicado na conferência *Machine Learning Data Mining* (MLDM) em 2019, classificada com o índice restrito B1 pela CAPES, considerando o padrão de classificação na época da publicação. Este artigo retrata a viabilidade do uso do M-DBScan no cenário de jogos RTS como o StarCraft;
- ❏ O artigo "*Adapting the Markov Chain based Algorithm M-DBScan to Detect Opponents' Strategy Changes in the Dynamic Scenario of a StarCraft Player Agent*", publicado na conferência *International Conference on Agents and Artificial Intelligence* (ICAART) - 2020, classificada com o índice restrito B1 pela CAPES, considerando o padrão de classificação na época da publicação. Este artigo descreve o uso adaptável de diferentes conjuntos de atributos para representar diferentes estágios do jogo StarCraft.

- O artigo "*Semantic-MDBScan: an Approach to Assign a Semantic Interpretation to Behavior Changes in Data Stream Scenarios*", aceito na conferência *International Conference on Information Technology : New Generations* (ITNG) - 2022, classificada com índice restrito A4. Este artigo retrata o desenvolvimento da abordagem Semantic-MDBScan na tarefa de atribuir uma semântica à mudança de comportamento.

- Artigo de Periódico: "*SDS-MDBScan: Assigning a Meaning to Changes in Data Stream Scenarios Based on the Statistical Calculation of the Data Semantic Trends*" - tal artigo apresenta os resultados das pesquisas conduzidas no Capítulo 6, e está em fase de submissão.

- Artigo de Periódico: "*A new approach to enhance the decision-making of agents operating in dynamic contest environments through semantic information about the opponent's behavior changes*" - tal artigo apresenta os resultados das pesquisas conduzidas no Capítulo 7, e está em fase de submissão.

Além dos resultados em publicações, foram produzidos vários códigos e bases de dados ao longo do desenvolvimento deste trabalho. Os códigos estão disponíveis no link <https://urlzs.com/tzxCE>, enquanto que as bases estão disponíveis no link <https://urlzs.com/YdQrj>.

1.6 Organização da tese

A presente tese de Doutorado está organizada da seguinte maneira: o Capítulo 2 descreve os conceitos teóricos básicos desta pesquisa; o Capítulo 3 descreve os trabalhos correlatos; no Capítulo 4 é apresentada a execução da *Etapa 1* da pesquisa, referente ao cumprimento do primeiro objetivo específico apontado na Seção 1.2, bem como os experimentos realizados; no Capítulo 5 é descrita a implementação da *Etapa 2*, a qual se refere à execução do segundo objetivo específico descrito na Seção 1.2, alem dos experimentos realizados; o Capítulo 6 descreve o desenvolvimento da *Etapa 3*, que se refere ao terceiro objetivo específico da Seção 1.2, bem como os experimentos realizados; no Capítulo 7 é descrita a *Etapa 4* da pesquisa, a qual se refere ao quarto objetivo específico apresentado na Seção 1.2, e seus experimentos correspondentes; por fim, no Capítulo 8 é apresentada a conclusão, listando as contribuições alcançadas com a pesquisa.

CAPÍTULO 2

Fundamentação Teórica

Neste capítulo são apresentados os fundamentos teóricos necessários para compreender o desenvolvimento da pesquisa descrita neste trabalho. A primeira seção desse capítulo apresenta detalhes do jogo StarCraft, que é o ambiente usado nos experimentos para validar essa pesquisa; em seguida, na Seção 2.2 são apresentados conceitos e formas de aplicação da modelagem de jogador, que é um ponto importante na pesquisa, visto que o adversário no StarCraft será modelado a partir de atributos, com o intuito de identificar mudanças em seu comportamento; na Seção 2.4 é fundamentada a detecção de mudanças no cenário de fluxo contínuo de dados; na Seção 2.5 é apresentado o algoritmo M-DBScan, escolhido nessa pesquisa para lidar com o processamento do dado que chega via fluxo contínuo, e para indicar mudanças presentes nesses dados. Esse algoritmo é a base deste trabalho, e a partir de modificações que visam incorporar novas funcionalidades ao M-DBScan, que serão atingidos os principais objetivos desta pesquisa. Na Seção 2.5 também são apresentados conceitos e detalhes de funcionamento do processo de detecção de novidades e mudanças no cenário de fluxo contínuo de dados.

Com o que é apresentado neste capítulo é possível compreender as principais bases dessa pesquisa que são: o cenário de estudo e teste, que é o jogo StarCraft; o processo de modelagem de jogador, que é crucial para a representação do adversário no jogo, e para que ocorra a identificação de mudanças em seu comportamento; e por fim, o algoritmo M-DBScan que é a base dessa pesquisa para processar o dado que chega via fluxo contínuo, com o intuito de detectar mudanças de comportamento e com o desenvolvimento deste trabalho também terá a capacidade de prover uma semântica às mudanças de comportamento identificadas.

2.1 Jogo StarCraft e sua API

StarCraft é um jogo RTS produzido pela *Blizzard Entertainment* em 1998 (ENTERTAINMENT, 1998), cujo pacote de expansão StarCraft: BroodWar foi lançado pouco tempo depois no mesmo ano, e é um jogo com grande popularidade entre pesquisas de IA, e foi escolhido para ser usado neste trabalho. Então, nesta seção será descrito o jogo StarCraft, além de sua interface de desenvolvimento *Brood War Application Programming Interface* (BWAPI), que permite

o desenvolvimento de um agente que vai atuar sobre o motor do jogo StarCraft. A Figura 1 é apresentada para ilustrar uma parte do ambiente do jogo.

Figura 1 – Ambiente do jogo StarCraft.

Fonte: Jogo StarCraft com pacote de expansão BroodWar

No jogo StarCraft, os jogadores podem controlar uma das três raças: *Terran*, *Protons* e *Zerg*. Cada raça possui suas próprias características que indicam suas forças e fraquezas. Existindo também diferenças entre os tipos de unidades de uma mesma raça, que compõem o exército do jogador, por exemplo, existem as unidades cuja principal função são tarefas de unidades trabalhadores, que vão extrair recursos do mapa ou construir estruturas, enquanto outras unidades são mais apropriadas para o combate.

O jogo StarCraft possui um desafio peculiar que consiste no fato de que nenhum jogador conhece o que existe em regiões ainda não exploradas do mapa, o que caracteriza um problema para a definição de uma estratégia, por ter um ambiente desconhecido ou parcialmente conhecido. Além disso, todas os desafios referentes ao cenário de fluxo contínuo de dados, estão presentes em um jogo como o StarCraft. Então, as amostras de dado obtidas do jogo StarCraft são geradas em uma frequência alta, refletindo a interação entre o jogo e o jogador. Nesta inte-

ração, o jogador pode ter, inicialmente, um tipo de comportamento, sendo, por exemplo, mais defensivo, e com o tempo, este comportamento pode se tornar mais agressivo, dependendo dos eventos correntes do jogo. Logo, um bom jogador deve saber como reagir a certos eventos, e isto implica em uma eventual mudança nos estilo de jogo.

Todos estes desafios a cerca de um jogo de estratégia em tempo real como o StarCraft, exigem um sistema adequado para, dinamicamente, verificar as alterações nas distribuições dos dados, uma vez que essas alterações são geradas, por exemplo, pela mudança de estratégia do adversário. Neste caso, o agente jogador deve estar apto a adequar as suas ações de acordo com tais alterações. Salienta-se que as mudanças de comportamento são passíveis de serem observadas a partir das mudanças nos cenários de jogo, os quais podem ser representados por meio de atributos específicos relacionados ao jogo.

O processo de tomada de decisões no StarCraft pode ser dividido em tarefas de *microgerenciamento* e *macrogerenciamento*. Sendo que o microgerenciamento engloba o controle tático e individual das unidades, enquanto o macrogerenciamento engloba o planejamento estratégico do que deve ser construído e a ordem em que as construções devem ocorrer. Esta pesquisa ficará focada no microgerenciamento.

Para o desenvolvimento de um agente que atua sobre o jogo StarCraft, é preciso uma forma de interação com o motor do jogo. Conforme mencionado, existe a interface de desenvolvimento conhecida como BWAPI, que permite, por exemplo, controlar comandos do jogo tais como construção de um exercício e ataque ao adversário. O motor do jogo StarCraft é, essencialmente, baseado em um sistema de regras produzido a partir de um dos seguintes três tipos principais de comando: i) locomoção: envolve comandos tais como movimentação de unidades pelo mapa buscando por novos recursos, ou locomoção de unidades até um certo ponto (que pode ser um alvo para ataque ou mesmo uma fonte de minerais a ser explorada); ii) construção: refere-se a comandos que definem como produzir novas unidades, como construir novas estruturas, ou como alocar unidades para executarem certas tarefas; iii) gerenciamento de combate: definem as tropas que entrarão em confronto. Para tanto, é preciso, continuamente, monitorar informações (como a existência de unidades inimigas dentro do raio de visão de suas unidades), realizando atribuições de comandos de combates ou renovando tais atribuições caso as anteriores já tenham sido concluídas. O Algoritmo 1 (CHURCHILL, 2016) apresenta o uso de regras para o gerenciamento de combate do jogo.

O BWAPI tem sido usado no desenvolvimento de diversos agentes em pesquisas que tem como cenário o jogo StarCraft. Um exemplo destes agentes é o UAlbertaBot (CHURCHILL; BURO, 2011; CHURCHILL; SAFFIDINE; BURO, 2012), que é a base para o desenvolvimento deste trabalho, que junto com a BWAPI, será possível desenvolver importantes etapas desta pesquisa.

Na Figura 2 é apresentada uma breve arquitetura do UAlbertaBot (CHURCHILL, 2020), focando no módulo *Game Commander*, visto que este trabalho atuará, essencialmente, nos comportamentos relacionados a combate. Nesta figura pode ser observada a existência de diferentes

Algoritmo 1 Regras de combate

```
1: if Existir unidades de combate disponíveis then
2:     if Conhece a localização de uma base inimiga then
3:         return Ataque à base
4:     else
5:         if Há unidades inimigas visíveis then
6:             return Ataque às unidades inimigas visíveis
7:         else
8:             if Conhece a localização de uma construção inimiga then
9:                 return Ataque à construção inimiga com localização conhecida
10:            else
11:                return Explore o mapa até encontrar um alvo para ataque
12:            end if
13:        end if
14:    end if
15: end if
```

sub-módulos dentro do módulo *Game Commander*, como os apresentados a seguir:

- O *Production Manager* lida com a construção de novas estruturas;
- O *Information Manager* mantém atualizadas as informações como a posição de unidades e a localização de unidades inimigas que foram detectadas;
- O *Worker Manager* lida com as unidades trabalhadoras, atribuindo tarefas como a extração de minerais ou gases de alguma localização no mapa do jogo;
- O *Strategy Manager* toma decisões como a ordem em que as construções são realizadas, quando deve ocorrer alguma expansão no mapa do jogo e outras decisões sobre as dinâmicas do jogo;
- O *MapTools* encontra o caminho a ser percorrido pelas unidades no mapa e armazena esta informação para um uso posterior;
- O *MapGrid*, para cada espaço ocupado no *grid*, mantém atualizado o momento da última visita de uma unidade naquele espaço;
- O *Combat Commander* gerencia os esquadrões para realizar ataques.

Como o foco desta pesquisa é detectar mudanças no comportamento do adversário quanto a questões de combate, então, na Figura 2 o módulo *Combat Commander* é mais detalhado, mas todos os demais módulos possuem mais conteúdos a serem explorados.

Os sub-módulos do *Combat Commander* são os seguintes:

- O *Squad* é responsável por executar os comandos do esquadrão como atacar um alvo ou fazer o esquadrão retornar para a base, este sub-módulo é composto pelo *Micro Manager* que é formado por três categorias de gerenciamento:

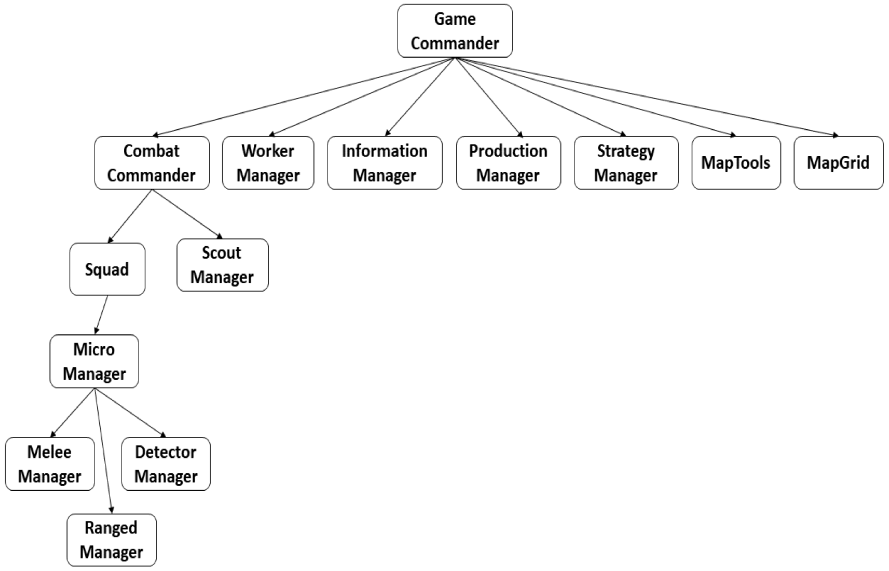


Figura 2 – Amostra da arquitetura do UAlbertaBot.

  - O *Melee Manager* lida com as unidades que podem atacar as unidades inimigas próximas, para um combate corpo-a-corpo.
  - O *Ranged Manager* lida com as unidades que podem atacar as unidades inimigas que estão longe.
  - O *Detector Manager* lida com as unidades ou estruturas capazes de detectar e revelar a existência de unidades inimigas que estão escondidas ou camufladas.

- O *Scout Manager* é responsável por obter informações sobre o inimigo como a localização das suas unidades.

Na próxima seção são apresentados detalhes sobre a modelagem de jogador, que é crucial para uma boa representação do adversário no StarCraft e, consequentemente, na detecção de mudanças sobre o seu comportamento.

2.2 Modelagem de jogador

Modelagem de jogador envolve o estudo de modelos computacionais de jogadores, buscando compreender como os mesmos interagem com o jogo, podendo envolver a detecção e predição de *propriedades* vinculadas ao seu modo de jogo, e como essas *propriedades* podem ser representadas (YANNAKAKIS et al., 2013).

Modelagem de jogador é normalmente usada na interação humano-computador, sendo usada também quando o jogador é um agente computacional. Neste caso, a modelagem do jogador pode ser aplicada para aprimorar estes agentes como apresentados nos trabalhos descritos em (SCHADD; BAKKES; SPRONCK, 2007; WEBER; MATEAS, 2009).

A modelagem de jogador é útil para identificar as particularidades de jogadores, mesmo quando eles apresentam habilidades semelhantes em um jogo. Os jogadores podem se diferenciar pela forma como cumprem desafios durante um jogo ou pela forma que interagem com o jogo, considerando aspectos como velocidade de cliques, ordem de ações e repetições de comandos (CHARLES; BLACK, 2004; VALLIM, 2013). Logo, uma boa representação do jogador é essencial para que os diferentes aspectos relacionados a sua caracterização estejam de fato presentes na modelagem.

Nesta pesquisa deseja-se identificar mudanças no modo de jogar do adversário, a partir de uma modelagem baseada em atributos que descrevem o cenário de jogo, refletindo as consequências das tomadas de decisão dos jogadores e as particularidades dos seus estilos de jogo. Esse modelo passa por atualizações constantes, buscando manter uma real representação do adversário, tendo como base as mais recentes informações obtidas.

Na próxima seção é apresentada uma fundamentação sobre o que consiste a detecção de mudanças no cenário de fluxo contínuo de dados e as características que tais mudanças podem ter.

2.3 Inteligência Artificial Explicável

A Inteligência Artificial Explicável ou *Explainable Artificial Intelligence* (XAI) tem sido usada para criar técnicas de aprendizado de máquina que permitem que usuários humanos entendam, confiem e gerenciem efetivamente sistemas artificialmente inteligentes (ARRIETA et al., 2020).

A explicabilidade em IA está associada à noção de explicação como uma interface entre humanos e um tomador de decisão (ARRIETA et al., 2020), se tornando uma forma de intermediário entre estes dois pontos, facilitando a compressão do lado humano sobre o que sistema retorna em seu processamento.

A técnicas que podem ser classificadas como XAI, permitem produzir modelos mais explicáveis, mantendo um alto nível de desempenho de aprendizagem, como na precisão de previsões. Além disso, dependendo do contexto e das necessidades dos usuários finais, eles podem confiar no XAI para responder a uma série de perguntas, como “Como funciona?”, “Que erros ele pode cometer?” e “Por que ele fez isso?” (KURDZIOLEK, 2022).

A literatura XAI usa os seguintes termos para descrever a compreensão humana quanto aos processos de aprendizado de máquina (ARRIETA et al., 2020; KURDZIOLEK, 2022):

1. Interpretabilidade: Uma característica passiva de um sistema de aprendizado de máquina. Se um sistema é interpretável, ele pode ser descrito por meio de termos compreensíveis pelos humanos.

2. Explicabilidade: Está associada à noção de explicação como uma interface entre humanos e um sistema de aprendizado de máquina que o torna compreensível.

3. Transparência: Um modelo é considerado transparente se, por si só, for compreensível. Um modelo é transparente quando um humano pode entender sua função sem qualquer necessidade de explicação.

4. Opaco: O oposto de um modelo transparente. Modelos opacos não são facilmente compreendidos por humanos. exigindo explicações.

5. Compreensibilidade: A propriedade que um algoritmo de aprendizado de máquina deve ter para representar o conhecimento aprendido de uma forma compreensível pelo ser humano.

Os termos "IA interpretável" e "IA explicável" às vezes são usados de forma intercambiável. No entanto, há uma clara distinção entre os dois termos. Uma IA interpretável é aquela que é compreensível para os humanos. A IA explicável é uma interface entre humanos e modelos de aprendizado de máquina que tenta tornar o modelo interpretável (KURDZIOLEK, 2022).

A seguir estão listados os principais tipos de explicação usadas com técnicas de aprendizado de máquina para desenvolver métodos XAI (KURDZIOLEK, 2022):

1. A *explicação por simplificação* fornece explicação por meio de extração e separação de regras.

2. A *explicação de relevância do recurso* fornece explicação por meio da classificação ou medição da influência que cada recurso tem em uma saída de previsão.

3. A *explicação visual* fornece explicação por meio da representação visual das previsões.

4. A *explicação por conceito* fornece explicação por meio de conceitos, que podem ser definidos pelo usuário.

5. A *explicação por exemplo* fornece explicações por analogia, embora apresente proponentes ou oponentes aos dados.

Na presente pesquisa a *explicação por conceito* será usada na tarefa de entender e definir a semântica dos comportamentos e as reações que o agente terá diante da detecção de cada mudança representada por uma semântica definida.

2.4 Detecção de mudança em fluxo contínuo de dados

Um fluxo contínuo de dados é composto de uma sequência de dados que chegam continuamente apresentando as seguintes características (GAMA; GABER, 2007): os dados chegam em tempo real; não há qualquer controle sobre a ordem de chegada dos dados; não há limite no tamanho do fluxo, e consequentemente, um dado já processado precisa ser descartado, pois é preciso lidar com recursos limitados como a memória.

Em um cenário de fluxo contínuo, quando a distribuição do dado muda ao longo do tempo, temos um evento conhecido como *mudança de conceito* (GAMA et al., 2014). Considere que D represente um fluxo contínuo de dados, composto pelas amostras $\{d_1, d_2, ...\}$, que podem ser organizadas em sequências $\{S_1, S_2, ...\}$ de amostras geradas por distribuições estacionárias G_i, onde $i \geq 1$. Cada sequência gerada representa um *contexto* (GAMA, 2010).

É possível que entre duas sequências consecutivas S_i e S_j geradas pelas distribuições G_i e G_j, respectivamente, ocorra uma fase de transição na qual amostras produzidas por uma distribuição diferente da G_j apareçam. Neste caso, tais amostras pode ser vistas como ruídos pela distribuição G_j (GAMA, 2010). Então, cada distribuição G_i produz um diferente conceito C_i. Neste caso, a expressão *mudança de conceito* indica que os conceitos associados às sequências de dados que compõem o fluxo, podem sofrer alterações ao longo do tempo (GAMA, 2010), e tais alterações são chamadas de mudanças.

A detecção de mudança pode apresentar diferentes níveis de dificuldade considerando a forma como ocorrem as mudanças. Para este caso, as mudanças podem ser classificadas como abruptas ou graduais.

As mudanças abruptas são mais fáceis de serem identificadas, pois como são uma troca imediata de um conceito para outro, sem uma etapa de transição, poucas amostras de dados acabam sendo necessárias para sua identificação (GAMA, 2010).

As mudanças graduais são mais difíceis de serem detectadas porque um novo conceito (padrão de dados) é introduzido lentamente, visto que no momento da mudança existe uma etapa de transição, que é formada por amostras de dados de diferentes conceitos que podem ser vistos como ruídos, sem a predominância de qualquer conceito neste intervalo (GAMA, 2010). Contudo, gradualmente, o número de amostras do novo conceito vai aumentando até ele se tornar o conceito que prevalece na sequência.

As mudanças abruptas ou graduais também podem apresentar as características de mudanças recorrentes. As mudanças recorrentes se referem a conceitos que reaparecem no fluxo de dados em momentos distintos (GAMA et al., 2014).

Os fluxos de dados apresentados nas Figuras 3 e 4 ilustram, respectivamente, a presença de mudanças abruptas com recorrência e graduais com recorrência, envolvendo os conceitos representados por C_1 (indicado pelos círculos azuis), C_2 (indicado pelos círculos verdes) e C_3 (indicado pelos círculos laranjas). Em ambas as figuras, as seguintes mudanças ocorrem: C_1/C_2 (ou seja, a mudança ocorre do conceito C_1 para C_2), C_2/C_3 e, finalmente, C_3/C_1.

O valor de m, indicado nas figuras, representa a quantidade de amostras entre os momentos de ocorrência das mudanças. Então, os valores de m_1, m_2, m_3 e m_4 indicam que este intervalo não precisa ser igual entre as mudanças. É interessante observar que a recorrência presente nos exemplos apresentados nas Figuras 3 e 4, se deve ao reaparecimento do conceito C_1 no fluxo de dados, depois dele já ter aparecido no início do fluxo.

Na Figura 4 as amostras de alguns conceitos podem ser considerados ruídos nas seguintes situações: C_1 e C_3 são ruídos no intervalo m_2; C_1 e C_2 são ruídos no intervalo m_3; C_2 e C_3 são

ruídos no intervalo m_4.

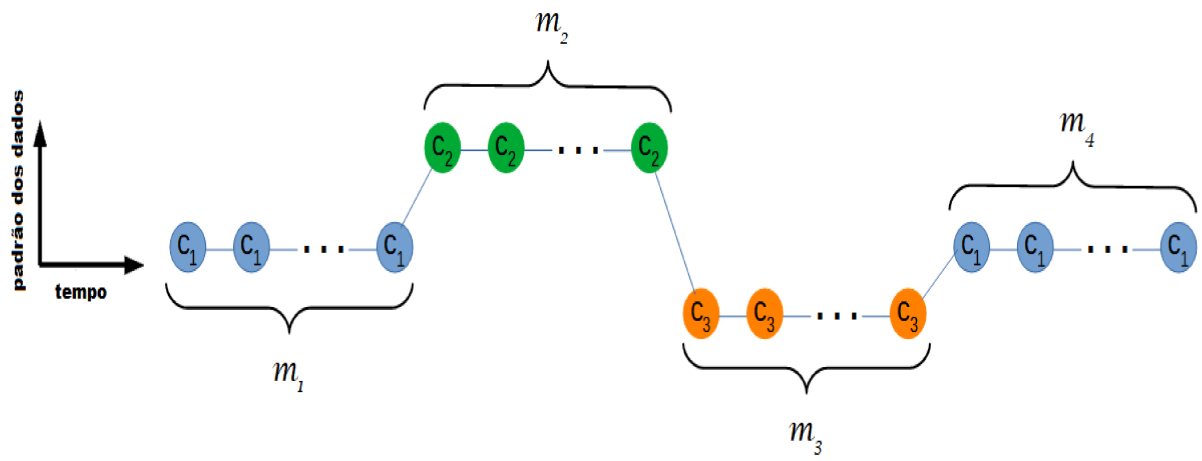


Figura 3 – Exemplo de mudanças abruptas com recorrência.

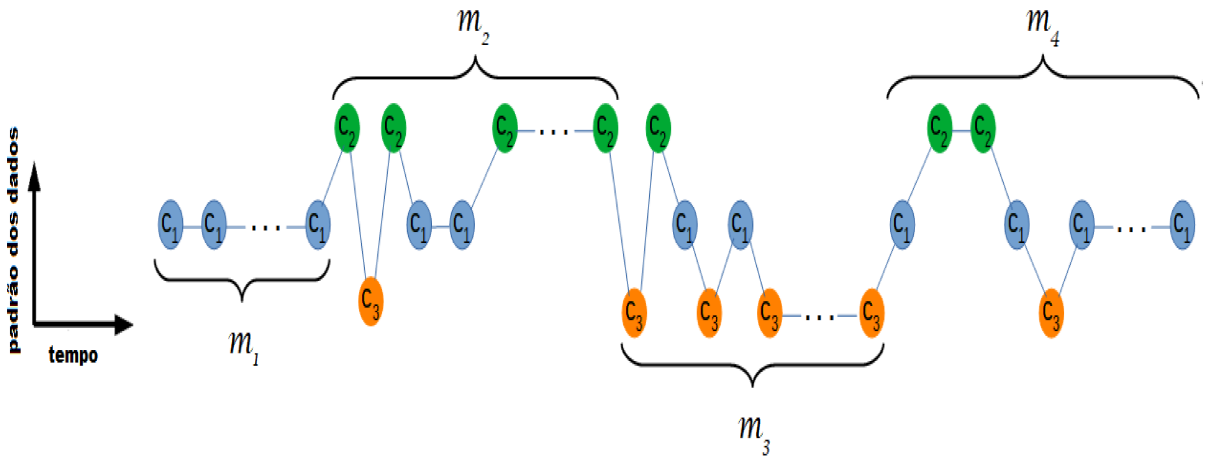


Figura 4 – Exemplo de mudanças graduais com recorrência.

Os vários métodos para lidar com mudança de conceito no cenário de fluxo contínuo de dados podem apresentar diferenças entre eles. Estas diferenças podem ser organizadas em função de dimensões que descrevem, por exemplo, a forma como as distribuições dos dados são monitoras para identificação de mudanças, ou como os modelos criados por estes métodos são atualizados (GAMA, 2010; GAMA et al., 2014). Essas dimensões são organizadas da seguinte forma:

1. *Data Management*: esta dimensão visa estabelecer o escopo dos dados a serem considerados no processo de monitoramento de mudanças na distribuição. Para isso, baseia-se preponderantemente nos dados mais recentes e não depende de parâmetros relacionados aos métodos de monitoramento. Com relação a esta dimensão, os métodos podem ser divididos nos seguintes grupos:

a) Abordagem *Partial Memory*: composto pelos métodos que monitoram as mudanças na distribuição dos dados, exclusivamente, por meio das amostras de dados mais recentes, cujo escopo é definido por meio de janelas fixas ou adaptativas.

b) Abordagem *Full Memory*: os métodos deste grupo, apesar de levarem em consideração todas as amostras de dados (mesmo as mais antigas), no processo de monitoramento de mudanças, eles atribuem pesos maiores às amostras mais recentes.

2. *Detection*: esta dimensão estabelece os parâmetros a serem monitorados para analisar as mudanças na distribuição dos dados. Nesta dimensão, os métodos podem ser divididos em dois grupos:

   a) Abordagem *Performance Parameter*: os métodos pertencentes a este grupo utilizam o desempenho de seus modelos de decisão como parâmetro para investigar mudanças na distribuição dos dados. Neste caso, uma mudança é identificada sempre que ocorrer uma redução no desempenho do modelo, decorrente das mudanças na distribuição dos dados.

   b) Abordagem *Time-window*: esse grupo é formado pelos métodos que utilizam informações específicas relacionadas às mudanças encontradas nas distribuições de dados de duas janelas de tempo distintas. A diferença observada entre as janelas serve como parâmetro para detectar uma mudança.

3. *Adaptation*: Esta dimensão estabelece o momento em que os métodos devem atualizar seus modelos de decisão. Considerando tal dimensão, os métodos podem ser divididos em duas abordagens:

   a) Abordagem *Blind*: os métodos pertencentes a este grupo atualizam seus modelos de decisão em intervalos regulares de tempo, independentemente do momento em que ocorreram mudanças na distribuição dos dados.

   b) Abordagem *Informed*: este grupo é composto pelos métodos que atualizam seus modelos de decisão sempre que uma mudança na distribuição dos dados é detectada.

Na próxima seção são apresentados detalhes sobre o funcionamento do M-DBScan e como ocorre a detecção de mudanças por meio deste algoritmo.

2.5 M-DBScan

O algoritmo M-DBScan foi criado para detectar mudanças de comportamento, baseadas em novidades sobre a distribuição dos dados, em cenários de fluxo contínuo. Este algoritmo possui um processo incremental de agrupamento, baseado no DenStream (CAO et al., 2006), que é uma técnica de agrupamento usada no cenário de fluxo de dados. O processo de agrupamento,

que também é baseado no DBSCAN (ESTER et al., 1996), é seguido por uma etapa de detecção de novidades usada na identificação de mudanças de comportamento (VALLIM, 2013).

O processo de agrupamento do M-DBScan pode ser dividido em duas etapas: *online* e *offline*. Então, para cada amostra de dado que chega do fluxo, o processo incremental de agrupamento é executado. Posteriormente, os grupos formados são usados em um processo de detecção de novidades. Este processo ocorre com o auxílio de uma Cadeia de Markov (CM), uma janela deslizante e uma medida de entropia.

Para compreender a dinâmica do processo de agrupamento é preciso conhecer alguns conceitos básicos como *core-micro-cluster*, *p-micro-cluster* e *o-micro-cluster*, que são apresentados nas sub-seções a seguir, e foram propostos, originalmente, com o algoritmo DenStream.

2.5.1 *Core-micro-cluster*

Um *core-micro-cluster* (*c-micro-cluster*) pode ser representado por uma tupla CMC(*w*,*c*,*r*), sendo *w* o peso, *c* o centro e *r* o raio, considerando um grupo de n exemplos próximos $p_1, ..., p_n$ com marcadores de tempo $T_1, ..., T_n$. O peso é calculado pela equação 1, o centro de um *c-micro-cluster* pode ser calculado pela equação 2, e o raio de um *c-micro-cluster* pode ser calculado pela equação 3.

Um *c-micro-cluster* pode ser visto como um *micro-cluster* denso, contudo a quantidade de *c-micro-clusters* é muito menor do que a quantidade de amostras de dados que chegam do fluxo. Dessa maneira, a tupla CMC(*w*,*c*,*r*) sumariza informações relacionadas ao resultado de um agrupamento de objetos próximos (CAO et al., 2006).

Uma importante questão quanto ao agrupamento em fluxo de dados é que o peso tem seu valor reduzido exponencialmente devido a uma função de decaimento, que usa como parâmetros o tempo corrente t, $f(t) = 2^{-\lambda t}$, onde λ é um fator de decaimento e $\lambda > 0$. Essa função de decaimento é aplicada sobre qualquer *micro-cluster* que não tenha recebido amostra de dados do fluxo.

$$w = \sum_{j=1}^{n} f(t - T_j), \tag{1}$$

$w \geq \mu$, μ é a mínima quantidade de pontos, e t é o tempo corrente.

$$c = \frac{\sum_{j=1}^{n} f(t - T_j) p_j}{w} \tag{2}$$

$$r = \frac{\sum_{j=1}^{n} f(t - T_j) dist(p_j, c)}{w} \tag{3}$$

$r \leq \varepsilon$, ε é o valor limite para o raio e $dist(p_j, c)$ é a Distância Euclideana entre o ponto p_j e o centro c.

Um conjunto de *c-micro-clusters* deve corresponder a cobertura dos núcleos de objetos presentes no grupo (ESTER et al., 1996). Ao longo de um fluxo de dados, aquilo que é visto

como um grupo ou como um *outlier* pode sofrer alterações devido a chegada ou não de novos dados, desencadeando o uso ou não da função de decaimento, e em virtude disso, serem classificados de outra maneira. Então, o algoritmo M-DBScan utiliza duas estruturas: *potential c-micro-cluster* (*p-micro-cluster*) e o *outlier-micro-cluster* (*o-micro-cluster*) para representar essas diferenças entre os *micro-clusters*.

2.5.2 *P-micro-cluster*

Um *p-micro-cluster* é representado por uma tupla $\{\overrightarrow{CF^1}, \overrightarrow{CF^2}, w\}$, considerando um grupo de exemplos $p_1, ..., p_n$ com marcadores de tempo $T_1, ..., T_n$. O vetor contendo a soma ponderada dos exemplos por dimensão é representado por $\overrightarrow{CF^1}$, sendo calculado pela equação 4. O vetor contendo a soma ponderada quadrática dos exemplos por dimensão, representado por $\overrightarrow{CF^2}$, é calculado pela equação 5, e o peso w é determinado pela equação 1. Contudo, para um *micro-cluster* ser classificado como um *p-micro-cluster*, é preciso atender a seguinte restrição: $w \geq \beta.\mu$, sendo β ($0 < \beta \leq 1$) um limiar de *outlier* que multiplicado pela mínima quantidade de pontos (μ), define o valor de referência usado para classificar um *micro-cluster* como *p-micro-cluster*.

$$\overrightarrow{CF^1} = \sum_{j=1}^{n} f(t - T_j) p_j \tag{4}$$

$$\overrightarrow{CF^2} = \sum_{j=1}^{n} f(t - T_j) p_j^2 \tag{5}$$

O centro de um *p-micro-cluster* (cPMC) é determinado pela equação 6, e o seu raio (rPMC) é calculado pela equação 7, sendo $rPMC \leq \varepsilon$ (ε é o limite máximo para o raio de um *micro-cluster*).

$$cPMC = \frac{\overrightarrow{CF^1}}{w} \tag{6}$$

$$rPMC = \sqrt{\frac{|\overrightarrow{CF^2}|}{w} - (\frac{|\overrightarrow{CF^1}|}{w})^2} \tag{7}$$

2.5.3 *O-micro-cluster*

Um *o-micro-cluster* é representado por uma tupla $\{\overrightarrow{CF^1}, \overrightarrow{CF^2}, w, t_0\}$. As definições de $\overrightarrow{CF^1}$, $\overrightarrow{CF^2}$ e w, são as mesmas que aquelas apresentadas na descrição de *p-micro-cluster* na Seção 2.5.2. Um dado importante sobre um *o-micro-cluster* é o seu momento de criação (t_0), visto que esta informação é usada para determinar o tempo de vida do *micro-cluster*. A principal diferença entre um *o-micro-cluster* e um *p-micro-cluster* é o peso, visto que se uma quantidade de exemplos que chegam do fluxo de dados é atribuída para um *o-micro-cluster*, o seu peso irá

mudar, e consequentemente, o *o-micro-cluster* pode se tornar um *p-micro-cluster* se a restrição de um *p-micro-cluster* ($w \geq \beta.\mu$) é satisfeita.

Por sua vez, um *p-micro-cluster* que não recebe mais exemplos de dados por um certo período de tempo, pode ser reclassificado como um *o-micro-cluster* decorrente do uso do fator de decaimento que atua sobre a tupla que representa o *p-micro-cluster* $\{\overrightarrow{CF^1}, \overrightarrow{CF^2}, w\}$.

2.5.4 Fases *online* e *offline* do M-DBScan

As fases *online* e *offline* do M-DBScan ocorrem de maneira sequencial e são essenciais para a construção da CM que ocorre com base no resultado do agrupamento decorrente dessas duas fases. A fase *online* está detalhada na Seção 2.5.4.1 e a fase *offline* está detalhada na Seção 2.5.4.2.

2.5.4.1 Fase *online* M-DBSCAN

A fase *online* do algoritmo M-DBScan pode ser descrita como um processo de manutenção de *micro-clusters*, no qual conjuntos de *o-micro-cluster* e *p-micro-cluster* são mantidos de maneira *online*, com o objetivo de descobrir na fase *offline* grupos a partir desses *micro-clusters*, que representam os exemplos que chegaram do fluxo de dados.

A fase *online* pode ser sumarizada pelos seguintes passos, que são repetidos sempre que uma amostra de dados *p* chega do fluxo de dados:

1. Uma vez que a amostra *p* acaba de chegar do fluxo, o algoritmo tenta atribuir o dado ao *p-micro-cluster* (*pmc*) mais próximo. Se *p* está dentro do raio de *pmc*, então *p* é atribuído a *pmc*.

2. Caso contrário, o algoritmo tentará atribuir *p* ao *o-micro-cluster omc* mais próximo. Se *p* está dentro do raio de *omc*, então *p* é atribuído a *omc*. Posteriormente, verifica-se se o peso do *omc* ultrapassou o limite de um *o-micro-cluster*, se tornando um *p-micro-cluster*. Caso esse limite tenha sido ultrapassado, o *omc* será promovido a um novo *p-micro-cluster* e será removido do *outlier-buffer*, onde os *o-micro-clusters* são mantidos.

3. Senão, um novo *o-micro-cluster new_omc* é criado para alocar a amostra *p*, e o *new_omc* é inserido no *outlier-buffer*.

Posteriormente à fase *online*, o fator de decaimento é aplicado a todo *micro-cluster*, com exceção daquele que recebeu a última amostra de dado que chegou do fluxo.

Com base em um intervalo de tempo definido previamente, os *micro-clusters* passam por uma etapa de verificação, na qual um *p-micro-cluster* pode ser reclassificado como *o-micro-cluster*, caso ele não atenda mais os requisitos de um *p-micro-cluster*, e um *o-micro-cluster* pode ser apagado, se ele não atender o critério de peso mínimo para um *micro-cluster*.

2.5.4.2 Fase *offline* M-DBSCAN

A fase *offline* do M-DBScan consiste na aplicação direta dos conceitos propostos originalmente pelo algoritmo *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN) (ESTER et al., 1996). O DBSCAN permite descobrir agrupamentos de dados de formatos variados, além de ruídos que possam existir nos dados obtidos, tendo como base a densidade das amostras de dados no espaço. Dessa maneira, todos os pontos que são alcançáveis por densidade podem ser agrupados com o uso dos parâmetros corretos do algoritmo (ESTER et al., 1996).

No caso da fase *offline* do M-DBScan, esse agrupamento é gerado a partir do resultado produzido pela fase *online*, com o objetivo de determinar um agrupamento final considerando os *micro-clusters* formados até o momento corrente pela fase anterior. A fase *offline* adota os conceitos de densidade e conectividade, que permitem produzir grupos nos quais os *p-micro-clusters* são, por exemplo, conectados por densidade. Esses conceitos são apresentados a seguir (CAO et al., 2006):

- Diretamente alcançável por densidade: Um *p-micro-cluster* PMC_1 é diretamente alcançável por densidade a partir de um *p-micro-cluster* PMC_2, se $PMC_2.peso > \mu$ e dist(centro PMC_1, centro PMC_2) $\leq 2.\varepsilon$, onde dist(centro PMC_1, centro PMC_2) é a distância Euclidiana entre os centros de PMC_1 e PMC_2.

- Alcançável por densidade: Um *p-micro-cluster* PMC_1 é alcançável por densidade a partir de um *p-micro-cluster* PMC_n, se existir um cadeia de *p-micro-clusters* entre PMC_1 e PMC_n, de tal modo que PMC_{i+1} é diretamente alcançável por densidade pelo PMC_i, onde $1 \leq i \leq n$.

- Conectado por densidade: Um *p-micro-cluster* PMC_1 é conectado por densidade a partir de um *p-micro-cluster* PMC_n, se existir um *p-micro-cluster* PMC_k, de tal modo que, tanto o PMC_1 quanto o PMC_n são alcançáveis por densidade a partir de PMC_k.

Uma vez que a fase *offline* é concluída, o módulo de detecção de novidade, que utiliza Cadeia de Markov (CM), inicia. Neste processo cada estado da CM representa um grupo produzido na fase *offline* do M-DBScan, e as probabilidades presentes nas transições entre os estados refletem a dinâmica de chegada de amostras do fluxo de dados, e para qual *micro-cluster* uma amostra de dado foi atribuída.

2.5.5 Módulo de detecção de novidade e mudança de comportamento

Com o objetivo de detectar novidades, o algoritmo M-DBScan utiliza medidas de entropia, sendo adotadas as entropias espacial e temporal, contudo, em alguns dos experimentos do trabalho foi usada somente a entropia espacial por ter apresentado melhores resultados.

A entropia espacial estima uma distribuição espacial dos dados em grupos. O número de amostras atribuídas a um grupo i é denotado por P_i, e esse valor é atualizado sempre que uma nova amostra de dado é atribuída ao grupo (equação 8). Contudo, se existir outros grupos aos quais não foi atribuída uma amostra de dado no momento corrente, então, esses grupos serão atualizados pela equação 9, onde η_s é um fator de ponderação utilizado em ambas atualizações.

$$P_i = (1 - \eta_s).P_i + \eta_s \tag{8}$$

$$P_i = (1 - \eta_s).P_i \tag{9}$$

A entropia espacial ($H(G)$) é calculada de acordo com a equação 10, na qual G é o conjunto de grupos e N é a quantidade total de grupos.

$$H(G) = -\sum_{i=0}^{N} P_i.\log_2(P_i) \tag{10}$$

Uma vez que os estados na CM são uma representação dos grupos produzidos na fase *offline* do M-DBScan, as transições na CM são atualizadas sempre que uma nova amostra de dado chega do fluxo. Sendo que a soma das probabilidades vinculadas às transições com o mesmo estado como origem, tem o seu valor máximo igual a 1. Essa probabilidade na transição é essencial para o uso da entropia temporal, visto que ela reflete a frequência em que amostras de dados são atribuídas a um grupo representado por um estado na CM. A probabilidade entre os estados m e n da CM é representada por $P_{m,n}$, onde m é o estado ao qual foi atribuída a amostra de dado no momento $T-1$ e n representa o estado ao qual foi atribuída a amostra de dado no momento T. O fator de ponderação η_t é usado para controlar a intensidade da atualização de cada probabilidade. As transições que não são mais ativadas, devido a dinâmica de chegada de novas amostras de dados, terão uma deterioração no seu valor decorrente das atualizações.

Considerando que o estado j acaba de ser definido como o destino de uma amostra de dado, e o estado anterior ao qual foi atribuído um dado é i, então, a atualização da probabilidade $P_{i,j}$ é realizada pela equação 11.

$$P_{i,j} = \frac{(1 - \eta_t).P_{i,j} + \eta_t}{\sum_{k=0}^{N}.P_{i,k}} \tag{11}$$

As outras transições na CM terão suas probabilidades atualizadas pela equação 12.

$$P_{a,b} = \frac{(1 - \eta_t).P_{a,b}}{\sum_{k=0}^{N}.P_{a,k}} \tag{12}$$

A entropia temporal ($H(M)$) é calculada pela equação 13, onde M é o conjunto de estados na CM.

$$H(M) = -\sum_{i=0}^{N}\sum_{j=0}^{N} P_{i,j}.\log_2(P_{i,j}) \tag{13}$$

Para identificar uma novidade, uma entropia precisa ultrapassar o seu limiar, que é calculado com base em uma média móvel sobre os valores do histórico de entropia Φ (equação 14) e um desvio padrão móvel Ω (equação 15), onde γ e δ são fatores de ponderação, cujos valores pertencem ao intervalo [0,1]. O elemento das equações representado por $H_t(.)$ indica o valor de entropia no momento t.

$$\Phi_t = (1-\gamma).\Phi_{t-1} + \gamma.H_t(.) \tag{14}$$

$$\Omega_t = (1-\delta).\Omega_{t-1} + \delta.|H_t(.) - \Phi_t| \tag{15}$$

Para calcular o limiar de uma entropia (τ) utiliza-se a equação 16. O algoritmo M-DBScan assume uma distribuição normal sobre os valores de entropia, adotando um parâmetro constante θ, que representa o número de desvios padrões a partir da média.

$$\tau = \Phi_t + (\Omega_t.\theta) \tag{16}$$

Para indicar uma mudança de comportamento, uma sequência de novidades precisa ocorrer. Então, o algoritmo M-DBScan requer que uma quantidade mínima de novidades deve ser identificada em um intervalo de tempo representado pelo valor k que é o tamanho de uma janela deslizante utilizada no processo. Cada ocorrência de novidade é registrada na janela deslizante, onde é preciso verificar se a quantidade de novidades registradas atingiu a quantidade mínima necessária para configurar uma mudança de comportamento. Uma novidade que ocorreu em um certo momento pode desaparecer da janela deslizante, caso essa janela tenha deslizado k vezes, considerando como início o momento em que a novidade foi registrada na janela.

Contudo, se uma mudança de comportamento foi detectada, a partir desse momento toda novidade será ignorada por um período de tempo igual a k. Este processo é usado para descartar novidades vinculadas à mudança de comportamento que já foi identificada (VALLIM et al., 2013).

A detecção de mudanças pode lidar com cenários em que as mudanças possuem diferentes características como as de mudanças abruptas e graduais. Nas mudanças abruptas, o conceito representado em uma sequência de dados muda repentinamente. Nas mudanças graduais, a mudança de conceito ocorre lentamente, de tal modo que no período de transição de uma mudança não há um conceito predominante, aparecendo amostras de dados de diferentes distribuições durante a transição antes que ocorra a estabilização e um único conceito prevaleça na sequência. Isso é um fator de dificuldade visto que o algoritmo pode interpretar que tais amostras, com conceitos aleatórios, sejam ruídos (GAMA, 2010). As mudanças abruptas e graduais podem apresentar a propriedade de serem recorrentes, ou seja, depois de um tempo em que um conceito apareceu no fluxo de dados, ele pode reaparecer(GAMA et al., 2014).

2.5.6 Exemplificando a aplicação do M-DBScan

A tarefa de detectar mudança no comportamento do jogador envolve os atributos que descrevem relevantes aspectos do estilo de jogo. Uma maneira de realizar essa tarefa é utilizar a detecção de novidade, que pode ser definida como um processo capaz de identificar uma amostra de dado que difere de algum conceito já conhecido. Esta área tem recebido uma grande atenção em pesquisas de aprendizagem de máquina e mineração de dados (FARIA et al., 2016).

Devido a interação entre jogadores no contexto do jogo, dos quais pode-se detectar novidades que venham a caracterizar uma mudança de comportamento, é possível obter uma série de dados na forma de fluxo contínuo que possuem características próprias como (GAMA; GABER, 2007): o dado chega de modo *online*; não há um controle sobre a ordem em que o dado chega; não há limite para o tamanho do fluxo de dados, e isso implica que uma amostra de dado já processada, é normalmente descartada, visto que o sistema computacional lida com recursos limitados como memória.

Para exemplificar a dinâmica de criação de estados da CM, será usada a seguinte sequência como parte de um fluxo de dados (0, 0, 1, 0, 1, 0, 0, 0, 0, 0), onde cada dado obtido do fluxo foi classificado em um dos dois grupos (grupo 0 ou grupo 1). Nessa exemplificação foram feitas algumas mudanças nos parâmetros do processo de agrupamento, como a redução do peso mínimo de um *micro-cluster*, para garantir a exemplificação de diferentes situações sem a necessidade de utilizar muitos dados do fluxo.

Com a chegada da primeira amostra de dado, representante do grupo 0, é criado um *o-micro-cluster* chamado de omc_1, visto que não existia nenhum *micro-cluster* ao qual pudesse ser atribuído o primeiro dado que chega do fluxo. Nesse momento também não é possível criar um estado na CM, pois ainda não existe nenhum grupo gerado na fase *offline* do M-DBScan.

Com a chegada da segunda amostra de dado, representante do grupo 0, considere que ela é atribuída ao *o-micro-cluster* omc_1, com isso o omc_1 é promovido a um *p-micro-cluster* chamado de pmc_1. Com a fase *offline* do M-DBScan, foi criado um grupo, que por sua vez, se tornou o estado 1 na CM ilustrada na Figura 5. Como todo dado que chegou está vinculado ao mesmo grupo, a probabilidade de transição é de 1 para se manter no mesmo estado. Nesse momento nenhuma novidade foi detectada.

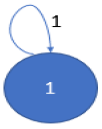


Figura 5 – Representação da CM com um único estado.

Fonte: Elaborada pelo autor.

Com a chegada da terceira amostra de dado, representante do grupo 1, considere a criação do *o-micro-cluster* chamado de omc_2, visto que considerando as medidas de raio do *micro-cluster*

existente, ele não conseguiu englobar o novo dado. Com a fase *offline* do M-DBScan, continua existindo somente um grupo, representado pelo estado 1 na CM ilustrada na Figura 5. O valor de probabilidade não é alterado por que o omc_2 não tem representação na CM. Nesse momento nenhuma novidade foi detectada.

Com a chegada da quarta amostra de dado, representante do grupo 0, considere que ela é atribuída ao *p-micro-cluster* já existente pmc_1. Com a fase *offline* do M-DBScan, continua existindo somente um grupo, representado pelo estado 1 na CM, e nenhuma novidade foi detectada.

Com a chegada da quinta amostra de dado, representante do grupo 1, considere que ela é atribuída ao *o-micro-cluster* já existente (omc_2), que é promovido a *p-micro-cluster* e será chamado de pmc_2. Com a fase *offline* do M-DBScan dois grupos foram formados, existindo assim dois estados na CM, como ilustrada a Figura 6. Com a existência de mais um estado observa-se mudanças nos valores de probabilidades de transição, que foram calculados e atualizados pelas equações 11 e 12. Nesse momento nenhuma novidade foi identificada.

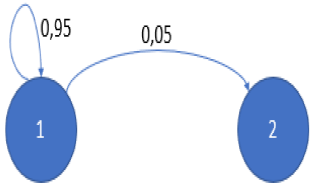


Figura 6 – Representação da CM com dois estados após a chegada da quinta amostra de dado.

Fonte: Elaborada pelo autor.

Com a chegada da sexta amostra de dado, representante do grupo 0, considere que ela é atribuída ao pmc_1. Com a fase *offline* do M-DBScan, continua a existir dois estados na CM, como ilustrada a Figura 7, sendo que os valores das probabilidades das transição são atualizados. Nesse momento, considere que uma novidade foi identificada pela entropia temporal, lembrando que para identificar a mudança de comportamento uma quantidade mínima de novidades precisa ser identificada.

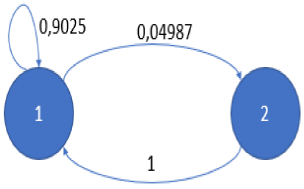


Figura 7 – Representação da CM com dois estados após a chegada da sexta amostra de dado.

Fonte: Elaborada pelo autor.

Com a chegada da sétima amostra de dado, representante do grupo 0, considere que ela é atribuída ao pmc_1. Com a fase *offline* do M-DBScan, continua a existir dois estados na

CM, como ilustrada a Figura 8, sendo que os valores das probabilidades das transições são atualizados. Nesse momento nenhuma novidade foi identificada.

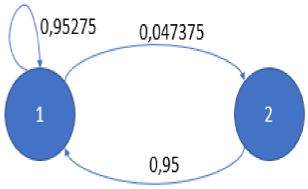


Figura 8 – Representação da CM com dois estados após a chegada da sétima amostra de dado.

Fonte: Elaborada pelo autor.

Com a chegada da oitava amostra de dado, representante do grupo 0, considere que ela é atribuída ao pmc_1. Com a fase *offline* do M-DBScan, continua a existir dois estados na CM, como ilustrada a Figura 9, sendo que os valores das probabilidades das transições são atualizados. Nesse momento nenhuma novidade foi identificada.

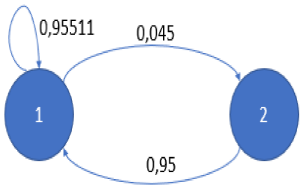


Figura 9 – Representação da CM com dois estados após a chegada da oitava amostra de dado.

Fonte: Elaborada pelo autor.

Com a chegada da nona amostra de dado, representante do grupo 0, considere que ela é atribuída ao pmc_1. Com a fase *offline* do M-DBScan, continua a existir dois estados na CM, como ilustrada a Figura 10, sendo que os valores das probabilidades das transições são atualizados. Nesse momento nenhuma novidade foi identificada.

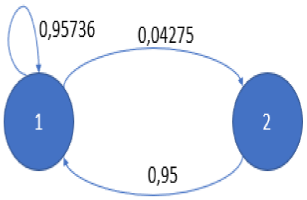


Figura 10 – Representação da CM com dois estados após a chegada da nona amostra de dado.

Fonte: Elaborada pelo autor.

Com a chegada da décima amostra de dado, representante do grupo 0, considere que ela é atribuída ao pmc_1. Com a fase *offline* do M-DBScan apenas um grupo é formado, como ilus-

trada a Figura 11. O desaparecimento de um grupo se deve ao uso do fator de decaimento que incidiu sobre o pmc_2, uma vez que, desde a sexta amostra, os dados eram atribuídos ao pmc_1, provocando o desaparecimento do pmc_2 que volta a ser um *o-micro-cluster*. Nesse momento nenhuma novidade foi identificada.

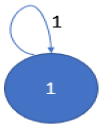


Figura 11 – Representação da CM com dois estados após a chegada da décima amostra de dado.

Fonte: Elaborada pelo autor.

Pelo exemplo usado para ilustrar a dinâmica de criação de grupos, foi possível observar a identificação de uma novidade com a entropia temporal, mas isso não foi suficiente para indicar uma mudança de comportamento, que talvez poderia ocorrer com a chegada de mais dados do fluxo.

2.6 Considerações Finais

Este capítulo apresentou conceitos importantes para o desenvolvimento desta pesquisa. O capítulo iniciou-se apresentando dados referentes ao jogo StarCraft, descrevendo informações básicas sobre o jogo, destacando desafios particulares do jogo, e como funciona atualmente o processo de tomada de decisões no StarCraft.

Na sequência foram apresentadas informações referentes a modelagem de jogador, que é um processo importante na pesquisa, visto que os dados obtidos do jogo StarCraft são representações do jogo que refletem, por exemplo, as ações do adversário, e a qualidade dessas representações estão diretamente relacionadas ao processo de detecção de mudanças sobre o comportamento do adversário.

Posteriormente, foi formalizado o conceito de detecção de mudanças, abordando as características das mudanças graduais e abruptas, além do conceito de mudanças recorrentes.

O último conceito apresentado neste capítulo se refere a versão original do algoritmo M-DBScan, onde o mesmo tem seu funcionamento descrito, com destaque para o processo de detecção de mudanças em comportamentos, que originalmente não possui um procedimento capaz de atribuir uma semântica à mudança de comportamento identificada.

No próximo capítulo serão descritos os trabalhos correlatos a esta pesquisa, destacando as diferenças entre os trabalhos existentes quanto ao que está sendo desenvolvido nesta pesquisa.

CAPÍTULO 3

Trabalhos Relacionados

Para realização desta pesquisa, estudos sobre trabalhos correlatos foram realizados, e estes trabalhos estão organizados em três subgrupos.

- ❏ Trabalhos que aprimoram jogos RTS, com destaque para o StarCraft;
- ❏ Trabalhos que lidam com questões referentes ao processamento de dados que chegam na forma de um fluxo contínuo, englobando a detecção de novidades sobre esses dados;
- ❏ Trabalhos que utilizam classificação para lidar com mudança de conceito.

3.1 Trabalhos relacionados a jogos RTS

As pesquisas relacionadas a jogos RTS, apresentadas nesta seção, podem ser divididas entre as seguintes categorias de pesquisas:

- ❏ Aprimoramento da gerência de ações em jogos RTS (Seção 3.1.1), que apresenta as seguintes abordagens:
  - Aprimoramento do microgerenciamento, que envolve as decisões sobre o combate no jogo.
  - Aprimoramento do macrogerenciamento, que envolve as decisões sobre as construções a serem feitas e a ordem em que devem ocorrer.
- ❏ Predições no contexto de um jogo RTS (Seção 3.1.2), que apresentam as seguintes abordagens:
  - Predições sobre composição do exército, tentando prever o que será enfrentado pelo agente.
  - Predições sobre a possível ação que o jogador adversário fará.
  - Predições sobre o adversário de uma partida, com base em um modelo genérico de jogador.

3.1.1 Trabalhos relacionados ao aprimoramento da gerência de ações em jogos RTS

Dentre os trabalhos que aplicam alguma forma de aprimoramento na gerência de ações de jogos RTS, destaca-se o trabalho descrito em (JUSTESEN; RISI, 2017), onde foi explorada uma importante tarefa presente no jogo StarCraft, que consiste na decisão sobre quais estruturas devem ser construídas. Em virtude disso, neste trabalho é apresentado o algoritmo *Continual Online Evolutionary Planning* (COEP) que é capaz de adaptar ao longo do jogo a ordem de construções que ocorrem no StarCraft. As adaptações na ordem em que ocorrem as construções, propostas pelo COEP, ocorrem durante todo o jogo, atuando no macrogerenciamento, que envolvem as decisões sobre quais construções devem ocorrer. As decisões sobre o controle das unidades no combate ao adversário não fazem parte do escopo deste trabalho. O COEP é um algoritmo evolutivo, que de maneira semelhante a um algoritmo genético, constrói sua população de possíveis soluções para o problema, que é definir a construção a ser realizada em um determinado momento. O COEP aplica técnicas como *crossover* para incorporar diversidade na população, além de fazer uso de uma função *fitness* que é calculada após a simulação da construção indicada pelo individuo avaliado. Essa avaliação utiliza conhecimentos específicos do jogo StarCraft, como por exemplo, dano e defesa das unidades que podem ser construídas e que indicam vantagens ou desvantagens que alguns tipos de unidades têm em relação a outras. O COEP possui o diferencial de execução em paralelo, então, quando deseja-se construir uma estrutura no jogo, utiliza-se as informações do indivíduo mais bem avaliado da população, e simultaneamente, atualiza-se uma representação do estado do jogo considerando essa nova construção. Posteriormente, é aplicado um processo de avaliação sobre os indivíduos tendo como base essa nova configuração do jogo. Sendo importante destacar que essa representação tem como base aquilo que o jogador já construiu no jogo, não existindo uma visão sobre o comportamento do adversário. Observa-se que além dessa adaptação das avaliações, sobre qual seria a melhor solução considerando a nova representação, este trabalho correlato se difere da presente pesquisa, por considerar somente dados do próprio jogador, não obtendo dados do adversário que poderiam ter influência sobre a decisão de qual estrutura construir, como por exemplo, se as estruturas criadas vão permitir desenvolver um exército que conseguirá ter sucesso ao enfrentar o adversário. Portanto, este trabalho correlato não lida com a identificação de mudanças de comportamento e nem busca vincular o processo decisório do jogo StarCraft às percepções em tempo real que se tem sobre o adversário.

Alguns trabalhos correlatos tentam aprimorar as habilidades de combate de seus jogadores automáticos, como ocorre, no trabalho apresentado em (SHAO; ZHU; ZHAO, 2017), onde é descrito como foi incorporada a aprendizagem por reforço em um jogador automático de StarCraft, com o objetivo de ensinar múltiplas unidades a combater, tendo como base a representação corrente do jogo. Neste trabalho também foi apresentada uma maneira mais eficiente de representação do jogo StarCraft, que quebra um pouco da sua complexidade. Essa representação é composta por três partes: a primeira parte contém as informações correntes do jogo

como, por exemplo, os pontos de vida das unidades, a distância entre as unidades do exército e a distância em relação as unidades do inimigo; a segunda parte contém as mesmas informações da primeira parte, mas referente a um momento anterior; a terceira parte contém informações sobre a última ação executada. A rede neural utilizada com a aprendizagem por reforço pode indicar como saída uma das oito possíveis movimentações a ser realizada por alguma unidade do jogo, ou realizar um ataque. A rede neural utiliza a função de ativação ReLU e a técnica de gradiente descendente Sarsa. Para o processo de aprendizagem por reforço foi definida uma função de recompensa, onde as recompensas seriam aplicadas em um intervalo de tempo relativamente pequeno. A aplicação de recompensa ocorre em função da diferença entre a perda de pontos de vida das unidades do adversário e do próprio jogador, sendo também aplicada uma punição quando uma unidade do jogador é destruída, pois isso tem um impacto negativo quanto ao objetivo de vencer a partida. Também é aplicada uma punição quando o caminho apontado pela rede neural não leva ao encontro do adversário. Os resultados apresentados em (SHAO; ZHU; ZHAO, 2017) mostram que as unidades conseguem vencer depois da rede passar por 1400 épocas de treinamento, e as taxas de vitória só são expressivas depois de 2000 épocas de treinamento da rede neural, chegando a quase 100% de vitória depois de 3600 épocas de treinamento. Este trabalho relacionado apresenta importantes diferenças quanto a presente pesquisa, devido ausência de qualquer análise sobre o comportamento do adversário, que poderia ser usada junto ao processo decisório do jogo, além disso, também não considera que diferentes atributos podem ser mais relevantes em certos momentos do jogo.

Em (PENG et al., 2017) é desenvolvido um estudo utilizando como cenário o jogo StarCraft, onde o objetivo é coordenar múltiplos agentes para derrotar o adversário. Foi apresentado neste trabalho o algoritmo *Multiagent Bidirectionally-Coordinated Network* (BiCNet), que é uma rede de aprendizagem profunda, que consegue lidar com diferentes tipos de combates e com uma quantidade variada de agentes. Este trabalho foca-se no microgerenciamento do jogo StarCraft, que contempla o controle das unidades de combate, sem se preocupar com o gerenciamento das construções. Neste trabalho, é aplicada uma comunicação bidirecional entre os agentes do mesmo exército caracterizando uma rede coordenada. Como cada dimensão na rede representa um agente, a coordenação é feita pelas ligações bidirecionais entre as camadas da rede neural. A BiCNet é baseada em uma rede neural recorrente, que utiliza de informações compartilhadas entre os agentes e de uma visão local para determinar a ação a ser executada pelo agente que está sendo controlado. O destaque das diferenças deste trabalho com a presente pesquisa estão vinculadas ao fato do trabalho correlato não aplicar técnicas que lidem com a aprendizagem sobre dados de um fluxo contínuo, e por não lidar com o objetivo de aprimorar o processo de tomada de decisão de um agente jogador, de tal modo que suas ações sejam tomadas considerando uma eventual mudança do estilo de jogo do adversário.

Em (The AlphaStar team, 2019) é descrito um trabalho feito sobre o jogo de RTS StarCraft II, que é uma versão diferente do StarCraft: BroodWar, mas também foi desenvolvido pela *Blizzard Entertainment*. No referido trabalho utiliza-se uma rede neural profunda com

aprendizagem por reforço e treinamento supervisionado. Neste treinamento, são usadas bases de dados de jogos, e também, são usadas partidas feitas contra o próprio jogador. A rede neural profunda recebe como entrada dados brutos obtidos diretamente da interface do jogo, que proporciona uma lista de unidades representadas por atributos, e a saída da rede é uma ação a ser executada pela unidade, que pode envolver, por exemplo, uma movimentação ou um ataque. Este trabalho relacionado não lida com detecção de mudanças e nem mesmo tenta interpretar o comportamento do adversário, de modo a reagir em função ao que foi identificado.

Na próxima seção são detalhados os trabalhos relacionados a tarefa de predição, detalhando como esta pode ser aplicada em jogos RTS.

3.1.2 Trabalhos relacionados a predições no contexto de jogos RTS

Dentre os trabalhos que envolvem predições, destaca-se a pesquisa presente em (WEBER; MATEAS, 2009), onde foi descrito um trabalho que fez uso de mineração de dados para a predição de estratégias no jogo StarCraft. Nesse trabalho, jogos foram convertidos em registros de ações, e então rotulados de acordo com uma estratégia. O objetivo foi aplicar mineração de dados em um conjunto de registros de jogos, para desenvolver um modelo que representa o adversário, sendo que esse modelo é treinado previamente considerando o estilo de jogo de diferentes oponentes. Também foram usados vetores de *features* para representar aspectos temporais das decisões estratégicas do jogador. Cada *feature* era formada pela informação do momento em que uma base era construída ou um tipo de unidade era treinada. O problema de predição estratégica descrita em (WEBER; MATEAS, 2009), foi tratado como um problema de classificação, e foram aplicados os seguintes algoritmos: árvore de decisão, **K-NN!** (**K-NN!**), *Non-Nested generalized exemplars* (NNge) e *Additive Logistic Regression*. Diferentemente do que é apresentado na presente pesquisa, o trabalho descrito em (WEBER; MATEAS, 2009) não lida com os desafios do cenário de fluxo contínuo de dados, e as técnicas de aprendizagem foram utilizadas no âmbito de análise e classificação. Enquanto que na presente pesquisa, almejasse o desenvolvimento de um módulo para tomada de decisão em tempo real, que funcionará com base na identificação de mudanças sobre o comportamento do adversário, para que o jogador automático reaja o mais breve possível e de forma contextualizada a estas mudanças, e não somente classifique as ações do adversário.

Em (SYNNAEVE; BESSIERE et al., 2012) é apresentada uma forma de descobrir táticas e estratégias no jogo StarCraft, não sendo utilizadas essas informações para influenciar a tomada de decisão do jogo, como propõe a presente pesquisa utilizando a indicação de mudança do estilo de jogo do adversário. Neste trabalho correlato foi criada uma base de dados, e nesta base foram analisadas as composições do exército durante as partidas. Esta análise envolveu a modelagem do exército gerando o agrupamento das suas composições, visto que este trabalho considera que um exército é uma mistura de vários componentes. A técnica utilizada foi a *Gaussian Mixture Model*, com o objetivo de se aplicar o agrupamento para identificar cada conjunto, que representa as diferentes formações que um exército pode ter, sendo que as suas

composições são independentes em relação às batalhas. Então, durante o jogo são observadas as unidades dos exércitos de ambos os jogadores, ou seja, são observados os elementos que compõem os grupos, sendo que o tipo da unidade tem o seu ponto forte e a sua fraqueza em relação a outros tipos de unidades. Com essas informações é possível aprender quais os melhores agrupamentos para enfrentar os grupos de unidades do adversário. Nos experimentos foram analisados os resultados dos agrupamentos, avaliando o conjunto de grupos que representam o exército do jogador, como um meio de predizer os resultados das batalhas. No trabalho descrito em (SYNNAEVE; BESSIERE et al., 2012) também foi criada uma heurística, que avalia as unidades para predizer quem provavelmente iria ganhar o confronto. O que foi observado nos resultados desse trabalho é que um exército bem montado pode ser melhor do que um com muitas unidades. Pela análise da formação dos exércitos ocorrer em modo *batch*, sem a necessidade de resultados em tempo real, configura-se uma importante diferença deste trabalho relacionado com a presente pesquisa, além da inexistência de uma análise sobre o comportamento do adversário, com o intuito de utilizar tal informação para a tomada de decisão do agente no StarCraft.

Em (STANESCU; ČERTICKỲ, 2014) é descrito um trabalho que tenta melhorar um jogador automático de jogos RTS especializado em batalhas. Neste trabalho foi descrito um método de predição da mais provável combinação de unidades produzidas por um oponente na composição de seu exército em certos períodos da partida de um jogo RTS. Este método foi aplicado nos jogos de StarCraft e Warcraft, ambos desenvolvido pela *Blizzard Entertainment*. O método de predição aplicado trabalha com as informações que podem ser obtidas pelo campo de visão do jogador, ou seja, somente o que está próximo das unidades do seu exército. Neste trabalho foi utilizado o *Answer Set Programming* (ASP) (HOTZ et al., 2014) para definir os conjuntos de unidades que representam diferentes exércitos, sendo que o objetivo é prever qual desses conjuntos será o usado pelo adversário na partida. Essa predição utiliza atributos como o custo para a construção das unidades, o tempo necessário para as construções e qualquer precondição do jogo para realizar as construções. Para testar a predição da mais provável combinação de unidades na formação do exército do adversário, para um dado momento do jogo, foram usados registros de jogos já concluídos. Os experimentos foram realizados considerando quatro momentos diferentes do jogo: 5 minutos, 10 minutos, 20 minutos e 30 minutos, após o início da partida. E foi observado que quanto mais longo o jogo, mais próxima fica a previsão de combinação de unidades do exército em relação ao que foi realmente usado pelo adversário. Diferentemente do que é apresentado na presente pesquisa, o trabalho descrito em (STANESCU; ČERTICKỲ, 2014) lida com aprendizagem sobre dados de um período do jogo para prever a formação do exército do adversário, contudo, os dados desse período do jogo são processados em *batch*, ou seja, esse trabalho não lida com os desafios da aprendizagem existentes em um fluxo contínuo de dados. Uma outra importante diferença, é que a previsão sobre a formação do exército do adversário não é usada estrategicamente no jogo, enquanto que este trabalho contempla alterações sobre as decisões do agente a partir da identificação de mudanças no comportamento do adversário.

Na pesquisa descrita em (BALLINGER; LIU; LOUIS, 2014) foram utilizadas técnicas de aprendizagem de máquina para identificar jogadores do jogo de RTS StarCraft II, e predizer as ações desses jogadores. Neste trabalho foram utilizados registros de jogos completos disponíveis na Internet, e que foram usados como um *replay*, simulando o andamento da partida. Foram utilizadas técnicas de aprendizagem de máquina disponíveis na ferramenta *Waikato Environment for Knowledge Analysis* (WEKA) (WITTEN et al., 2016) para identificar o jogador que atuava na partida e predizer suas ações. As técnicas utilizadas do WEKA foram: Árvore de Decisão, Rede Neural, Adaptative Boosting e Random Forest. Para atingir os objetivos foram utilizadas duas formas de extração de características, onde uma dessas formas foi utilizada na tarefa de identificação do jogador durante a partida, enquanto a outra era destinada à predição da próxima ação do jogador. No procedimento de extração de características para identificação do jogador, foram obtidas informações genéricas do jogo como: o tempo que durou a partida; o vencedor da partida; a relação entre quantidade de ações do jogador por minuto; a quantidade de unidades e estruturas construídas a cada três minutos; a informação sobre o momento em que cada unidade e estrutura foi construída pela primeira vez. No segundo processo de extração de características, o foco foi sobre informações com perfil mais estratégico, que permitiram prever a próxima ação do jogador como, por exemplo, a ordem de construção das estruturas, e os momentos em que foram realizados ataques. Então, com esses dois processos de extração de características foi possível identificar o jogador que atuava na partida (considerando um conjunto de 41 jogadores), e predizer a próxima ação a ser executada pelo jogador. Observa-se que o trabalho descrito em (BALLINGER; LIU; LOUIS, 2014) se difere da presente pesquisa por limitar seus objetivos à identificação do jogador que atua na partida, e à predição da próxima ação do jogador, não realizando nenhuma identificação sobre mudanças no estilo de jogo do jogador, e nem mesmo aplicando mudanças no módulo de decisões do jogo para que ele se adapte, estrategicamente, ao estilo de jogo do adversário.

No trabalho (DACHOWICZ et al., 2022) foi apresentado um *framework* que permite modelar de maneira eficiente missões para jogos RTS, construindo com eficiência um modelo de simulação do ambiente de missão. Esse *framework* combina técnicas de design de experimentos para coleta de dados, meta-modelagem com modelos de aprendizado de máquina e técnicas de quantificação de incerteza, além de IA explicável para validar o modelo explorando o ambiente da missão. O *framework* foi testado usando um jogo RTS de código aberto chamado microRTS. Um modelo de rede neural é então treinado com base nos dados obtidos dos experimentos modelados pelo *framework*, incorporando à rede uma forma de predição sobre o que pode ocorrer neste cenários. Os resultados mostram sucesso na captura de dados que refletem as características do jogo presentes nas missões modeladas.

Na próxima seção são detalhados os trabalhos relacionados a tarefa de processamento de dados obtidos de um fluxo contínuo, que em alguns casos envolve a detecção de novidades sobre grupos formados ou conceitos já existentes.

3.2 Trabalhos relacionados a processamento de dados em fluxo contínuo

As pesquisas relacionados a processamento de dados em fluxo contínuo, apresentadas nessa seção, podem ser divididas entre as seguintes categorias de pesquisa:

- ❏ Inovação na abordagem em que se aplica técnicas de aprendizagem de máquina sobre os dados obtidos do fluxo.
- ❏ Inovação na identificação de alguma forma de novidade sobre os dados obtidos do fluxo.

3.2.1 Trabalhos com destaque na abordagem em que se aplica técnicas de aprendizagem de máquina sobre os dados do fluxo

Dentre os trabalhos que inovam pela forma de aplicar técnicas de aprendizagem de máquina no processamento em tempo real dos dados que chegam de um fluxo contínuo, destaca-se o trabalho apresentado em (READ; PEREZ-CRUZ; BIFET, 2015), onde é descrita uma pesquisa no contexto de fluxo contínuo de dados que apresenta avanços no aprendizado incremental, usando memória constante, e utilizando dados não rotulados ou parcialmente rotulados. Nesta pesquisa foram apresentadas duas abordagens que têm como base a rede *Deep Belief Network* (DBN) (HINTON; OSINDERO; TEH, 2006), uma das técnicas desenvolvidas é o DBN-h, no qual um classificador "h" é criado a partir da saída da camada mais profunda da rede, como se ela fosse a fonte original dos dados de entrada. Então, com a saída da última camada verifica-se a existência de rótulos para estes dados. Se existir utiliza-se o rótulo do dado e o classificador é atualizado, caso contrário, o classificador trabalha com uma estimação para definir o rótulo. A segunda abordagem desenvolvida é o DBN-BP, na qual utiliza-se uma rede para predizer diretamente os rótulos, adicionando uma camada final, com uma dimensão equivalente a quantidade de rótulos conhecidos. Nesta segunda técnica utiliza-se o algoritmo *backpropagation* para refinar os pesos da rede. A presente pesquisa se distingue do que é apresentado em (READ; PEREZ-CRUZ; BIFET, 2015), pois este trabalho correlato não atua sobre a tomada de decisão de um agente, enquanto que a presente pesquisa visa atribuir uma semântica às mudanças identificadas pelo M-DBScan, e utilizar tal informação para guiar o módulo de decisão do StarCraft.

Em (HATAMIKHAH et al., 2018) é descrita uma pesquisa que considera que a mudança de conceito possui grande importância no cenário de fluxo de dados, e por isso propõe uma solução para aumentar a acurácia na identificação dessas mudanças utilizando um *Streaming Ensemble Algorithm* (SEA). O SEA recebe como entrada um fluxo de dados dividido em janelas que são processadas individualmente. Essa abordagem tem como base uma rede de aprendizagem profunda, a DBN, que é o modelo básico para o SEA. A DBN é construída com a chegada da primeira janela de dados, e esta DBN irá compor um conjunto de classificadores de tamanho fixo. Então, com a chegada de novas janelas de dados o modelo será atualizado gerando uma

nova versão do classificador (rede DBN) que também irá compor o conjunto de classificadores. Se o conjunto de classificadores treinados atingiu seu tamanho máximo, então o classificador menos eficiente será deletado e substituído por um melhor. Nesta abordagem utiliza-se o erro de classificação como indicador da ocorrência de uma mudança de conceito sobre o dado, enquanto as identificações de mudanças na presente pesquisa ocorrem sobre os dados no processo de agrupamento. Esse trabalho correlato também não tenta aprimorar um processo de tomada de decisão em tempo real, baseado em informação semântica, como almeja o presente trabalho com o jogo StarCraft.

Em (BODYANSKIY; TYSHCHENKO; KOPALIANI, 2015) é descrito um trabalho que lida com questões importantes no processamento de dados de um fluxo contínuo, como a não linearidade, a existência de dados não estacionários e a incerteza sobre a quantidade de grupos que poderão ser formados a partir dos dados que chegam do fluxo. Nesse trabalho correlato considera-se como boa solução para essas questões, o uso de um sistema computacional híbrido, composto pela técnica *Kohonem Self-organized map* (SOM) e o algoritmo de agrupamento *fuzzy C-means*. Então, um sistema neuro-fuzzy em cascata, para agrupamento *online*, foi proposto para lidar com tarefas neste cenário de incerteza acerca do processamento de dados obtidos de um fluxo contínuo. Cada nó da cascata resolve a tarefa de agrupamento independentemente dos outros, o que torna possível aumentar a velocidade de todo o processamento. Este trabalho se difere da atual pesquisa por não utilizar a informação gerada a partir do processamento dos dados do fluxo para nortear um processo de tomada de decisão, e por não aplicar uma técnica de detecção de mudanças sobre o agrupamento formado.

Na próxima seção são detalhados os trabalhos relacionados a tarefa de identificação de alguma forma de novidade sobre os dados do fluxo.

3.2.2 Trabalhos com destaque na identificação de alguma forma de novidade sobre os dados do fluxo

Dentre os trabalhos que se destacam pelo processo de identificação de novidade, há um trabalho diretamente relacionado com a presente pesquisa por utilizar o algoritmo M-DBScan, descrito em (VALLIM et al., 2013). Nessa pesquisa é apresentado um trabalho que por meio da modelagem do comportamento de um jogador de jogos digitais, é possível detectar eventuais mudança de comportamento do jogador, utilizando o algoritmo M-DBScan. O comportamento do jogador é representado por atributos que refletem a sua interação com o jogo, mas não considerando informações estratégicas para tomada de decisão no contexto do jogo. Os principais experimentos utilizaram dados de um jogo de primeira pessoa, e esses dados são compostos por atributos como a quantidade de troca de armas, o número de tiros disparados, frequência de utilização dos modos de tiro, entre outros. Dados referentes ao jogo StarCraft também foram usados nesse trabalho correlato, porém com o objetivo de mostrar a capacidade de generalização da técnica, podendo ser usada em outros problemas. Os dados usados do StarCraft foram gerados a

partir de *logs* de ações e não descreviam o cenário de jogo, como por exemplo, a quantidade de unidades mortas ou o poder de ataque do exército formado. O modelo criado para representar o jogador sofre alterações de modo a refletir o seu comportamento mais recente, que é uma das capacidades do algoritmo M-DBScan. Nesse trabalho foi usada a versão original do algoritmo M-DBScan, ou seja, ele não fornece um significado à mudança apontada pelo algoritmo. Uma outra diferença em relação ao trabalho apresentado em (VALLIM et al., 2013) se deve ao fato deste trabalho identificar novidades no estilo de jogo do jogador humano, usando para isso atributos que descrevem o seu modo de jogar. Enquanto que o presente trabalho, busca identificar mudanças no comportamento do adversário, usando atributos que descrevem o cenário de jogo a partir de informações obtidas do motor do StarCraft, possibilitando que a identificação de mudanças seja usada com o módulo de decisão do agente do StarCraft. Uma outra importante diferença se deve ao fato da presente pesquisa utilizar diferentes conjuntos de atributos para gerar uma melhor representação do cenário de jogo em diferentes momentos, e isso gera uma alteração no M-DBScan que consiste no uso de uma CM para cada momento representado.

Em (FARIA; GAMA; CARVALHO, 2013) foi descrita uma técnica para detecção de novidades para o cenário de fluxo contínuo de dados chamada *Multi-class Learning Algorithm for Data Stream* (MINAS). Essa técnica cria um modelo para representar classes e suas extensões, sendo que cada classe é representada por um conjunto de grupos. Caso o dado a ser classificado não é conhecido pelo modelo, então esse dado será classificado como desconhecido (*unknown*). Entre as principais contribuições do MINAS destacam-se: o modelo de decisão que representa os conceitos já conhecidos, configurando o MINAS como solução para um problema de múltiplas classes; o uso de amostras de dados não conhecidas pelo modelo, para aprender novos conceitos ou ampliar os conceitos já conhecidos, tornando o modelo dinâmico; o MINAS possui a capacidade de detectar diferentes novidades e fazer com que o modelo de decisão aprenda essas novidades; um único modelo de decisão é utilizado para aprender as classes já conhecidas do problema, e para aprender as novidades enquanto chega dados do fluxo; o MINAS consegue diferenciar a existência de uma novidade de um *outlier*. Em (FARIA; GAMA; CARVALHO, 2013) também foi proposto uma nova forma de avaliar o desempenho de métodos de detecção de novidades em problemas com várias classes. Esse método avalia o processo de classificação ao longo do fluxo, e não no final do processo de classificação em modo *batch*. O presente trabalho se diferencia do descrito em (FARIA; GAMA; CARVALHO, 2013) principalmente por questões relacionadas ao uso de uma técnica de detecção de mudanças no processo de tomada de decisões de um jogo RTS como o StarCraft. Também existe uma diferença quanto ao uso de múltiplas classes como ocorre no MINAS e a atribuição de uma semântica a uma mudança de comportamento na presente pesquisa, visto que a mudança de comportamento não indica o surgimento de novos conceitos (rótulos), além disso, essas mudanças terão seus valores semânticos definidos a partir de rótulos obtidos de um conjunto de dados já pré-definido.

No trabalho descrito em (HAQUE; KHAN; BARON, 2016) é proposto um *framework* denominado *Semi-Supervised Adptative Novel Class Detection and Classification over Data*

Stream (SAND), composto por um detector de novidades em classes para lidar com as mudanças de conceitos. No *framework* SAND há um conjunto de classificadores do tipo *K-Nearest Neighbors* (KNN), e ao invés de aplicar a detecção de mudanças na taxa de erro, ela é aplicada na estimativa de confiança do classificador para detectar mudanças de conceitos. Esse *framework* interpreta a existência de uma considerável presença de *outliers* que possuem forte coesão entre eles como sendo o surgimento de uma nova classe. Nesse trabalho foi apresentado um método não supervisionado para estimar a confiança do classificador na predição de rótulos das amostras de dados do fluxo, sendo que o classificador é atualizado utilizando uma quantidade limitada de amostras selecionadas, evitando um *overhead* pela reutilização das pontuações de confiança do classificador durante as predições. Com relação a presente pesquisa destaca-se a diferença quanto a mudança de comportamento do adversário não estar relacionada a novas classes como faz este trabalho correlato, uma vez que uma semântica é definida a partir de um conjunto de amostras de dados rotuladas, mas nenhum rótulo novo irá surgir no processo. Além disso, as mudanças de comportamento estão vinculadas ao dado e ao grupo ao qual ele pertence, mas o significado da mudança será definido pela proximidade do dado que desencadeou a mudança em relação aos rótulos já pré-estabelecidos.

Na próxima seção são detalhados os trabalhos que utilizam classificação em dados obtidos de um fluxo contínuo, destacando as suas diferenças com a presente pesquisa, especialmente quanto a tarefa de atribuir uma semântica a uma mudança de comportamento identificada.

3.3 Trabalhos que utilizam classificação no cenário de fluxo de dados

Apesar dos trabalhos relacionados não serem capazes de junto da identificação de mudanças no cenário de fluxo contínuo de dados, também atribuir uma semântica a alguma mudança identificada, muitos desses trabalhos utilizam da classificação para definir o significado isolado de uma amostra de dados, o que não representa o significado de uma mudança. Esses trabalhos serão apresentados em função de dois grupos:

- ❏ Classificação de dados obtidos de um fluxo contínuo;
- ❏ Uso de classificação para detecção de mudança de conceito.

3.3.1 Trabalhos relacionados que classificam dados obtidos de um fluxo contínuo

A classificação de dados de um fluxo contínuo é um problema de predição de classe de cada amostra de dado não rotulada do fluxo (GOMES et al., 2017). Muitos algoritmos de classificação em fluxo contínuo de dados foram desenvolvidos como o apresentado em (JALALI et al., 2016), em que descreve o comportamento humano usando dados obtidos como fluxo de um

smartphone. Esses dados são compostos por atributos como: tempo; atividade como correr, caminhar ou ficar parado; localização; configuração de som; aplicativo usado no smartphone; estado do telefone. Em seguida, os dados foram convertidos em uma representação de eventos como: estudar; adormecido; transporte de veículos; andando; interagindo com o telefone; verificação de e-mail; e outros. Para identificar os eventos foi utilizada a técnica *Formal Concepts Analysis* (FCA), que é uma técnica de estrutura de conceito baseada na fusão de dados.

Outra pesquisa é apresentada em (GUTIERREZ-GALAN et al., 2018), e descreve um trabalho que utiliza uma rede neural para classificar o comportamento animal, utilizando para isso, dados obtidos de um sistema de monitoramento baseado em uma rede de sensores sem fio e um dispositivo de coleira inteligente colocada nos animais. A rede neural foi embutida na coleira do animal, permitindo uma classificação do comportamento em tempo real. Em (FAWCETT; PROVOST, 1999) é descrito o uso de um *framework* no problema de monitoramento de atividades, que envolve monitorar o comportamento por meio de celulares para detectar fraudes. O *framework* usado codifica semelhanças de tarefas, apontando as diferenças que pode ser indício de fraude, sendo utilizado o *Receiver Operating Characteristic* (ROC) como método de análise, que após algumas modificações, foi criada a técnica *Activity Monitor Operating Characteristic* (AMOC). O trabalho descrito em (READ; PEREZ-CRUZ; BIFET, 2015), que já foi apresentado na seção 3.2.1, também se encaixa nessa categoria por lidar com classifcação em fluxo contínuo de dados. Uma abordagem desenvolvida em (READ; PEREZ-CRUZ; BIFET, 2015) é o DBN-h, na qual um classificador "h" é criado a partir da saída da camada mais profunda da rede DBN como se fosse a fonte original dos dados de entrada. Então, se houver um rótulo para esses dados, ele é usado e o classificador é atualizado, senão, utiliza uma estimativa para definir o rótulo.

O trabalho apresentado em (XIANG et al., 2021), propõe uma abordagem de detecção de mudanças em imagens utilizando o *Siamese Convolution Network*. Neste trabalho, a semântica atribuída a estas mudanças correspondem às alterações observadas nas imagens obtidas em diferentes momentos, sem realizar uma análise sobre a distribuição dos dados que causaram a mudança. Os rótulos usados nesse trabalho, chamados de semânticas, são os seguintes: *no change*, *low vegetation*, *non-vegetation ground surface*, *tree*, *water*, *buildings* e *playground*. Esses rótulos são usados para categorizar regiões das imagens, eles não dão um significado a mudança. As mudanças neste trabalho relacionado, referem-se as alterações nestes rótulos associados às regiões das imagens. Essa categorização se baseia no que é visto nas imagens, não utilizando atributos para representar todo um contexto do problema, como ocorre com a presente pesquisa.

Os trabalhos descritos em (GOMES et al., 2017; JALALI et al., 2016; GUTIERREZ-GALAN et al., 2018; FAWCETT; PROVOST, 1999; READ; PEREZ-CRUZ; BIFET, 2015; XIANG et al., 2021) apresentam uma diferença significativa para o presente trabalho, pois eles não identificam mudanças monitorando a distribuição dos dados, nem tentam atribuir um significado a uma mudança. Pois, classificar uma amostra de dado não fornece o significado que

representa a tendência semântica de uma sequência de dados que venha a provocar uma mudança ao longo do tempo.

Na seção seguinte serão descritos os trabalhos correlatos que lidam com a classificação para a tarefa de detectar mudanças de conceitos, além disso, serão destacadas as diferenças desses trabalhos com a presente pesquisa.

3.3.2 Trabalhos relacionados que que utilizam classificação para detecção de mudança de conceito

Em cenários não estacionários, onde a distribuição de dados pode mudar ao longo do tempo, as técnicas precisam detectar e se a adaptar às mudanças de conceito implícitas ou explícitas (GOMES et al., 2017). Contudo, no caso de detecção de mudança nos referimos às técnicas capazes de lidar com mudanças explícitas de conceitos (GAMA et al., 2014).

De acordo com (GAMA, 2010), as técnicas de detecção de mudança podem ser classificadas em duas categorias principais: i) monitoramento da evolução dos indicadores de performance e ii) monitoramento de distribuições em diferentes janelas de tempo.

Na primeira categoria, alguns indicadores são monitorados ao longo do tempo. A técnica *Drift Detection Method* (DDM) (KRAWCZYK et al., 2017) é um dos algoritmos mais conhecidos desta categoria. Ele assume que o erro do classificador e seu desvio padrão deve cair a medida que mais dados são recebidos do fluxo. Então, se ocorrer do erro do classificador aumentar, a técnica DDM indica a ocorrência de uma mudança de conceito. A técnica *Early Drift Detection Method* (EDDM) é uma extensão do DDM para tratar mudanças graduais (KRAWCZYK et al., 2017). Ambas as abordagens são supervisionadas porque elas consideram o rótulo das amostras para calcular o erro do classificador, que é monitorado para indicar a mudança de conceito.

Ainda considerando a primeira categoria, as técnicas *Cumulative Sum approach* (CUSUM) (PAGE, 1954) e *Page-Hinkley* (SEBASTIAO; GAMA, 2009) estão entre as principais técnicas de detecção de mudanças. O CUSUM lida com a atualização do próprio modelo e dispara um alarme sempre que a média de um certo parâmetro, relacionado ao dado de entrada, apresentar uma mudança que excede um limite pré-definido. Este parâmetro pode ser, por exemplo, o erro do classificador, calculado com base nas amostras rotuladas da base de treinamento. O *Page-Hinkley* é uma variação do CUSUM que também monitora a evolução de algum parâmetro associado aos dados obtidos do fluxo, atualizando o próprio modelo sempre que há uma mudança significativa no sinal da Gaussiana relacionado ao parâmetro monitorado. Essa técnica pode ser usada em problemas de inspeção, onde amostras de dados são examinadas em momentos predefinidos, buscando por variações no parâmetro analisado.

Um trabalho já mencionado em outro grupo de trabalhos relacionados, mas que também se enquadra nesta primeira categoria é o descrito em (HATAMIKHAH et al., 2018) que utiliza o algoritmo SEA para aumentar a acurácia na identificação de mudanças de conceito. Além

disso, utiliza-se da rede DBN para compor um conjunto de classificadores de tamanho fixo, e o classificador menos eficiente será substituído por um melhor.

A segunda categoria inclui métodos que comparam a distribuição sobre janelas de dados distintas, onde a primeira sumariza informações passadas e a segunda armazena as amostras de dados mais recentes. Testes estatísticos são usados para comparar ambas as distribuições. Se a hipótese nula é rejeitada, então a mudança foi detectada. Um exemplo de algoritmo que se encaixa nessa categoria é o *Adaptive Window Algorithm* (ADWIN) (BIFET; GAVALDA, 2007), que foi originalmente criado para dados unidimensionais. Contudo, a técnica pode ser adaptada para cenário de dados multidimensionais a partir de múltiplas execuções, uma para cada dimensão.

É importante salientar que os trabalhos (KRAWCZYK et al., 2017; PAGE, 1954; SEBASTIAO; GAMA, 2009; BIFET; GAVALDA, 2007) não são capazes de identificar o significado de mudanças observáveis na distribuição dos dados de uma sequência, como propõe a presente pesquisa.

Como forma de apresentar de forma organizada as características dos trabalhos relacionados foram definidas as características a seguir, que serão relacionadas na Tabela 1:

- ❏ C_1:Utiliza técnicas de aprendizado de máquina para gerenciar ações a serem performadas em jogo RTS;
- ❏ C_2:Utiliza técnicas de aprendizado de máquina para realizar predições no contexto de jogos RTS;
- ❏ C_3:Utiliza técnicas de aprendizado de máquina no cenário de fluxo contínuo de dados;
- ❏ C_4:Detecção de novidades no cenário de fluxo contínuo de dados;
- ❏ C_5:Classificação de dados obtidos de um fluxo contínuo;
- ❏ C_6:Uso de classificação para detectar mudança de conceito.

3.4 Considerações finais

Neste capítulo foram descritos os trabalhos correlatos, que foram divididos em três grupos principais: os trabalhos relacionados a aprimoramentos de jogos RTS, como o StarCraft; os trabalhos relacionados a processamento de dados que chegam na forma de fluxo contínuo; e os trabalhos que rotulam dados na tentativa de gerar uma semântica.

Dentre os trabalhos relacionados a jogos RTS destacam-se pesquisas dedicadas a aprimorar o processo de decisão dos jogos, sendo que essas decisões podem envolver ações de confronto no jogo ou ações gerenciais como a definição da ordem de construção de estruturas importantes no jogo. Também são destaques os trabalhos relacionados a jogos RTS que aplicam alguma forma de predição. Essas predições podem ser aplicadas para prever as próximas ações do

Tabela 1 – Resumo das características dos trabalhos relacionados

Referência do trabalho relacionado	C_1	C_2	C_3	C_4	C_5	C_6
(JUSTESEN; RISI, 2017)	X					
(SHAO; ZHU; ZHAO, 2017)	X					
(PENG et al., 2017)	X					
(The AlphaStar team, 2019)	X					
(WEBER; MATEAS, 2009)		X				
(SYNNAEVE; BESSIERE et al., 2012)		X				
(STANESCU; ČERTICKỲ, 2014)		X				
(HOTZ et al., 2014)		X				
(BALLINGER; LIU; LOUIS, 2014)		X				
(WITTEN et al., 2016)		X				
(DACHOWICZ et al., 2022)		X				
(READ; PEREZ-CRUZ; BIFET, 2015)			X		X	
(HATAMIKHAH et al., 2018)			X			X
(BODYANSKIY; TYSHCHENKO; KOPALIANI, 2015)			X			
(VALLIM et al., 2013)				X		
(FARIA; GAMA; CARVALHO, 2013)				X		
(HAQUE; KHAN; BARON, 2016)				X		
(GOMES et al., 2017)					X	
(JALALI et al., 2016)					X	
(GUTIERREZ-GALAN et al., 2018)					X	
(FAWCETT; PROVOST, 1999)					X	
(XIANG et al., 2021)					X	
(KRAWCZYK et al., 2017)						X
(PAGE, 1954)						X
(SEBASTIAO; GAMA, 2009)						X
(BIFET; GAVALDA, 2007)						X

jogador ou mesmo para adivinhar como será a composição do exército adversário, que pode influenciar nas decisões do agente quanto a composição do seu próprio exército.

Em relação aos trabalhos sobre processamento de dados de fluxo contínuo existe um subgrupo contendo pesquisas que se destacam quanto a abordagem em que técnicas de aprendizagem de máquina foram aplicadas sobre os dados que chegam do fluxo, mantendo o processamento em tempo real e conseguindo lidar com o uso de um recurso limitado como a memória. Outros trabalhos relacionados destacam-se quanto a forma como processaram os dados com o objetivo de detectar alguma novidade, que pode estar vinculada a um conceito já existente, o surgimento de uma nova classe, ou mesmo uma novidade sobre o agrupamento gerado.

Com relação aos trabalhos que classificam dados do fluxo ou utilizam um classificador para detectar o surgimento de um novo conceito, salienta-se que nenhuma dessas técnicas tenta atribuir uma semântica a uma mudança, e que a classificação de um dado isolado não reflete a tendência semântica de uma sequência de dados.

No próximo capítulo são apresentados detalhes da proposta, destacando-se como essa pes-

quisa tem sido desenvolvida, além das contribuições de cada etapa realizada.

CAPÍTULO 4

Etapa 1: Viabilidade do M-DBScan com dados do StarCraft

Esta etapa da pesquisa visa estudar a viabilidade do uso do algoritmo M-DBScan na identificação de mudanças no comportamento do adversário no jogo RTS chamado StarCraft. Esta identificação ocorrerá com base em informações que retratam os cenários de jogo e, consequentemente, o comportamento de todos os jogadores envolvidos no confronto.

Esta etapa 1 foi avaliada por meio de experimentos que utilizam bases de dados geradas por dados do jogo StarCraft. Cada amostra de dados nestas bases é formada por atributos que visam representar o cenário de jogo, possibilitando a identificação de mudanças de comportamento do adversário. Na Seção 4.2 estão descritos os experimentos e resultados desta etapa.

Na seção seguinte está descrito o método de desenvolvimento desta primeira etapa da pesquisa.

4.1 Desenvolvimento da etapa 1

No jogo StarCraft, os jogadores podem controlar diferentes raças, que possuem diferentes tipos de unidades que, por sua vez, possuem características como forças e fraquezas, que variam em intensidade. Em virtude dessas variações, a qualidade do exército de um jogador pode ser avaliada com base nas unidades que o compõem, tornando possível obter o poder de ataque e de defesa desse exército, bem como a intensidade do dano que ele pode causar ou sofrer.

A etapa 1 desta pesquisa usará informações relativas à composição do exército adversário, as quais servirão para apontar qualquer mudança significativa no seu estilo de jogo. Então, o M-DBScan lidará com dados que descrevem o inimigo por meio dos atributos presentes nas amostras de dados, sendo que a maioria desses dados serão obtidos a partir do momento em que ocorrem confrontos, visto que este é o momento que proporciona uma visão mais rica sobre o adversário.

Diferentemente do trabalho que utilizou o M-DBScan para identificar mudanças de comportamento com base em atributos que descreve o modo de jogo de um jogador humano, nesta etapa

da pesquisa serão considerados atributos que descrevem o cenário de jogo, os quais sofrem os efeitos das decisões de todos os jogadores envolvidos na partida. Por exemplo, o atributo *unidades mortas do inimigo* está diretamente relacionado ao potencial de defesa do adversário e ao potencial de ataque do agente. Assim sendo, a partir dessa representação com atributos que envolvem todos os jogadores, deseja-se identificar mudanças no comportamento de apenas um deles, no caso, o adversário do agente.

Uma vez que se deseja detectar mudanças no estilo de jogo do adversário, cada atributo foi selecionado adotando a perspectiva desse oponente. Para tanto, foram criadas bases de dados a partir de disputas controladas no StarCraft. Cada amostra das bases de dados usada na etapa 1 é composta pelos seguintes atributos:

- Atributo 1 (Unidades mortas do inimigo): Fornece a quantidade de unidades mortas do inimigo até o momento corrente.
- Atributo 2 (Poder de ataque): Representa o dano que o exército adversário pode causar. Ele é calculado considerando o dano de cada unidade do exército adversário visível para o agente.
- Atributo 3 (Poder de defesa): Representa o poder de defesa do exército adversário. Ele é calculado considerando o nível de defesa de cada unidade do exército adversário visível para o agente.
- Atributo 4 (Dano em construções): Representa o quanto de dano o exército adversário causou às estruturas do agente. Este atributo é obtido em função do quanto que foi tirado daquilo que é considerado a "*vida da estrutura*".

Cada passo do algoritmo M-DBScan está ilustrado no fluxograma apresentado na Figura 12. O algoritmo começa buscando por uma amostra de dado do fluxo (seta #1), caso encontre (seta #2), o dado é apresentado como entrada para a fase *online* do M-DBScan (seta #3), como visto na Seção 2.5.4. Posteriormente, o fator de decaimento é aplicado a cada *p-micro-cluster* e a cada *o-micro-cluster* ao qual não foi atribuído o mais recente dado do fluxo (seta #4). Em seguida, o algoritmo checa se é o momento de executar uma verificação de *micro-clusters* (seta #5), tal verificação deve ser feita em intervalos de tempo pré-estabelecidos. Caso seja, (seta #6), qualquer *p-micro-cluster* que deixe de atender ao requisito exigido para ser *p-micro-cluster* (veja Seção 2.5.2) volta a ser um *o-micro-cluster*. Em seguida, qualquer *o-micro-cluster* que deixe de atender aos requisitos mínimos de um *micro-cluster* (veja Seção 2.5.3) é excluído (seta #7). Na sequência, a fase *offline* do M-DBScan é executada (seta #8) (Seção 2.5.4). Por outro lado, caso o teste (seta #5) fracasse, ou seja, não é o momento para verificação de *micro-clusters*, então a fase *offline* do M-DBScan será executada diretamente após a aplicação do fator de decaimento (seta #9). Uma vez concluída a fase *offline* (seta #10), calcula-se a entropia, temporal (equação 13) ou espacial (equação 10). Em seguida, o limiar da entropia usada é calculado (seta #11) (equação 16). Então, verifica-se se uma mudança de comportamento foi identificada

no último intervalo equivalente ao tamanho da janela deslizante (W) (seta #12). Caso não tenha, efetua-se uma comparação entre a entropia e seu limiar (seta #13). Se a entropia superar o seu limiar, então a novidade é detectada e registrada na janela deslizante (seta #14). Em seguida (seta #15), ativa-se o módulo de detecção de mudança de comportamento onde é feita uma verificação quanto à ocorrência da mínima quantidade de novidades em um intervalo de tempo, que é necessária para caracterizar uma mudança de comportamento. O mesmo módulo de mudança de comportamento também controla a janela deslizante (veja Seção 2.5.5). Na sequência, verifica-se a existência de uma nova amostra de dados no fluxo (seta #16). Contudo, se nenhuma novidade for detectada (seta #17), então a verificação da existência de uma nova amostra de dado ocorre diretamente, sem passar pelo módulo de detecção de mudança de comportamento. Caso não haja mais dado a ser processado, o algoritmo é finalizado (seta #19). Caso contrário, a execução do algoritmo se repete (seta #2). Se uma mudança tiver sido detectada no último intervalo de tamanho W (seta #18), o processo de detecção de novidades será ignorado por um período igual a W, mas os dados continuarão sendo lidos do fluxo.

Os experimentos realizados na etapa 1 serão apresentados na Seção 4.2. Conforme mencionado anteriormente, a etapa 1 avalia a viabilidade da aplicação do M-DBScan aos dados do jogo StarCraft, que nos experimentos serão representados pelos quatro atributos apresentados no início desta seção.

4.2 Experimentos da etapa 1

Nesta seção está descrito o método de avaliação, os experimentos, os resultados e suas análises efetuados sobre a etapa 1 apresentada neste capítulo 4.

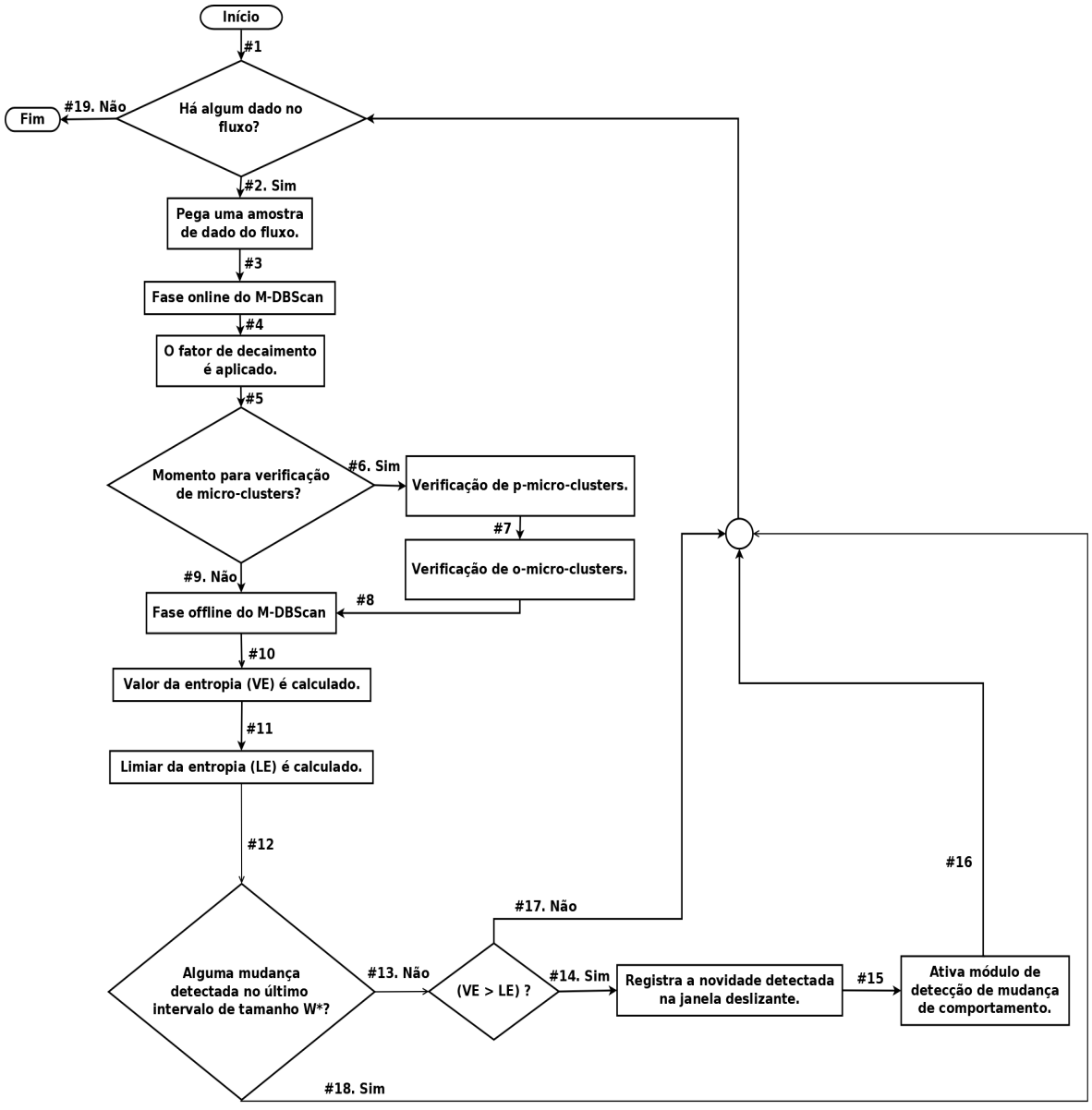
A fim de mostrar a eficiência e viabilidade do M-DBScan na identificação de mudanças de comportamento em partidas do StarCraft, foi usado um conjunto de cenários de teste representando situações distintas de jogo.

Os resultados dos experimentos são apresentados utilizando as métricas falso positivo (FP), falso negativo (FN) e verdadeiro positivo (VP). Considerando que existe um intervalo de m amostras entre duas mudanças de comportamento, os atrasos na detecção de mudanças de comportamento são indicados quando a detecção de uma mudança ocorrer após a chegada de 10% até 30% de m. As métricas citadas são apresentadas para cada uma das medidas de entropia (temporal e espacial).

Os parâmetros do algoritmo M-DBScan e das entropias são os seguintes (os parâmetros foram descritos na Seção 2.5):

- Do M-DBScan são: $\mu = 9$; $\beta = 0.1$; $\varepsilon = 20$; $\lambda = 0.03$.

- Dos experimentos com a entropia temporal são: quantidade mínima de novidade = 2; tamanho da janela deslizante = 20; $\eta_t = 0{,}05$; $\gamma = 0{,}05$; $\delta = 0{,}03$; $\theta = 3$.



** W representa o tamanho da janela deslizante.*

Figura 12 – Fluxograma do M-DBScan.

Fonte: Adaptado de (VIEIRA; JULIA; FARIA, 2019)

- Dos experimentos com a entropia espacial são: quantidade mínima de novidade = 2; tamanho da janela deslizante = 20; $\eta_s = 0{,}05$; $\gamma = 0{,}05$; $\delta = 0{,}002$; $\theta = 3$.

Os diferentes parâmetros usados na etapa de agrupamento e cálculo de entropia possuem influência direta nos resultados das detecções de mudanças de comportamento. Para definir os valores dos atributos da etapa de agrupamento precisa-se considerar valores que garantam que micro-clusters semelhantes venham a fazer parte de um mesmo grupo no agrupamento final, visto que o agrupamento final definirá os estados da CM, impactando os cálculos de entropia. Por sua vez, os parâmetros vinculados à entropia devem ser definidos considerando a intensidade dos ajustes do seu valor, evitando que poucas amostras de dados sejam suficientes para ultrapassar o limiar da entropia. Por outro lado, deve-se evitar ajustes muito pequenos que demandariam uma quantidade muito grande de amostras para que o valor de entropia ultrapasse seu limiar.

Os dados utilizados nos experimentos foram obtidos em partidas controladas do StarCraft, que permitiram reproduzir cenários específicos envolvendo diferentes comportamentos. Cada amostra de dados é extraída a cada 15 frames de percepção do jogo, considerando os atributos apresentados neste capítulo. O uso de 15 frames visa garantir um intervalo em que as amostras na base de dados apresentem diferenças entre si, e que sejam suficientes para provocar uma significativa variação nos valores dos atributos em diferentes momentos. Nos testes investigativos, quando foi usada uma quantidade menor de frames, tal como 5, por exemplo, foi necessário o registro de dezenas de amostras idênticas antes que se obtivessem amostras que produzissem alguma diferença nos valores dos atributos.

A fim de criar cenários de teste mais complexos, bem como conhecer o momento em que uma mudança de comportamento ocorre, foram jogadas várias partidas do jogo StarCraft, cada uma delas composta por ações do adversário que se enquadram em um mesmo padrão de comportamento ao longo de toda a partida. Assim sendo, as bases de dados foram criadas a partir da justaposição de sequências de dados provenientes de partidas distintas. Logo, o ponto de junção entre as bases de dados é o momento em que ocorre a mudança de comportamento, caracterizando mudanças abruptas. Nos experimentos da etapa 1, todo comportamento é representado por uma sequência de 1000 amostras de dados, que são apresentadas como entrada para o algoritmo M-DBScan como um fluxo contínuo.

4.2.1 Análise dos experimentos da etapa 1

Os cenários de teste criados e os resultados obtidos com os experimentos são os seguintes:

- Cenário 1: Este é um teste simples, com apenas uma mudança de comportamento, na qual o adversário interrompe a criação de unidades para compor o exército, focando na extração de recursos e construção de novas estruturas.

  Como pode ser observado na Tabela 2, o M-DBScan, com ambas as entropias, foi capaz de identificar a mudança de comportamento esperada, sendo que em nenhuma situação

ocorreu atraso na detecção. A precisão na detecção de mudanças, para cada entropia, é igual a 1, assim como a revocação e o *F1-Score*.

- Cenário 2: Neste cenário de teste ocorrem três mudanças de comportamento representando as seguintes situações: criação de unidades para aumentar o poder de ataque do exército; foco nos confrontos matando unidades; e, por fim, um novo aumento no poder de ataque do exército.

  Os resultados com o cenário de teste 2, apresentados na Tabela 2, mostram que o M-DBScan, utilizando as duas entropias, foi capaz de identificar todas as mudanças esperadas, apresentando valor 1 para as medidas de precisão, revocação e *F1-Score*. Destaca-se que, em nenhum caso, ocorreu atraso na detecção.

- Cenário 3: Neste cenário, inicialmente o adversário concentra-se em construir unidades próprias para combate. Em seguida, ocorre uma mudança, que aumenta o poder de defesa do exército (por meio de uma atualização no nível de defesa de suas unidades). Depois, em uma segunda mudança, aumenta-se o poder de ataque do exército, também por meio de um atualização nas suas unidades.

  Como pode ser visto na Tabela 2, o M-DBScan, com ambas as entropias, foi capaz de identificar as mudanças esperadas, sendo que com a entropia temporal houve a ocorrência de um falso positivo. Além disso, com a entropia temporal a precisão foi 0,66, a revocação foi 1, e o *F1-Score* foi 0,8, indicando um bom balanceamento entre as medidas. A precisão, a revocação e o *F1-Score* na identificação de mudanças utilizando entropia espacial foi 1. Em nenhuma situação ocorreu atraso na detecção.

- Cenário 4: Neste cenário o adversário irá alternar dois comportamento, chamados de B1 e B2. O comportamento B1 será focado no ataque, buscado provocar o maior número de mortes no exército do agente, enquanto que o comportamento B2 consiste em atacar as estruturas construídas pelo agente. Neste cenário haverá uma alternância e repetição dos comportamentos B1 e B2, sendo repetido 3 vezes, gerando um total de 6 mudanças de comportamento: Comportamento inativo (aguardando início do confronto) $\rightarrow$ B1 $\rightarrow$ B2 $\rightarrow$ B1 $\rightarrow$ B2 $\rightarrow$ B1 $\rightarrow$ B2.

Tabela 2 – Resultados dos cenários de testes da etapa 1

	Entropia Temporal				**Entropia Espacial**			
	FP	**FN**	**VP**	***F1-score***	**FP**	**FN**	**VP**	***F1-score***
Cenário de teste 1	0	0	1	1	0	0	1	1
Cenário de teste 2	0	0	3	1	0	0	3	1
Cenário de teste 3	1	0	2	0,8	0	0	2	1
Cenário de teste 4	1	3	3	0,6	0	2	4	0,8
	Média *F1-score*			**0,85**	**Média *F1-score***			**0,95**

Os resultados referentes ao cenário 4, apresentados na Tabela 2, mostram que o M-DBScan, com nenhuma das entropias, foi capaz de identificar todas as mudanças de comportamento do teste. Com a entropia temporal, o M-DBScan identificou três mudanças de comportamento sem atrasos, tendo registrado um falso positivo. Analisando o experimento, observou-se que as mudanças detectadas foram as três primeiras. No caso em que o M-DBScan usa a entropia espacial, foram identificadas as três primeiras mudanças de comportamento, bem como a última mudança de comportamento B2 para B1, todas sem atraso. Destaca-se que após as três primeiras mudanças de comportamento, considerando a configuração da CM e a probabilidade nas suas transições, atingiu-se valores de entropia tais que, mesmo com a chegada de mais dados, vinculados às mudanças de comportamento seguintes, não foi possível superar o limiar das entropias e provocar a detecção das mudanças que foram registradas na Tabela 2 como FN.

A precisão na identificação das mudanças de comportamento, usando a entropia espacial, foi igual a 1; a revocação está em torno de 0,66; o *F1-Score* foi de 0,8, demostrando que a entropia espacial é confiável na detecção de comportamento mesmo em cenários mais complexos. Com relação à entropia temporal, a precisão foi 0,75; a revocação foi 0,5; o *F1-Score* foi de 0,6, o que prova que ela, apesar de não ter apresentado o mesmo desempenho que a entropia espacial, ainda assim apresentou resultados aceitáveis no processo de detecção de mudanças de comportamento.

Pela análise dos experimentos, observa-se que, quando a entropia espacial tem uma elevação no seu valor, isso indica que há um aumento de heterogeneidade nos dados que chegam do fluxo. Por outro lado, em situação em que há uma redução no valor da entropia espacial, os dados apresentam maior similaridade. Caso esta última situação permaneça por muito tempo, haverá efeitos nos grupos formados, pois como poucos grupos ou somente um recebe os dados, os demais grupos vão sofrer por mais tempo com o fator de decaimento, tendendo ao desaparecimento.

Valores baixos da entropia temporal sugerem que as amostras de dados não tem sido distribuídas entre grupos distintos, indicando que por um certo intervalo de tempo, as amostras de dados estão concentradas em uma quantidade pequena de grupos, ou mesmo em um único grupo. Contudo, se ocorre um aumento no valor da entropia temporal, tem-se uma indicação de que diferentes transições da CM foram ativadas, o que aponta para o fato de que os dados estão sendo distribuídos entre diferentes grupos, decorrente da baixa similaridade dos dados.

O teste de Wilcoxon (DERRAC et al., 2011) foi aplicado para comparar a performance do M-DBScan utilizando a entropia temporal e a espacial. A ferramenta utilizada para aplicação do teste de Wilcoxon foi a KELL (ALCALÁ-FDEZ et al., 2011).

O teste de Wilcoxon com o M-DBScan utilizou a seguinte hipótese nula: *O M-DBScan com entropia espacial apresentou o mesmo desempenho que o M-DBScan com entropia temporal*. O teste de Wilcoxon apresentou R^+ igual a 8,5 e R^- igual a 1,5, com assintótico p-valor igual a 0,1441, que não permite rejeitar a hipótese nula e afirmar que o M-DBScan com alguma das entropias usadas é estatisticamente melhor que a outra. Apesar disso, o M-DBScan com

entropia espacial apresenta uma média de *F1-score* igual a 0,95 contra 0,85 da versão com entropia temporal.

4.3 Considerações finais

Pelo que foi apresentado neste capítulo, observou-se que os experimentos realizados na etapa 1 da pesquisa conseguiram validar o uso do M-DBScan na detecção de mudança de comportamento do adversário de um agente do StarCraft, em situações em que os dados do fluxo (cenários de jogo) são representados por atributos que retratam a atuação dos jogadores envolvidos no confronto. Isso demonstra que é viável o uso do M-DBScan em cenários dinâmicos como o observado em jogos. Validando a primeira hipótese apresentada na Seção 1.3.

Os experimentos nesta etapa 1 reproduziram diferentes cenários, ilustrando as mudanças de comportamento do adversário. Os resultados obtidos com a detecção de mudanças foram satisfatórios utilizando ambas as entropias, e por meio de um teste estatístico, observou-se que não é possível afirmar que uma é melhor que a outra, considerando todos os cenários de teste. Mas analisando isoladamente o cenário de teste mais complexo (cenário 4), a entropia espacial se saiu um pouco melhor.

A partir da etapa 1, surgiu a hipótese que desencadeou a segunda etapa executada nessa pesquisa, na qual se assume que os atributos sofrem variações nos seus níveis de relevância em função do momento do jogo. Então, adequar a representação via atributos com base no momento do jogo pode aumentar a precisão do algoritmo M-DBScan nas indicações de mudança de comportamento. Essa segunda etapa é detalhada no próximo capítulo.

CAPÍTULO 5

Etapa 2: Aplicação de diferentes atributos para diferentes estágios do jogo StarCraft

A etapa 2 da pesquisa ocorreu com o objetivo de investigar os ganhos que se pode ter por utilizar diferentes conjuntos de atributos para representar, mais adequadamente, diferentes momentos do jogo StarCraft. Uma outra questão a ser respondida com o uso desta nova representação, está relacionada ao impacto na acurácia do M-DBScan, quanto a identificação de mudanças de comportamento. Afim de avaliar esta etapa da pesquisa, diferentes bases de dados foram criadas para representar diversos cenários de jogo. Os experimentos estão descritos na Seção 5.2.

Na seção seguinte será descrito em detalhes o método de desenvolvimento desta etapa da pesquisa.

5.1 Desenvolvimento da etapa 2

O estudo da etapa 2 visa identificar o impacto do uso de diferentes conjuntos de atributos, cada um retratando especificidades de fases relevantes do jogo, que podem ter impacto na acurácia da detecção de novidades. Salienta-se que tais novidades são decorrentes de significativas alterações no cenário de jogo que, eventualmente, podem indicar uma mudança de comportamento do adversário.

Os estágios de jogo aqui considerados são:

- *Estágio de formação da batalha* (EFB), no qual os adversários iniciam o conflito;
- *Estágio intermediário da batalha* (EIB), no qual mortes começam a ocorrer em qualquer um dos exércitos envolvidos no confronto;
- *Estágio de conclusão da batalha* (ECB), o qual é caracterizado por uma expressiva redução no número de unidades em confronto de qualquer um dos exércitos envolvidos na batalha.

Para representar esses diferentes estágios de jogo, são usados conjuntos distintos de atributos extraídos de um total de dezesseis atributos, sendo que essa seleção ocorreu de maneira empírica, considerando a relevância dos atributos nos estágios do jogo. Dessa forma, durante o processo de detecção de mudança no estilo de jogo do adversário, a presente proposta utiliza três distintas CMs, uma para cada estágio de jogo. O uso de distintas CMs se justifica pelo fato de que os estágios são representados por conjuntos com diferentes quantidades de atributos, dessa maneira, as amostras de dados utilizadas em cada CM seguirão a correspondente representação.

O uso de várias CMs não é adequado caso deseja-se manter as informações já registradas na CM. Visto que a cada nova CM, toda a informação presente nas probabilidades das transições serão apagadas a partir do momento em que uma nova CM é inicializada.

A Figura 13 ilustra o processo de atribuição do dado obtido do fluxo para a CM correta.

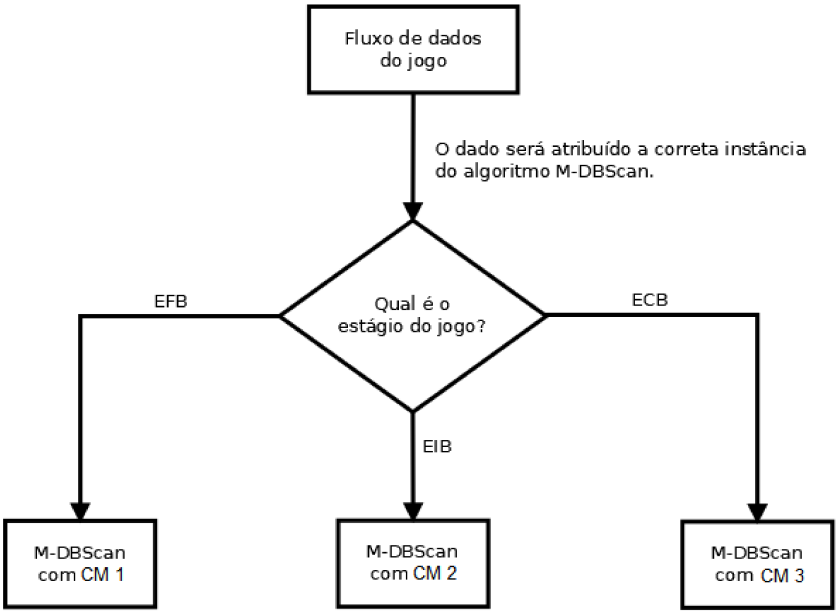


Figura 13 – Seleção da versão apropriada da CM.

Fonte: Elaborada pelo autor.

Na subseção a seguir será apresentado como ocorreu a extração de atributos nesta etapa da pesquisa.

5.1.1 Extração dos atributos

Para a criação dos atributos usados nesta etapa, foram feitas considerações sobre os tipos de unidades, de modo a definir grupos em função do seu poder de ataque. Esses grupos foram gerados por meio da discretização dos intervalos de valores de poder de ataque, que produziu os seguintes grupos de unidades:

- Tipo 1, composto pelas unidades com poder de ataque baixo;
- Tipo 2, composto pelas unidades com poder de ataque médio;

❑ Tipo 3, composto pelas unidades com poder de ataque alto.

Cada amostra de dados é formada por atributos descritos na Tabela 3. É importante mencionar que subconjuntos deste conjunto de atributos serão usados para representar diferentes estágios de jogo.

Tabela 3 – Conjuntos de atributos da Etapa 2

Atributo	**Descrição do atributo**
Atributo 1 (*Dead enemy unit*)	Fornece o número de mortes no exército adversário.
Atributo 2 (*Low HP Agent Unit 1*)	Fornece o número de unidades tipo 1, pertencentes ao exército do agente, com *Hit points* (HP) baixo (HP é uma indicação do quanto de "vida" tem a unidade).
Atributo 3 (*Low HP Agent Unit 2*)	Fornece o número de unidades tipo 2 com HP baixo pertencentes ao exército do agente.
Atributo 4 (*Low HP Agent Unit 3*)	Fornece o número de unidades tipo 3 com HP baixo, pertencentes ao exército do agente.
Atributo 5 (*High HP Agent Unit 1*)	Fornece o número de unidades tipo 1 com HP alto pertencentes ao exército do agente.
Atributo 6 (*High HP Agent Unit 2*)	Fornece o número de unidades tipo 2 com HP alto pertencentes ao exército do agente.fornece o número de unidades tipo 2 com HP alto pertencentes ao exército do agente.
Atributo 7 (*High HP Agent Unit 3*)	Fornece o número de unidades tipo 3 com HP alto pertencentes ao exército do agente.
Atributo 8 (*Attack power*)	Representa o dano que o exército adversário pode causar.
Atributo 9 (*Defense power*)	Representa o poder de defesa do exército adversário.
Atributo 10 (*Damage in construction*)	Fornece uma pontuação referente ao dano que o adversário causou nas estruturas pertencentes ao agente.
Atributo 11 (*Low HP Enemy Unit 1*)	Fornece o número de unidades tipo 1 com HP baixo pertencentes ao exército do adversário.
Atributo 12 (*Low HP Enemy Unit 2*)	Fornece o número de unidades tipo 2 com HP baixo pertencentes ao exército do adversário.
Atributo 13 (*Low HP Enemy Unit 3*)	Fornece o número de unidades tipo 3 com HP baixo pertencentes ao exército do adversário.
Atributo 14 (*High HP Enemy Unit 1*)	Fornece o número de unidades tipo 1 com HP alto pertencentes ao exército do adversário.
Atributo 15 (*High HP Enemy Unit 2*)	Fornece o número de unidades tipo 2 com HP alto pertencentes ao exército do adversário.
Atributo 16 (*High HP Enemy Unit 3*)	Fornece o número de unidades tipo 3 com HP alto pertencentes ao exército do adversário.

O subconjunto de atributos selecionados para representar o estágio EFB, denominado V_1, é

composto pelos seguintes atributos: 5, 6, 7, 8, 9, 14, 15 e 16. O estágio EIB é representado pelo conjunto completo dos dezesseis atributos, denominado V. Por fim, o estágio ECB é representado pelo seguinte subconjunto de atributos: 2, 3, 4, 5, 6, 7, 11, 12, 13, 14, 15 e 16, denominado V_2. Todos os atributos e a formação de subconjuntos foram formados empiricamente, considerando o que é relevante para cada momento, por exemplo, unidades com HP baixo não irão existir no início de confrontos, mas são importantes, por exemplo, na etapa ECB do jogo, onde muitas unidades com HP baixo seria decisivo para a conclusão da partica.

Os detalhes dos experimentos realizados com esta abordagem estão descritos na Seção 5.2.

5.2 Experimentos da etapa 2

Os experimentos desenvolvidos para a etapa 2 buscam mostrar que o uso de conjuntos distintos de atributos, para representar diferentes estágios do jogo StarCraft, pode aprimorar a capacidade de detecção de mudanças de comportamento do algoritmo M-DBScan.

Diferentes bases de dados foram criadas para representar diversos cenários de jogo, incluindo bases em que o intervalo entre as mudanças, referentes ao comportamento, não são constantes.

As bases de dados foram coletadas de partidas reais do jogo StarCraft. Cada amostra de dado retrata o cenário de jogo, sumarizando intervalos de 15 *frames*, por meio dos dezesseis atributos apresentados neste capítulo, ou por um subconjunto desses atributos, dependendo do momento do jogo.

As bases de dados usadas na etapa 2 estão descritas na Tabela 4, onde: a coluna *ID Mudança* é um número sequencial para identificar cada mudança de comportamento na base de dados; *Timestamp* indica quantas amostras de dados chegaram desde o início do fluxo de dados até a ocorrência da mudança de comportamento; *Momento do jogo* indica em que momento do jogo ocorreu a mudança de comportamento (EFB, EIB, ou ECB); *Estratégia* indica um código que referência a estratégia usada na mudança de comportamento. As estratégias adotas são construídas com base na combinação das seguintes ações: *aumento no poder de ataque*; *aumento no poder de defesa*; *aprimoramento do exército com a construção de unidades mais fortes*; *resistência defensiva para o momento de fim de partida*.

Os parâmetros apresentados a seguir são do M-DBScan e foram usados nos experimentos da etapa 2 e selecionados com base em testes empíricos (os parâmetros foram descritos na Seção 2.5):

- Os parâmetros usados na etapa de agrupamento do M-DBScan são: $\mu = 9$; $\beta = 0.1$; $\varepsilon = 20$; $\lambda = 0{,}03$.

- Os parâmetros usados nos experimentos com a entropia temporal são: quantidade mínima de novidades = 2; tamanho da janela deslizante = 20; $\eta_t = 0{,}05$; $\gamma = 0{,}05$; $\delta = 0{,}03$; $\theta = 3$.

Tabela 4 – Descrição das bases de dados da etapa 2

Base de dados	ID Mudança	Timestamp	Momento do jogo	Estratégia
Base de dados 1	1	500	EFB	D1S1
	2	1000	EIB	D1S2
	3	1500	ECB	D1S3
Base de dados 2	1	500	EFB	D2S1
	2	1000	EIB	D2S2
	3	1500	EIB	D2S3
	4	2000	EIB	D2S4
	5	2500	EIB	D2S5
	6	3000	EIB	D2S6
	7	3500	ECB	D2S7
Base de dados 3	1	149	EFB	D3S1
	2	468	EIB	D3S2
	3	528	EIB	D3S3
	4	626	EIB	D3S4
	5	1019	EIB	D3S5
	6	1196	EIB	D3S6
	7	1901	EIB	D3S7
	8	3413	ECB	D3S8
Base de dados 4	1	206	EFB	D4S1
	2	676	EIB	D4S2
	3	1546	EIB	D4S3
	4	1774	EIB	D4S4
	5	2734	EIB	D4S5
	6	5605	EIB	D4S6
	7	6067	EIB	D4S7
	8	7648	EIB	D4S8
	9	7772	ECB	D4S9

- ❑ Os parâmetros usados nos experimentos com a entropia espacial são: quantidade mínima de novidades = 2; tamanho da janela deslizante = 20; $\eta_s = 0{,}05$; $\gamma = 0{,}09$; $\delta = 0{,}002$; $\theta = 3$.

As medidas FP, FN, VP e *F1-Score* são usadas para avaliar os experimentos com cada medida de entropia.

5.2.1 Análise dos experimentos da etapa 2

A Tabela 5 apresenta os resultados dos experimentos da etapa 2, destacando os ganhos obtidos pela presente abordagem, que é baseada em três conjuntos de atributos e três CMs (uma para cada estágio do jogo), sendo que a abordagem tradicional é baseada em um único conjunto de atributos e uma única CM.

É importante destacar que todos os falsos positivos nos experimentos da etapa 2 ocorreram no momento EIB, que é um momento do jogo onde pode ocorrer uma variedade muito grande

de eventos e todos eles com impacto sobre a dinâmica da CM, mas nem sempre sendo suficiente para caracterizar uma mudança que tenha ocorrido no comportamento do adversário.

Tabela 5 – Resultados dos experimentos da etapa 2

Quantidade CM	**Base de dados**	**Entropia Temporal**				**Entropia Espacial**			
		VP	FP	FN	F1	VP	FP	FN	F1
1 CM	**Base de dados 1 (3 mudanças)**	2	1	1	0.666	3	1	0	0.857
	Base de dados 2 (7 mudanças)	3	3	4	0.461	4	1	3	0.666
	Base de dados 3 (8 mudanças)	3	4	5	0.4	3	2	5	0.461
	Base de dados 4 (9 mudanças)	4	4	5	0.47	5	2	4	0.625
3 CMs	**Base de dados 1 (3 mudanças)**	3	1	0	0.857	3	0	0	1
	Base de dados 2 (7 mudanças)	4	3	3	0.571	5	1	2	0.769
	Base de dados 3 (8 mudanças)	4	2	4	0.571	5	2	3	0.666
	Base de dados 4 (9 mudanças)	6	4	3	0.631	6	1	3	0.75

Considerando os dados da Tabela 5, observa-se que a entropia espacial obteve melhores resultados que a temporal. A Figura 14 apresenta uma comparação entre a ocorrência real de mudanças e o momento correspondente de detecção do algoritmo M-DBScan com a entropia espacial. Em tal figura, estão indicados os momentos exatos em que ocorre uma mudança no comportamento do adversário, e o momento em que cada abordagem do M-DBScan (1 CM ou 3 CMs) identificou uma mudança ou indicou um falso positivo. Nos gráficos apresentados na Figura 14, as linhas verticais em destaque indicam as separações entre os estágios EFB, EIB e ECB, os quais sempre ocorrem nessa ordem. É possível observar na Figura 14 que os falsos positivos ocorreram nas seguintes situações: quando a detecção apresentou um atraso considerável; ou quando não existe nenhuma mudança de comportamento, e mesmo assim, uma mudança foi indicada.

Analisando a Figura 14, todas as três mudanças de comportamento na Base de dados 1 foram detectadas pela abordagem com 3 CMs, ao passo que a última delas não foi detectada pela abordagem com 1 CM. Considerando a Base de dados 2, entre as sete mudanças previstas, quatro foram detectadas pela abordagem com 3 CMs, enquanto que somente três foram detectadas pela abordagem com 1 CM. Assim como na Base de dados 1, na Base 2 a mudança no momento ECB foi detectada pela abordagem com 3 CMs, mas isso não ocorreu com a abordagem com 1 CM. Em relação à Base de dados 3, entre as oito mudanças de comportamento esperadas, quatro foram detectadas pela abordagem com 3 CMs, enquanto três mudanças foram detectadas pela abordagem com 1 CM. Assim como ocorreu com as Bases de dados 1 e 2, na Base 3 houve diferença nos resultados entre as abordagens com 1 CM e com 3 CMs, sendo que esta diferença está na detecção ou não de mudanças no momento ECB. Considerando a Base de dados 4, na qual são esperadas nove mudanças de comportamento, seis delas foram detectadas pela abordagem com 3 CMs, enquanto que somente quatro foram detectadas pela abordagem com 1 CM. Nesta última base de dados, as diferenças de detecção entre as abordagens com 1 CM e com 3 CMs ocorreram nos momentos EIB e ECB.

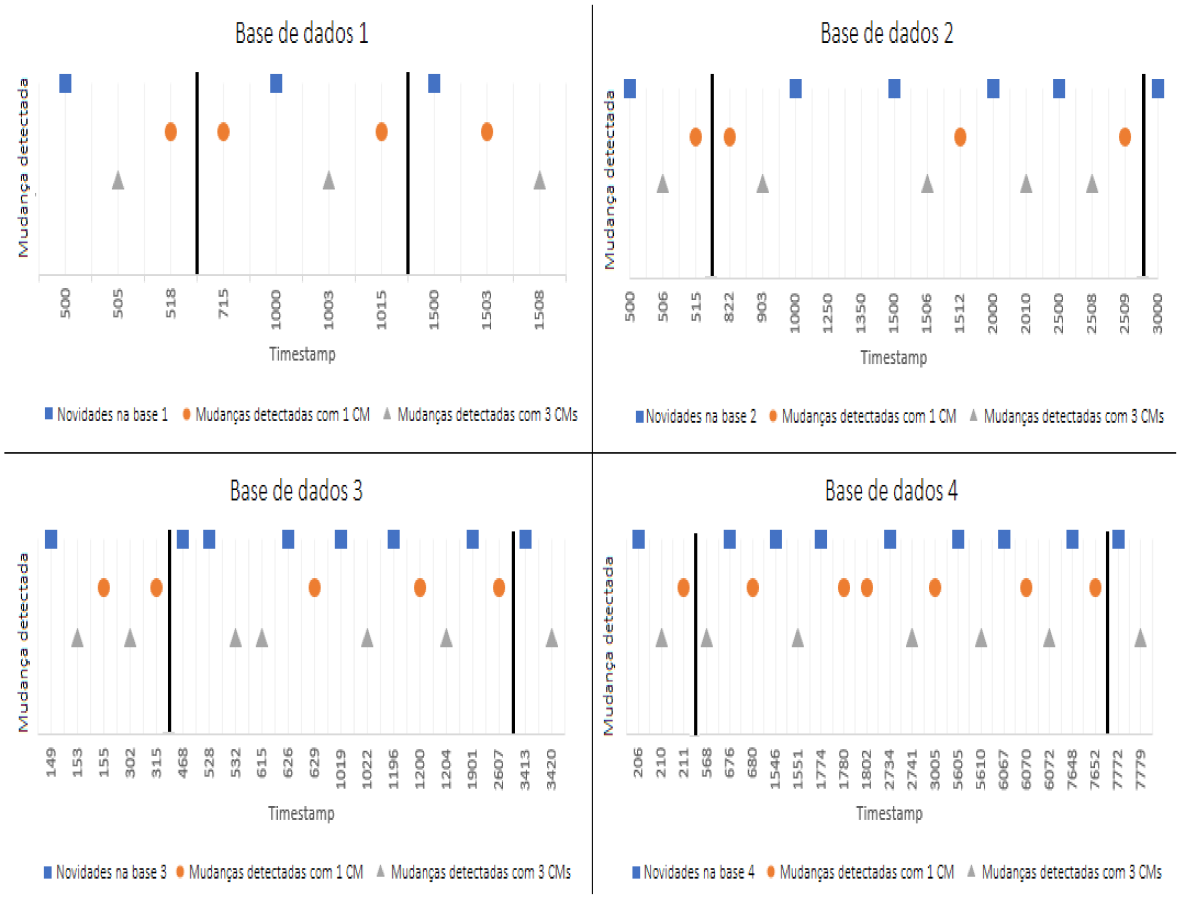


Figura 14 – Gráficos indicando o momento de detecção de mudança de comportamento com a entropia espacial.

Pelo observado nos experimentos da etapa 2, o uso de 1 CM não apresenta bons resultados na detecção de mudanças em momentos mais avançados do jogo. Supõe-se que isso ocorre porque na abordagem com 1 CM todo o histórico da chegada dos dados está concentrado nessa CM. Se somente um tipo de dado D_1 (dados gerados pela mesma distribuição) é obtido do fluxo e isto ocorre por um período longo, para que um outro tipo de dado D_2 caracterize uma mudança de comportamento, deverá chegar uma significativa quantidade de dados D_2, para mudar o que já tinha sido estabelecido devido a constante chegada de dados D_1. Enquanto que com a abordagem de três CMs, o histórico de chegada de dados será dividido entre as CMs de cada momento, logo, a chegada constante de um único tipo de dado pode ter efeito distribuído sobre as entropias de cada CM e não se concentrando em apenas uma, tornando mais fácil a identificação de um novo comportamento, pois ficará mais fácil a entropia superar seu limiar.

O teste de Wilcoxon foi aplicado para comparar a performance do M-DBScan utilizando entropia temporal e espacial, nas abordagens com 1 CM e com 3 CMs. A ferramenta utilizada para aplicação do teste de Wilcoxon foi a KELL (ALCALÁ-FDEZ et al., 2011).

O primeiro teste de Wilcoxon com o M-DBScan utilizou a seguinte hipótese nula: *O M-DBScan com entropia espacial e 3 CMs apresentou o mesmo desempenho que o M-DBScan com entropia espacial e 1 CM*. Como o M-DBScan com entropia espacial e 3 CMs apresentou

uma média de *F1-score* igual a 0,796 contra 0,653 da versão com 1 CM, isso permite formular a seguinte hipótese alternativa: *O M-DBScan com entropia espacial e 3 CMs apresentou um melhor desempenho que o M-DBScan com entropia espacial e 1 CM*. O teste de Wilcoxon apresentou R^+ igual a 10 e R^- igual a 0, com assintótico p-valor igual a 0,04461, permitindo rejeitar a hipótese nula e afirmar a hipótese alternativa.

O segundo teste de Wilcoxon aplicado nesta etapa 2 da pesquisa utilizou a seguinte hipótese nula: *O M-DBScan com entropia temporal e 3 CMs apresentou o mesmo desempenho que o M-DBScan com entropia temporal e 1 CM*. Como o M-DBScan com entropia temporal e 3 CMs apresentou uma média de F1-score igual a 0,658 contra 0,5 da versão com 1 CM, isso permite formular a seguinte hipótese alternativa: *O M-DBScan com entropia temporal e 3 CMs apresentou um melhor desempenho que o M-DBScan com entropia temporal e 1 CM*. O teste de Wilcoxon apresentou R^+ igual a 10 e R^- igual a 0, com assintótico p-valor igual a 0,04461, permitindo rejeitar a hipótese nula e afirmar a hipótese alternativa.

Logo, com ambos os testes estatísticos, foi comprovado que o uso dos conjuntos apropriados de atributos para cada momento do jogo, melhorou o desempenho do algoritmo M-DBScan, independente da entropia utilizada.

5.3 Considerações finais

Os experimentos realizados na etapa 2 da pesquisa indicaram sucesso na tentativa de aumentar a precisão na detecção de mudança de comportamento pelo M-DBScan, com o uso de atributos mais relevantes para cada estágio do jogo.

Com os experimentos deste capítulo, também foi possível validar a hipótese definida na Seção 1.3, que diz o seguinte: "À medida em que uma partida de um jogo RTS evolui, alguns atributos podem se tornar mais relevantes, ou até mesmo perder a sua relevância. Assim sendo, o uso de diferentes conjuntos de atributos para representar fases distintas de evolução do ambiente, proporcionará um aumento na precisão da identificação de mudanças de comportamento pelo algoritmo M-DBScan, em virtude de uma representação mais adequada.".

O sucesso nos experimentos foi alcançado tanto utilizando entropia espacial quanto a temporal, apesar da espacial ter apresentado resultados um pouco melhores com mais ocorrências de VPs e menos ocorrências de FPs, por exemplo.

No capítulo a seguir está descrita a etapa de desenvolvimento do módulo semântico do M-DBScan, que permitirá obter o significado de uma mudança, além de apresentar os experimentos realizados.

CAPÍTULO 6

Etapa 3: Desenvolvimento do módulo semântico do M-DBScan

O desenvolvimento do módulo semântico do M-DBScan consiste em modificações no M-DBScan cujo propósito é atribuir uma semântica às mudanças de comportamento detectadas pelo algoritmo. Tal informação é relevante para nortear o processo de tomada de decisão de um agente, possibilitando a execução de ações contextualizadas à mudança identificada, o que seria de grande relevância em um contexto competitivo como o observado em jogos.

O desenvolvimento do módulo semântico resultou em duas abordagens:

1. Na primeira abordagem foi atribuído como semântica de comportamento o resultado da classificação da amostra de dado que desencadeou a mudança. Essa abordagem foi chamada de *Semantic-MDBScan*, que será detalhadamente descrita na Seção 6.2.

2. Na segunda abordagem, para definir a semântica de comportamento, as amostras de dados que geram novidades são classificadas, e quando se atinge o mínimo necessário de novidades para indicar uma mudança de comportamento, será identificada a semântica com maior relevância estatística entre tais novidades, e essa semântica é definida como a semântica da mudança. Essa abordagem foi chamada de *Statistically Defined Semantic-MDBScan* (SDS-MDBScan), que será detalhadamente descrita na Seção 6.3.

A nomenclatura da semântica gerada por ambas as abordagens (Semantic-MDBScan ou SDS-MDBScan) permite ao usuário entender a reação que o agente terá quando cada comportamento for identificado. Logo, a semântica funciona como um modo de aplicar a IA explicável (ARRIETA et al., 2020), visto que a atribuição de uma semântica às mudanças de comportamento funciona como uma interface que auxilia na compreensão dos usuários sobre as tomadas de decisão que serão adotadas, considerando o que cada mudança de comportamento representa.

Os experimentos nesta etapa 3 visam avaliar as abordagens desenvolvidas, quanto a sua capacidade de detectar mudanças de comportamento e atribuir uma correta semântica a tais mudanças. Diferentes bases de dados são utilizadas nos experimentos, retratando cenários com mudanças abruptas e graduais. Os experimentos são apresentados na Seção 6.4.

Na próxima seção é feita uma formalização do problema que envolve atribuir uma semântica a uma mudança de comportamento detectada no cenário de fluxo de dados.

6.1 Definição do problema de atribuição de semântica a uma mudança

Considere um fluxo de dados D, com tamanho ilimitado S, composto pelas amostras $\{d_1, d_2, ..., d_S\}$, sendo que cada amostra é representada por um conjunto apropriado de atributos. Uma mudança de comportamento ocorre sempre que amostras de dados, geradas seguindo um distribuição de distribuição, apresentarem alterações que ultrapassem um limite pré-definido, como por exemplo, a quantidade de novidades identificadas em um intervalo de tempo e registradas em uma janela deslizante, como faz o M-DBScan (Seção 2.5).

Este problema envolve, essencialmente, duas partes: a primeira parte consiste na detecção de mudanças; a segunda parte consiste na atribuição de uma semântica para uma mudança detectada. Assim, para desenvolver a segunda parte, é preciso, primeiramente, definir um conjunto de semânticas que representam os significados dos possíveis comportamentos presentes em uma base de dados. As semânticas serão obtidas a partir dos rótulos gerados na classificação de amostras de dados compostas por atributos que, em conjunto, representam tais comportamentos. Os comportamentos em uma base de dados podem variar com base nos valores de seus atributos. Detalhes sobre o processo de atribuição de uma semântica a uma mudança, serão apresentados nas seções que descrevem as duas abordagens desenvolvidas: Semantic-MDBScan (Seção 6.2) e SDS-MDBSCan (Seção 6.3).

Para ilustrar o processo de mudança de comportamento, considere um jogador de um jogo digital, e este jogador está executando ações que caracterizam um comportamento defensivo, repentinamente, ele começa a executar ações de um comportamento agressivo, de tal modo, que depois de um certo tempo este comportamento agressivo irá prevalecer. Este exemplo, pode ser formalizado da seguinte maneira: as primeiras $n-1$ amostras de dados representam um comportamento defensivo, que predomina nesta sequência. Suponha que no momento n o comportamento começa a mudar para agressivo, iniciando um período de transição que será concluído no momento $n+x$. Isto significa que a partir do momento $n+x+1$, o comportamento predominante do jogador será o agressivo, permanecendo nele até que uma nova mudança de comportamento ocorra. Importante destacar que durante o período de transição, enquanto a mudança ainda não for concretizada, o comportamento defensivo ainda será considerado como o comportamento que prevalece. Este exemplo está ilustrado na Figura 15.

É importante salientar que cada amostra de dados é composta por um conjunto de atributos, cuja variação de valores permitirá a representação de diferentes comportamentos. Logo, em um intervalo de tempo onde há a predominância de um comportamento, é possível obter do fluxo uma amostra de dados que representa um outro comportamento. Contudo, a quantidade de

Figura 15 – Exemplo de mudança de comportamento.

amostras obtidas deste outro comportamento pode não ser grande o suficiente para caracterizar uma mudança e alterar a tendência de comportamento que prevalece na sequência de dados.

Na próxima seção será detalhada a abordagem Semantic-MDBScan, descrevendo todo o procedimento de atribuição de semântica às mudanças de comportamento.

6.2 Semantic-MDBScan

A proposta de criação do Semantic-MDBScan consiste em desenvolver um algoritmo capaz de operar no cenário de fluxo contínuo de dados, detectar eventuais mudanças na distribuição das sequências de dados, e também, ser capaz de atribuir uma semântica a tais mudanças.

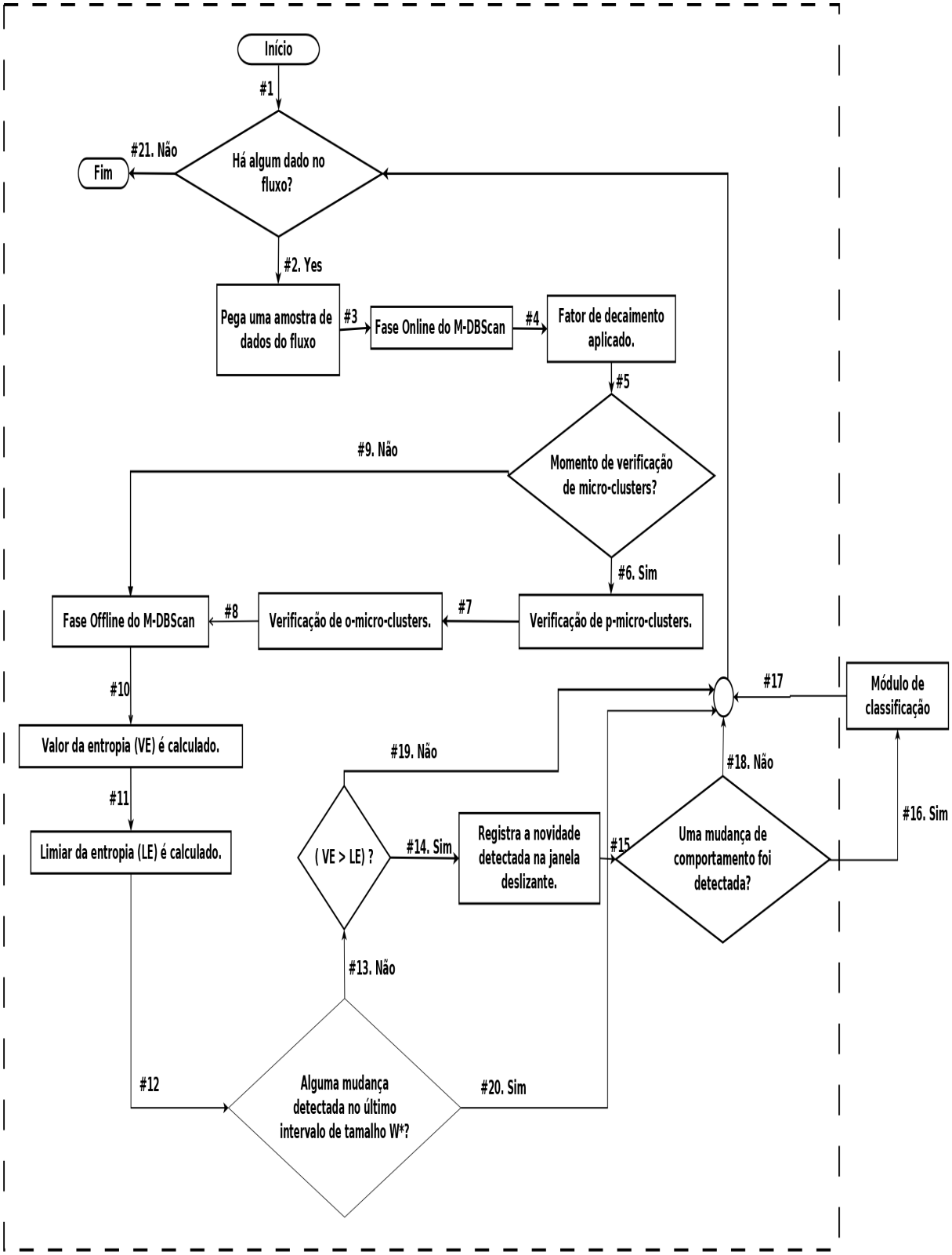
O processo de detecção de mudança de comportamento no Semantic-MDBScan é semelhante ao existente no M-DBScan (descrito na Seção 2.5). A detecção de mudança ocorre sempre que uma quantidade mínima de novidades dentro de um intervalo de tempo, representado por uma janela deslizante, é atingida. A diferença entre o M-DBSCAN e o Semantic-MDBScan é que o último possui um processo de atribuição de semântica. Para isso, o Semantic-MDBScan executa o processo de detecção de mudança de comportamento, e caso uma mudança seja detectada, o algoritmo tenta atribuir um significado a ela. Isto ocorre, por meio da classificação da amostra de dados que provocou a última novidade necessária para declarar que uma mudança de comportamento ocorreu, ou seja, o rótulo obtido neste processo de classificação será a semântica da mudança. Um fluxograma representando o algoritmo Semantic-MDBScan é apresentado na Figura 16, sendo que os procedimentos que existe em comum entre os algoritmos estão delimitados pelo quadrado tracejado.

Primeiramente, o algoritmo verifica a disponibilidade de alguma amostra de dados no fluxo (seta #1): se sim (seta #2), a amostra será processada pela parte *online* do algoritmo (seta #3). Na sequência, um fator de decaimento é aplicado em todos os *micro-clusters*, com exceção daquele ao qual foi atribuída a última amostra de dados obtida do fluxo (seta #4). O fator de decaimento é usado para reduzir certas propriedades que representam tais *micro-clusters*, de tal modo que eles tendem a desaparecer se não atenderem mais os requisitos mínimos de um *micro-cluster*. A seguir, o algoritmo verifica se é o momento para realizar a reclassificação de *micro-clusters*, (seta #5). Se for o momento, o algoritmo converte em *o-micro-cluster* todo

p-micro-cluster que não atende mais os requisitos de um *p-micro-cluster* (CAO et al., 2006) (seta #6). Depois, o algoritmo descarta todos os *o-micro-clusters* que não atendem os requisitos mínimos para continuar existindo (seta #7). Então, quando finalizada essa reclassificação de *micro-clusters* (seta #8) ou caso não seja o momento de fazê-la (seta #9), inicia-se a parte *offline* do algoritmo (Seção 2.5.4.2). Uma vez que a parte *offline* é concluída, o valor de entropia e seu limiar são calculados (setas #10 e #11, respectivamente). Agora, o algoritmo verifica se ocorreu alguma mudança de comportamento no último período de tempo equivalente ao tamanho da janela deslizante (W) (seta #12), pois se ocorreu o processo de identificar novas mudanças é ignorado por um período igual ao tamanho da janela deslizante, ou seja, os eventos nas setas #13, #14, #15 e #16 não são executados. Caso não tenha ocorrido (seta #13), o Semantic-MDBScan verifica se o valor de entropia ultrapassou seu limiar, pois se tiver ultrapassado (seta #14), uma novidade foi detectada e deve ser registrada na janela deslizante, mas caso não haja uma novidade (seta #19), o algoritmo tentará buscar uma nova amostra de dados do fluxo (seta #2 ou #21). Porém, se uma nova novidade foi identificada, é preciso certificar se esta novidade desencadeou ou não uma mudança de comportamento (seta #15). Se sim (seta #16), a amostra de dados que desencadeou a mudança é submetida ao *Módulo de Classificação* e o rótulo produzido é usado como semântica (significado) da mudança de comportamento. Depois, o algoritmo busca por uma nova amostra do fluxo (seta #17), o que também ocorre se uma mudança de comportamento não for detectada (seta #18). Se existir uma nova amostra de dado (seta #2), todo o algoritmo será executado novamente. Caso contrário, a execução do Semantic-MDBScan é finalizada (seta #21).

Com base nas dimensões apresentadas na Seção 2.4, de maneira semelhante ao M-DBScan, o Semantic-MDBScan se encaixa nas seguintes dimensões: considerando a dimensão *Data Management*, o algoritmo se encaixa na abordagem *Full Memory approach*, pois, para monitorar as mudanças na distribuição dos dados, considera-se o impacto na CM e a amostra de dado mais recente tem maior peso no cálculo da entropia. Quanto a dimensão *Detection*, o Semantic-MDBScan usa a entropia como um parâmetro para monitorar as mudanças na distribuição dos dados, se encaixando na abordagem *Performance Parameter-based*. Considerando a dimensão *Adaptation*, o Semantic-MDBScan se encaixa no grupo da abordagem *Blind*, uma vez que o algoritmo atualiza o modelo sempre que uma nova amostra de dados é lida do fluxo.

Com o objetivo de aprimorar o processo de atribuição semântica apresentado nesta seção, almejando um processo mais assertivo, foi desenvolvido o algoritmo SDS-MDBScan. Esse novo algoritmo define a semântica de uma mudança com base na relevância estatística da semântica na sequência de dados onde foi identificada a mudança de comportamento. Detalhes do desenvolvimento do SDS-MDBScan estão apresentados na Seção 6.3.



* *W é o tamanho da janela deslizante*

Figura 16 – Fluxograma do Semantic-MDBScan

6.3 SDS-MDBScan

O SDS-MDBScan tem o objetivo de identificar as mudanças de comportamento no cenário de fluxo contínuo de dados, visando atribuir uma semântica a estas mudanças, que seja relevante estatisticamente. Para isso, serão consideradas todas as novidades que juntas provocaram a detecção da mudança de comportamento, e que serão usadas para obter uma semântica que de fato descreve a mudança.

Para atingir o objetivo apresentado do SDS-MDBScan, o algoritmo executa um processo de detecção de mudança baseado em uma análise das novidades registradas em uma janela deslizante. Para cada mudança de comportamento detectada, o algoritmo busca por sua semântica considerando a mais recente sequência de dados, cujo processamento gerou as novidades registradas na janela deslizante. Reforçando que esta informação semântica será de grande valor para o processo de tomada de decisão de agentes que operam no contexto de um fluxo contínuo de dados, uma vez que esta informação ajudará o agente a selecionar as ações que melhor se encaixam como uma reação ao cenário corrente do problema.

No contexto de agentes jogadores, por exemplo, a analise de amostras de dados que representam o contexto corrente do jogo, deixará o agente ciente de eventuais mudanças no modo de operação do seu adversário. Tal informação pode aprimorar as decisões do agente, pois elas serão contextualizadas ao que é observado.

Detalhes sobre o processo de detecção de novidades com o uso de janela deslizante é apresentado na Seção 6.3.1.

6.3.1 SDS-MDBScan: particularidade no uso da janela deslizante para detectar novidades

De maneira semelhante ao que é feito pela abordagem original do M-DBScan, o SDS-MDBScan realiza a detecção de novidades da seguinte maneira: considere um fluxo contínuo D com tamanho S composto pelas amostras $\{d_1, d_2, ..., d_S\}$. Tendo uma janela deslizante (com tamanho W) $\{d_i, ..., d_{i+W-1}\}$, onde $1 \leq i$ e $i+W-1 \leq S$. Sempre que o SDS-MDBScan detecta uma mudança de comportamento devido ao processamento de uma amostra d_P, o procedimento funcionará da seguinte maneira: a partir deste ponto, toda novidade que eventualmente ocorrer será ignorada, ou seja, ela não será usada para identificar a próxima mudança de comportamento. Contudo, as amostras de dados continuarão a ser usadas para atualizar a CM. Neste caso, o SDS-MDBScan só voltará a considerar, novamente, as novidades, após um período equivalente ao tamanho da janela deslizante, que seria a partir da amostra d_{P+W}. Isso significa que o SDS-MDBScan opera de forma a detectar uma mudança de comportamento por janela deslizante. Caso tenha ocorrido uma mudança, o processo de detecção de novidades não será executado por um período igual a W, que é suficiente para transladar toda a janela deslizante, e consequentemente, apagar qualquer dado remanescente da última mudança de comportamento.

Detalhes sobre o processo de detecção de mudanças e de atribuição de uma semântica a uma mudança são apresentados na Seção 6.3.2.

6.3.2 SDS-MDBScan: Atribuindo uma semântica para mudanças detectadas

Esta seção apresenta como o SDS-MDBScan indica a ocorrência de mudanças em um fluxo contínuo e atribui semânticas a elas. Em tal procedimento, todo *micro-cluster* tem associado a ele um rótulo que representa a semântica das amostras de dados que ele recebe ao longo do fluxo. Para isso, foi necessário a incorporação de um campo na estrutura de cada *micro-cluster* para armazenar este rótulo. Nos experimentos descritos na Seção 6.4.5, os algoritmos KNN (DUDANI, 1976) e o *Multilayer Perceptron* (MLP) (HAYKIN, 2010) foram testados na tarefa de classificação.

Os algoritmos de classificação usados neste trabalho contam com uma base de dados rotulada chamada *Training Dataset*, que é usada de maneira específica por cada algoritmo de classificação: no caso do KNN, ele tenta encontra as k amostras de dados da *Training Dataset* que são mais similares à amostra de dados a ser classificada; no caso da rede MLP, a *Training Dataset* é usada no treinamento da rede neural.

A escolha destes algoritmos de classificação foi baseada nas suas características, como: a rede MLP produz um modelo, que apesar da etapa de treinamento necessária, ele é tradicionalmente mais rápido que o KNN, especialmente se a *Training Dataset* for muito grande (AL-NABI; AHMED, 2013). Contudo, mudanças na distribuição dos dados obtidos do fluxo podem comprometer o desempenho da rede MLP, exigindo um novo treinamento para que a rede se adapte a tais mudanças. Por outro lado, o KNN é um algoritmo incremental que pode ser facilmente adaptado às mudanças na distribuição dos dados, bastando atualizar a *Training Dataset*, sem necessidade de treinamento.

O critério de detecção de mudança de comportamento usado pelo SDS-MDBScan é mais restritivo que o usado originalmente pelo M-DBScan (veja Seção 2.5.5) e pelo Semantic-MDBScan (veja Seção 6.2). De fato, o SDS-MDBScan somente indica uma mudança de comportamento quando o número de novidades, com a mesma semântica, registradas na janela deslizante ultrapassar uma quantidade mínima definida previamente. O motivo para utilizar um critério mais restritivo é para garantir que uma mudança só será indicada pelo algoritmo, devido a frequência, estatisticamente relevante, de novidades associadas a uma mesma semântica, representando a tendência desta sequência de dados.

Para executar tal procedimento de detecção de mudanças, o SDS-MDBScan precisa guardar a semântica de cada novidade registrada na janela deslizante. Neste caso, o rótulo (semântica) de cada novidade corresponde àquele presente na estrutura do *micro-cluster*, ao qual, a amostra de dado, cujo processamento desencadeou em uma novidade, foi atribuída.

Para ilustrar o processo de detecção de mudança, considere uma janela deslizante de ta-

manho 5 e o mínimo de 2 novidades com mesma semântica para indicar uma mudança de comportamento. Na Figura 17, observa-se a detecção de uma terceira novidade registrada na janela deslizante devido ao processamento da amostra de dados D, onde D é a segunda novidade na janela com semântica igual a X. Neste caso, o processamento da amostra D provocou a detecção uma mudança de comportamento cuja semântica é X.

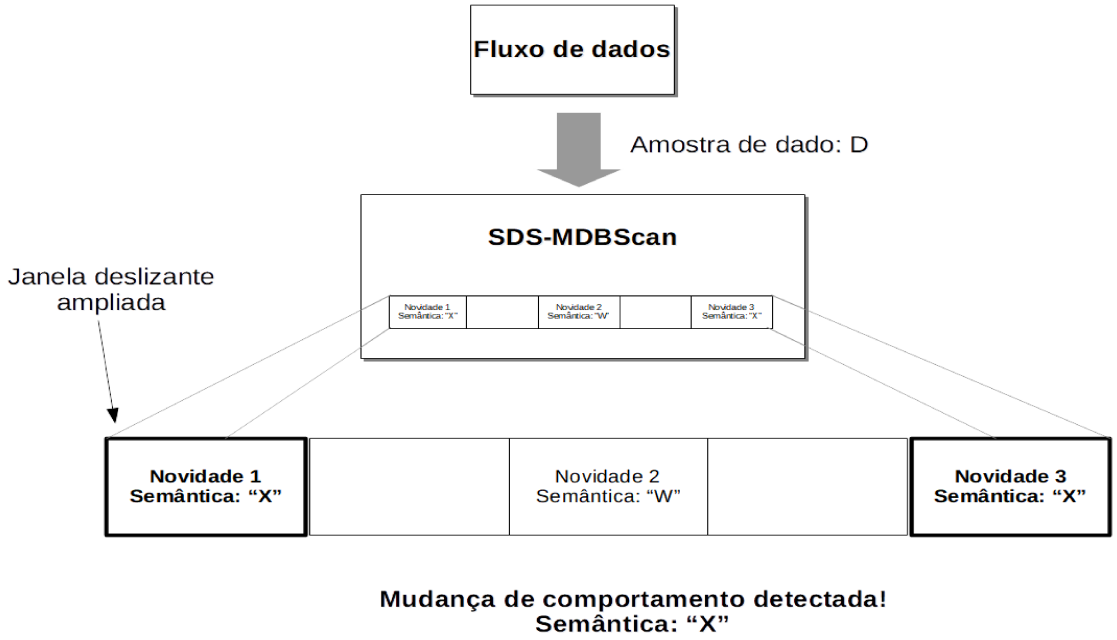


Figura 17 – SDS-MDBScan: Um exemplo de atribuição semântica a uma mudança.

A Figura 18 apresenta um fluxograma representando o processamento do SDS-MDBScan. O quadrado com linhas tracejadas em verde delimita a parte do SDS-MDBScan que é igual a versão original do M-DBScan, e o quadrado com linhas tracejadas em azul delimita os módulos introduzidos pelo SDS-MDBScan. Inicialmente, o algoritmo verifica se há uma amostra de dados s a ser lida do fluxo de dados (seta #1): se sim, o algoritmo lê a amostra do fluxo (seta #2) e a apresenta como entrada para a parte *online* do algoritmo (seta #3). Se a amostra s é alocada em um *micro-cluster* (podendo ser um *p-micro-cluster* ou *o-micro-cluster*) já existente o, então já existe uma semântica atribuída a ele. Caso contrário, um novo *micro-cluster* o_n é criado, e o *Módulo de atribuição de semântica* é ativado para classificar s, e o resultado da classificação é usado como semântica do o_n (seta #4). Na sequência, retorna-se para a parte *online* do algoritmo (seta #5). Uma vez concluída a parte *online*, um fator de decaimento é aplicado a todo *micro-cluster* com exceção daquele que recebeu a amostra s (seta #6). Então, o algoritmo testa (seta #7) se é o momento para uma verificação de *micro-clusters*: se sim, cada *p-micro-cluster* que não atender mais os requisitos de um *p-micro-cluster* (devido ao uso do fator de decaimento) se tornará um *o-micro-cluster* (CAO et al., 2006) (seta #8). Na sequência, todos os *o-micro-clusters* que não atenderem os requisitos mínimos de um *micro-cluster* (também devido ao uso do fator de decaimento) serão apagados (seta #9). Então, a parte *offline* do algoritmo será

executada (seta #10). Caso não seja o momento de realizar a verificação de *micro-clusters* (seta #7), a parte *offline* é iniciada diretamente (seta #11). Uma vez concluída a parte *offline*, o valor de entropia e seu limiar são calculados (setas #12 e #13, respectivamente). Agora, verifica-se se ocorreu alguma mudança de comportamento no último período de tempo equivalente ao tamanho W da janela deslizante (seta #14), pois se ocorreu, o processo de identificar novas mudanças é ignorado por um período igual ao tamanho da janela deslizante. Caso não tenha ocorrido (seta #15), o SDS-MDBScan verifica se o valor de entropia ultrapassou seu limiar. Se sim (seta #16), então uma novidade deve ser registrada na janela deslizante com a sua semântica, que é a mesma presente no *micro-cluster* ao qual a amostra s foi atribuída. Na sequência, o algoritmo ativa o módulo que verifica se esta nova novidade provocou uma mudança de comportamento de acordo com o critério do SDS-MDBScan, (seta #17). Depois, tendo detectada uma mudança ou não, o algoritmo tenta ler mais um dado do fluxo (seta #18). Se existir mais um dado (seta 2), todo o processo será repetido a partir da parte *online* do algoritmo (seta #3). Caso contrário, o processamento do SDS-MDBScan é finalizado (seta #21).

Considerando uma outra alternativa para o teste na seta #15, onde o valor de entropia não ultrapassou seu limiar, o algoritmo verifica se há uma nova amostra de dados para ler do fluxo (seta #19). Se tiver (seta 2), todo o processo será repetido a partir da parte *online* do algoritmo (seta #3). Caso contrário, o processamento do SDS-MDBScan é finalizado (seta #21).

Considerando uma outra alternativa para o teste na seta #14, se tiver sido detectada uma mudança de comportamento no último intervalo de tamanho igual a janela deslizante (seta #20), o algoritmo verifica se há uma nova amostra a ser lida do fluxo. Se tiver (seta #2), todo o processo é repetido. Contudo, os procedimentos nas setas #15, #16 e #17 serão ignorados por um período de tempo equivalente ao tamanho da janela deslizante, para evitar a detecção de novidades relacionadas à mudança já identificada. Se não tiver outra amostra a ser lida do fluxo, então a execução do SDS-MDBScan será finalizada (seta #21).

Considerando as dimensões apresentadas na Seção 2.4, de maneira semelhante ao M-DBScan, o SDS-MDBScan se encaixa nas seguintes dimensões: considerando a dimensão *Data Management*, o algoritmo se encaixa na abordagem *Full Memory approach*, pois, para monitorar as mudanças na distribuição dos dados, considera-se o impacto na CM e a amostra de dado mais recente tem maior peso no calculo da entropia. Considerando a dimensão *Detection*, o SDS-MDBScan usa a entropia como uma parâmetro para monitorar as mudanças na distribuição dos dados. Então, o algoritmo se encaixa na abordagem *Performance Parameter-based*. Considerando a dimensão *Adaptation*, o SDS-MDBScan se encaixa no grupo da abordagem *Blind*, uma vez que o algoritmo atualiza o modelo sempre que uma nova amostra de dados é lida do fluxo.

Na Seção 6.4 estão descritos todos os experimentos relacionados a este capítulo que aborda a criação do Semantic-MDBScan e do SDS-MDBScan.

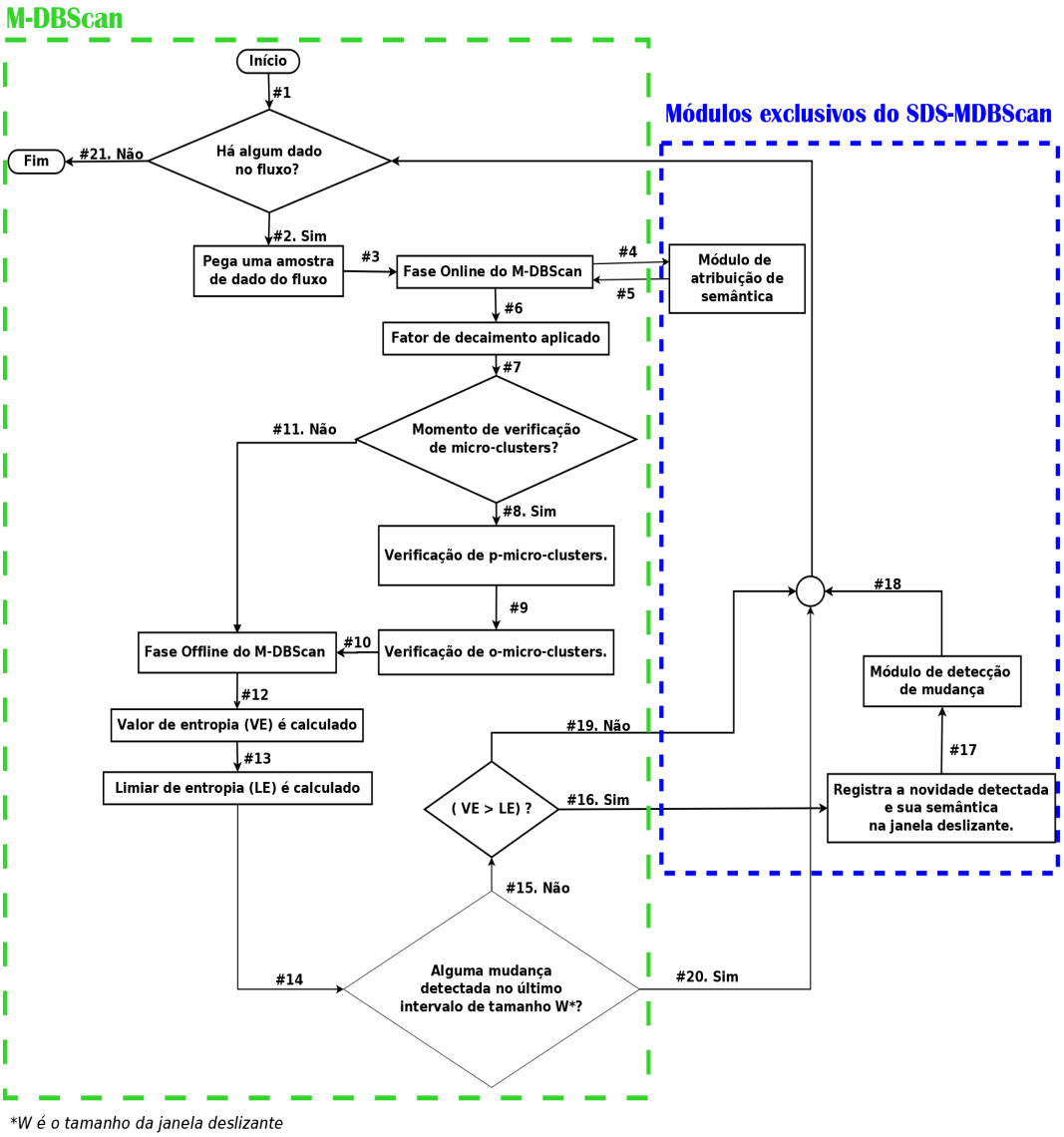


Figura 18 – Fluxograma do SDS-MDBScan.

6.4 Experimentos da etapa 3

Esta seção apresenta os resultados obtidos com o desenvolvimento do módulo semântico que gerou duas abordagens: A primeira consiste no desenvolvimento do Semantic-MDBScan (Seção 6.2), cujos experimentos estão descritos na Seção 6.4.4; A segunda consiste no desenvolvimento do SDS-MDBScan (Seção 6.3), cujos experimentos estão descritos na Seção 6.4.5.

Nos experimentos foram testados os algoritmos classificadores KNN e MLP. A escolha de tais algoritmos de classificação se deve a algumas especificidades que cada um apresenta, tal como explicado a seguir. O KNN é um algoritmo incremental que não demanda treinamento, o que o torna bastante interessante para ser usado em um cenário de fluxo contínuo de dados. Por outro lado, a MLP, apesar de demandar treinamento, uma vez treinado, é mais ágil do que o KNN para efetuar classificações. Contudo, em um problema que envolve fluxo contínuo de dados, torna-se interessante averiguar se as eventuais alterações da distribuição de dados não

comprometem a acurácia da classificação efetuada pela MLP (o que requereria re-treinamentos periódicos da MLP).

Antes de apresentar os resultados dos experimentos, na Seção 6.4.1 são apresentadas as bases artificiais usadas tanto nos experimentos do Semantic-MDBScan quanto do SDS-MDBScan, posteriormente, na Seção 6.4.5.1, serão apresentadas as bases reais usadas no experimentos com o SDS-MDBScan. Na Seção 6.4.2 são apresentadas as medidas de avaliação usadas na análise dos resultados e na Seção 6.4.3 são apresentadas as configurações dos experimentos.

6.4.1 Base de dados artificiais dos experimentos com Semantic-MDBScan e SDS-MDBScan

Em todas as bases artificiais criadas para os experimentos com Semantic-MDBScan e com SDS-MDBScan, há mudanças ocorrendo em intervalos fixos de tempo representados por m amostras de dados, sendo que temos $m = 1000$ para as bases de dados 2, 3, 4 e 6. Enquanto que para as bases 1, 5 e 7 temos $m = 20000$. Cada amostra nestas bases de dados é representada por 10 atributos.

Para gerar diferentes cenários de teste com dados produzido a partir de diferentes distribuições, nas primeiras 5 bases artificiais as mudanças são do tipo abrupta/recorrente, enquanto que as bases artificiais 6 e 7 possuem mudanças do tipo gradual/recorrente, que representam cenários mais desafiadores.

Com o propósito de auxiliar os classificadores (KNN e rede MLP), sete bases rotuladas, chamadas de *Training Dataset*, foram criadas, uma para cada base de dados dos experimentos.

Todas as *Training Dataset* e bases dos experimentos foram criadas usando amostras de dados geradas pelo *framework Massive Online Analysis* (MOA) (BIFET et al., 2010), aplicando o gerador *random radial basis function generator*. Lembrando, que as primeiras cinco bases de dados apresentam mudanças do tipo abrupta/recorrente, sendo que a sexta e sétima bases possuem mudanças do tipo gradual/recorrente.

Em toda *Training Dataset*, cada comportamento é representado por uma quantidade de amostras que equivale a 15% do valor de m da base usada no experimento, ao qual a *Training Dataset* está vinculada. Este valor foi empiricamente definido para assegurar que o tempo de execução do KNN não comprometa a eficiência do processamento, garantindo que a identificação das mudanças de comportamento ocorra enquanto as mudanças ainda são vigentes.

Originalmente, as bases 6 e 7 também são geradas por amostras de dados do MOA, usando o gerador *random radial basis function generator*. Contudo, para garantir as mudanças graduais foi aplicada a seguinte modificação: algumas amostras nas transições das mudanças são substituídas por amostras geradas por diferentes distribuições de dados para representar a etapa de transição.

Mais especificamente, nas bases 6 e 7, o processo gradual de mudança nas transições entre duas sequências com semânticas distintas, dura por uma quantidade de $0,1 * m$ amostras. En-

tão, neste período há uma oscilação de semântica, mas depois de passado este momento uma semântica irá prevalecer.

Considerando o processo de criação das bases artificiais, uma diferença importante está no parâmetro *speed*, que define o movimento de *kernels* ao longo da chegada de dados do fluxo: para a base 1 este parâmetro recebeu valor 500, e em todas as outras bases foi usado o valor 100.

As bases de dados artificiais usados nos experimentos do Semantic-MDBScan e do SDS-MDBScan estão descritas na Tabela 6: a coluna *ChangeID* enumera as mudanças de comportamento em cada base; o *Timestamp* indica o momento da mudança, apresentando a quantidade de amostras lidas do fluxo até esse momento da mudança; e a coluna *Semântica* indica a semântica que prevalece na sequência de dados a partir da mudança. Como são bases artificiais, a semântica é indicada por rótulos genéricos.

6.4.2 Medidas de avaliação

Nesta etapa 3 não basta avaliar as técnicas somente em função das detecções, é preciso considerar se a semântica atribuída a uma mudança está correta. Por isso, novas situações foram consideradas nas medidas usadas para a avaliação, que passam a contemplar as seguintes situações:

- *Verdadeiro Positivo (VP)*: indica o número de detecções de mudanças corretas e com a correta atribuição de semântica para a mudança. O atraso permitido para a detecção é até 50% de m;
- *Falso Positivo (FP)*: indica o número de detecções incorretas de mudanças;
- *Falso Negativo (FN)*: esta métrica pode ocorrer nas seguintes situações: 1) O algoritmo detecta uma mudança real, mas isso só ocorre depois do atraso máximo permitido (referenciado como *FN-D*, *FN delayed*); 2) O algoritmo detecta uma mudança real, mas atribui uma semântica errada para a mudança (referenciado como *FN-S*, *FN semantic*); e 3) O algoritmo não detecta a mudança (referenciado como FN-B, *FN blind*);
- *Verdadeiro Negativo (VN)*: indica o número de amostras na base de dados que o algoritmo processou e corretamente avaliou como não sendo uma mudança de comportamento. Então, tal número corresponde a diferença entre o total de amostras de dados na base e a soma do VP, FP e do FN.
- *F1*: A medida F1 expressa a medida harmônica entre a precisão e revocação.

Como as principais contribuições dos algoritmos Semantic-MDBScan e SDS-MDBScan são as suas abordagens na definição da semântica de uma mudança. Então, a medida FN-S assume uma importância maior, pois caso estes algoritmos errem na atribuição da semântica a uma mudança, o FN-S refletirá tais erros.

Tabela 6 – Descrição das bases de dados artificiais

Base de dados	**ChangeID**	**Timestamp**	**Semântica**
Base de dados 1	1	20000	class2
	2	40000	class3
	3	60000	class4
	4	80000	class1
	5	100000	class2
	6	120000	class3
	7	140000	class4
	8	160000	class2
	9	180000	class4
Base de dados 2	1	1000	class2
	2	2000	class1
	3	3000	class2
Base de dados 3	1	1000	class3
	2	2000	class2
	3	3000	class1
	4	4000	class3
Base de dados 4	1	1000	class2
	2	2000	class3
	3	3000	class4
	4	4000	class1
Base de dados 5	1	20000	class2
	2	40000	class3
	3	60000	class4
	4	80000	class1
	5	100000	class2
	6	120000	class3
	7	140000	class4
	8	160000	class2
	9	180000	class4
Base de dados 6	1	1000	class2
	2	2000	class1
	3	3000	class2
Base de dados 7	1	20000	class2
	2	40000	class3
	3	60000	class2

6.4.3 Configurações dos experimentos com o Semantic-MDBScan e com o SDS-MDBScan

Os experimentos com o Semantic-MDBScan e com o SDS-MDBScan ocorreram utilizando parâmetros definidos empiricamente. Os parâmetros da etapa de agrupamento foram os seguintes: $\mu = 10$; $\beta = 0.105$; $\varepsilon = 0.45$; $\lambda = 0.05$, onde μ é um número mínimo de amostras necessárias para ser considerado um *micro-cluster*; ε é o limite de raio para um *micro-cluster*; β é o limiar de *outlier*; e λ é fator de decaimento.

Os parâmetros usados com a entropia espacial foram os seguintes: número mínimo de novidades para uma mudança = 2; tamanho da janela deslizante = 20. Também foram definidos os seguintes valores: $\eta_s = 0.05$; $\gamma = 0.5$; $\delta = 0.02$; $\theta = 2$, onde η_t é um peso usado para controlar a intensidade da atualização das probabilidades; γ e δ são pesos usados no cálculo do limiar da entropia; e θ é o número de desvio padrão utilizado para definir uma distribuição normal para os valores de entropia.

No KNN, o valor de K foi definido por meio de experimentos, onde os valores de 1 a 10 foram avaliados. Os melhores resultados foram obtidos usando $K = 5$, que geraram 89% de acurácia utilizando o KNN nestes experimentos avaliativos.

Com o propósito de reduzir a quantidade de FPs nos experimentos com as bases 6 e 7, que apresentam mudanças graduais, foram aplicadas as seguintes mudanças nos parâmetros: a quantidade mínima de novidades para ocorrência de mudança de comportamento passou a ser 4; e o fator de decaimento (λ) passou a ser 0,001, para reduzir a velocidade do fator de esquecimento dos *micro-clusters*, que consequentemente traz estabilidade para os estados da CM, reduzindo os FPs provenientes do desaparecimento destes estados, impactando os valores de entropia.

Uma rede MLP foi criada para cada base de dados e treinada usando dados da respectiva *Training Dataset*. Como o SDS-MDBScan utilizou bases reais além das artificiais usadas pelo Semantic-MDBScan, isso gerou algumas diferenças nas configurações das redes MLP para cada abordagem.

A arquitetura das redes MLP, nos experimentos com o Semantic-MDBScan, foram definidas empiricamente ao longo do treinamento, tendo: 10 neurônios na camada de entrada; 2 camadas intermediárias, onde a primeira tem seis neurônios e a segunda tem três. Estas quantidades de neurônios na camada de entrada e nas intermediárias também se repetem para as redes MLP utilizadas nos experimentos com o SDS-MDBScan.

Por sua vez, a quantidade de neurônios na camada de saída depende da base de dados para qual a rede MLP está sendo treinada, mas esta quantidade varia entre 2 e 4 neurônios para as bases artificiais que foram usadas tanto para os experimentos com o Semantic-MDBScan, quanto para os experimentos com o SDS-MDBScan. Contudo para este último, também foram usadas bases reais, então, esta quantidade de neurônios na última camada varia entre 2 e 5 neurônios.

A função de ativação utilizada nestas redes MLP foi a tangente hiperbólica, e a taxa de

aprendizagem foi de 0,1, para ambas as abordagens, sendo que o treinamento da rede era concluído quando o erro global fosse inferior a 0,01 para as bases artificiais e 0,03 para as bases reais.

6.4.4 Experimentos com o Semantic-MDBScan

Estes experimentos visam avaliar o Semantic-MDBScan na tarefa de atribuir a semântica correta para as mudanças de comportamento detectadas pelo algoritmo.

Para fins de comparação, dois métodos de classificação forão investigados: o K-Nearest-Neighbors (KNN) (DUDANI, 1976); e o *Multilayer Perceptron* (MLP) (HAYKIN, 2010). Ambas as abordagens são avaliadas utilizando sete bases de dados distintas, apresentadas na Seção 6.4.1.

Convém salientar que não foram realizados experimentos com o Semantic-MDBScan utilizando bases reais pelo seguinte motivo: com os resultados obtidos com as bases artificiais, foi previamente possível identificar a fragilidade de tal abordagem na identificação da semântica das mudanças (conforme descrito na análise dos resultados). Os experimentos posteriores, realizados com o SDS-MDBScan, comprovaram que essa segunda abordagem conseguiu sanar com êxito as fragilidades da primeira, logo, somente o SDS-MDBScan foi testado para as bases reais.

6.4.4.1 Resultados dos experimentos com o Semantic-MDBScan

A Tabela 7 apresenta os resultados dos experimentos que avaliam a habilidade do Semantic-MDBScan com KNN em detectar mudanças e atribuir um significado adequado a elas. A Tabela 8 apresenta os resultados produzidos pelo Semantic-MDBScan com MLP com o mesmo objetivo.

Tabela 7 – Resultados dos experimentos: *Semantic-MDBScan com KNN*

Tipo de mudança	**Base de dados**	**Medidas**				
		VP	**FP**	**FN-B / D / S**	**VN**	**F1**
Abrupta	**Base de dados 1 (9 mudanças)**	6	6	1 / 2 / 0	199985	0,57
	Base de dados 2 (3 mudanças)	2	0	1 / 0 / 0	3997	0,8
	Base de dados 3 (4 mudanças)	3	1	1 / 0 / 0	4995	0,75
	Base de dados 4 (4 mudanças)	3	2	1 / 0 / 0	4994	0,66
	Base de dados 5 (9 mudanças)	7	3	1 / 1 / 0	199988	0,73
	Média do *F1-score* (Abrupta)					**0,702**
Gradual	**Base de dados 6 (3 mudanças)**	1	1	2 / 0 / 0	3996	0,4
	Base de dados 7 (3 mudanças)	1	13	0 / 0 / 2	79984	0,11
	Média do *F1-score* (Gradual)					**0,255**

Antes de analisar os resultados é interessante destacar que os classificadores (KNN ou MLP) não possuem influência sobre o processo de detecção de mudanças, eles somente atuam na atribuição de um significado para as mudanças.

Tabela 8 – Resultados dos experimentos: *Semantic-MDBScan com MLP*

Tipo de mudança	**Base de dados**	**Medidas**				
		VP	**FP**	**FN-B / D / S**	**VN**	**F1**
Abrupta	**Base de dados 1 (9 mudanças)**	6	6	1 / 2 / 0	199985	0,57
	Base de dados 2 (3 mudanças)	2	0	1 / 0 / 0	3997	0,8
	Base de dados 3 (4 mudanças)	3	1	1 / 0 / 0	4995	0,75
	Base de dados 4 (4 mudanças)	3	2	1 / 0 / 0	4994	0,66
	Base de dados 5 (9 mudanças)	7	3	1 / 1 / 0	199988	0,73
	Média do *F1-score* (Abrupta)					**0,702**
Gradual	**Base de dados 6 (3 mudanças)**	1	1	2 / 0 / 0	3996	0,4
	Base de dados 7 (3 mudanças)	2	13	0 / 0 / 1	79984	0,22
	Média do *F1-score* (Gradual)					**0,31**

Considerando as bases 1, 2, 3, 4 e 5, que possuem mudanças abruptas/recorrentes, as Tabelas 7 e 8 mostram que ambas as abordagem do Semantic-MDBScan apresentaram bons resultados, sendo que não ocorreu nenhum erro de atribuição semântica, que é o fator que poderia diferenciar as abordagens usando KNN ou MLP.

Ainda considerando os resultados resultados das bases 1, 2, 3, 4 e 5, o número de VPs indica que mais de 50% das mudanças foram detectadas, o que reflete no valor de F1 que varia de 0,57 até 0,8. A fim de mostrar a ocorrência de VP, FP, VN e FN ao longo do fluxo, as Figuras 19, 20, 21, 22 e 23 foram criadas.

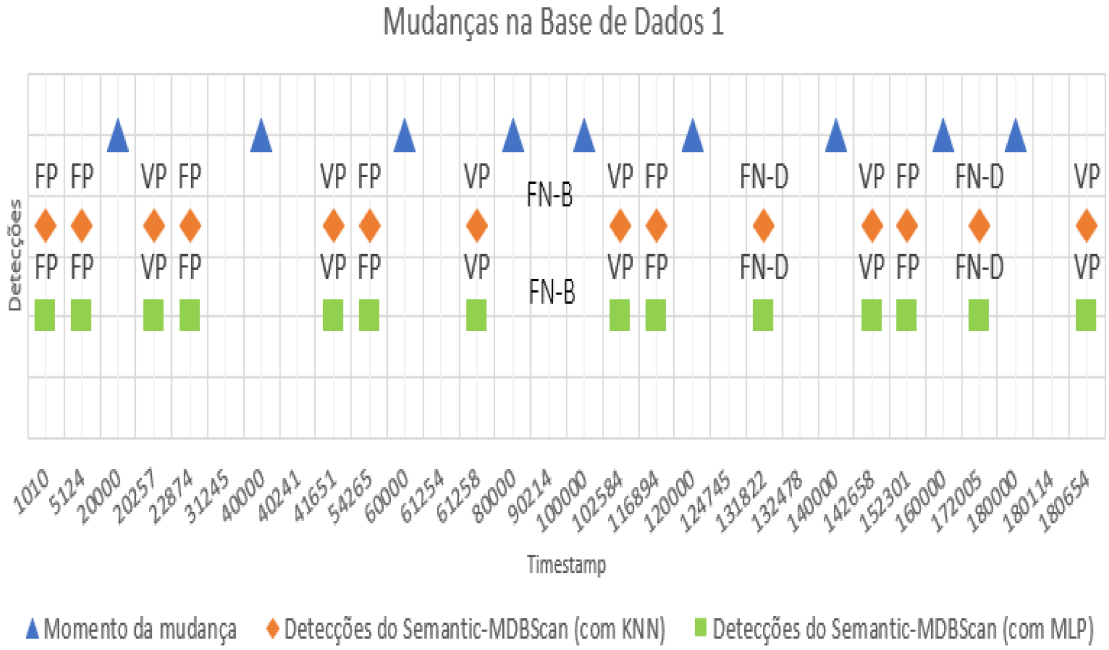


Figura 19 – Detecções do Semantic-MDBScan na Base de dados 1.

Sobre a Base de dados 6, que apresenta mudanças graduais/recorrentes, o Semantic-MDBScan, usando KNN ou MLP, apresentou F1 igual a 0,4 em ambas as abordagens. Esse valor menor de

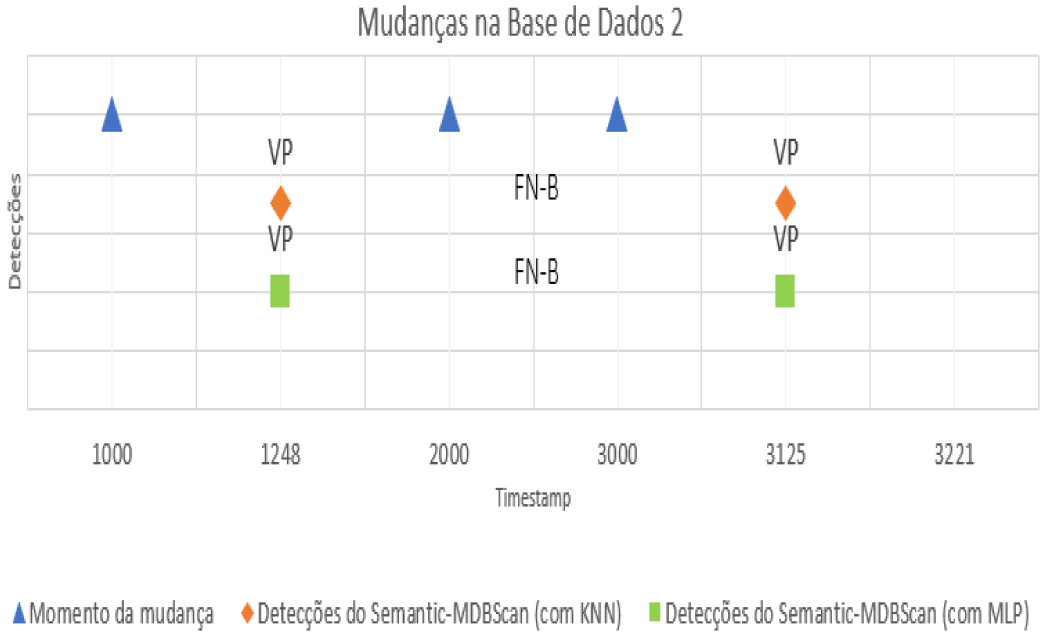


Figura 20 – Detecções do Semantic-MDBScan na Base de dados 2.

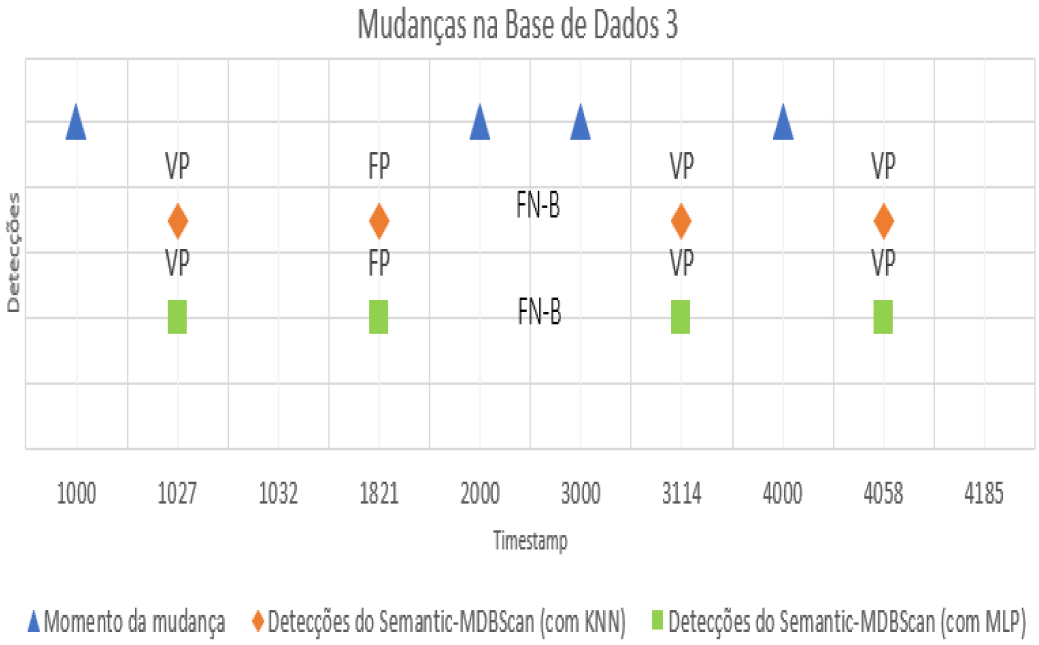


Figura 21 – Detecções do Semantic-MDBScan na Base de dados 3.

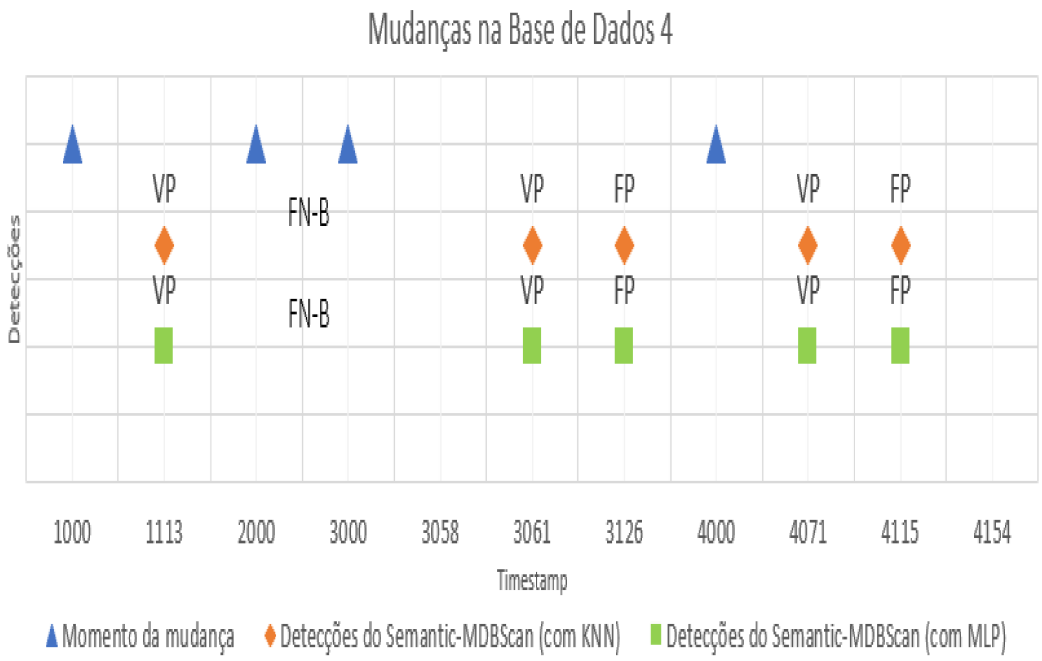


Figura 22 – Detecções do Semantic-MDBScan na Base de dados 4.

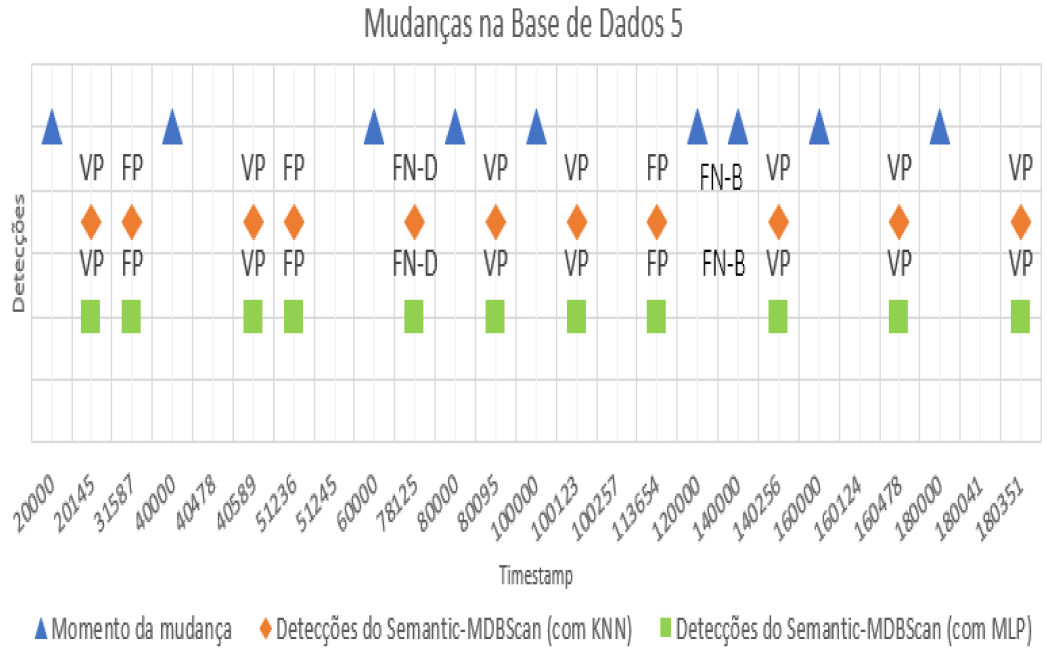


Figura 23 – Detecções do Semantic-MDBScan na Base de dados 5.

F1, se comparado às bases com mudanças abruptas, é justificado pelo cenário desafiador de mudanças graduais, que apresenta um período de transição em que há variações nas amostras antes que se atinja uma estabilidade semântica na sequência de dados. Esta queda no valor do F1 também é observável na Base de dados 7, com valor de 0,22 de F1 para o Semantic-MDBScan com MLP e de 0,11 para a versão com KNN. Detalhes sobre as detecções nas bases 6 e 7 são apresentados nas Figuras 24 e 25.

As únicas ocorrências de FN-S estão vinculadas aos experimentos com a base 7. Este é um resultado importante, pois o Semantic-MDBScan conseguiu um resultado satisfatório na tarefa de atribuir um significado a uma mudança de comportamento, que é o principal objetivo com o desenvolvimento deste novo algoritmo. Os FN-S ocorreram em uma base que representa um cenário mais desafiador por possuir mudanças graduais e o intervalo entre as mudanças (valor de m) ser muito grande, 20000 amostras. Além disso, mesmo utilizando o KNN que é um algoritmo de classificação mais simples que o MLP, por não demandar treinamento, os resultados foram satisfatórios na tarefa de atribuir semântica a uma mudança.

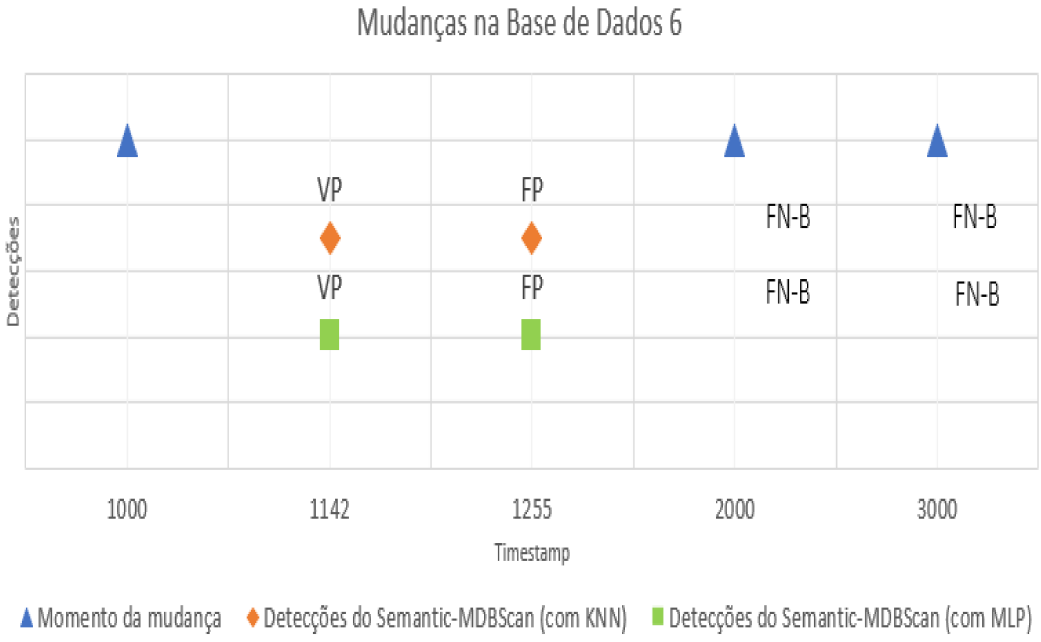


Figura 24 – Detecções do Semantic-MDBScan na Base de dados 6.

Para validar os resultados dos experimentos com o Semantic-MDBScan, foi aplicado o teste estatístico de Wilcoxon, utilizando para isso a ferramenta KEEL (ALCALÁ-FDEZ et al., 2011). A hipótese nula usada é a seguinte: *O Semantic-MDBScan com MLP apresenta um desempenho semelhante ao Semantic-MDBScan com KNN na tarefa de atribuir corretamente uma semântica às mudanças detectadas pelo algoritmo*. Os resultados apresentam R^+ igual a 17,5 e R^- igual a 10,5, com p-valor assintótico igual a 0,498, que não permite rejeitar a hipótese nula.

Mesmo sendo um método de classificação mais simples que o MLP e também tendo um custo computacional maior na etapa de classificação, o KNN apresentou bons resultados, por-

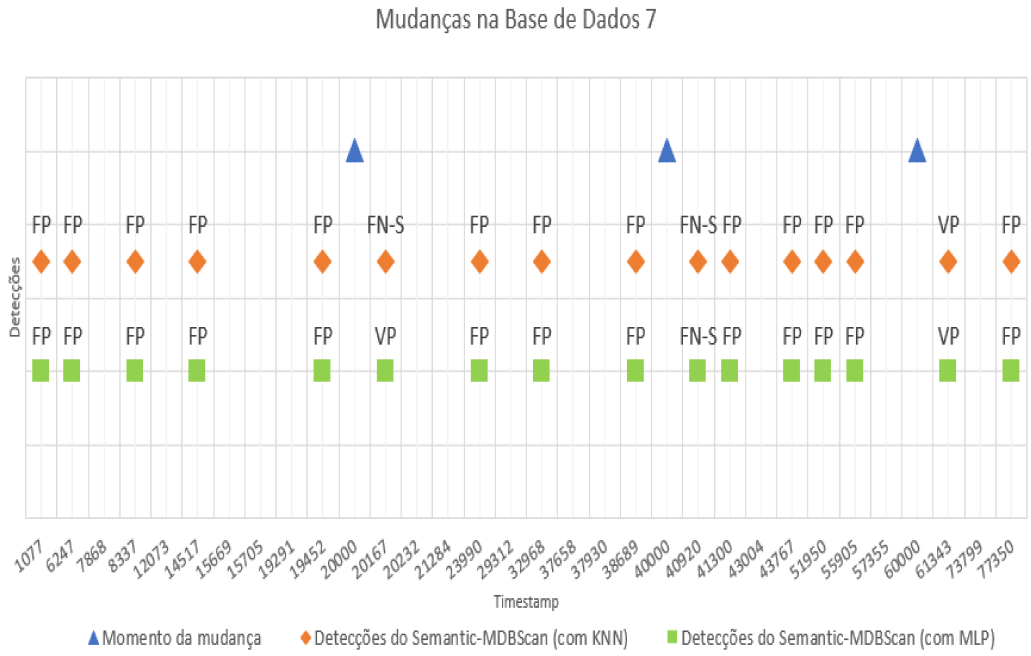


Figura 25 – Detecções do Semantic-MDBScan na Base de dados 7.

que a classificação ocorre somente com as amostras de dados que desencadearam uma mudança de comportamento. Por exemplo, em uma base com $m = 1000$, as amostras a serem classificadas representam 0,1% do total de amostras ou menos. Além disso, as características de um algoritmo incremental do KNN, o torna uma boa escolha de algoritmo de classificação para cenários dinâmicos, com o benefício de não ter uma etapa de treinamento como ocorre com o MLP.

Na seção seguinte são apresentados os resultados obtidos com os experimentos envolvendo o algoritmo SDS-MDBScan.

6.4.5 Experimentos com o SDS-MDBScan

Os experimentos descritos nesta seção visam testar a habilidade do SDS-MDBScan em detectar e associar semânticas às mudanças que ocorrem no cenário de fluxo contínuo de dados.

Para os experimentos, dezoito bases de dados foram usadas, sendo sete delas artificiais, compostas por amostras produzidas pelo framework MOA (BIFET et al., 2010). As bases de dados artificiais são as mesmas usadas nos experimentos com o Semantic-MDBScan (veja Seção 6.4.1). As demais são bases reais geradas a partir de partidas do StarCraft.

Assim como nos experimentos com o Semantic-MDBScan, o SDS-MDBScan também foi testados com dois métodos de classificação: KNN e MLP. O método de classificação é usado para definir a semântica das mudanças, cujo modo de uso foi descrito na Seção 6.3. Para auxiliar o processo de classificação de tais métodos, o SDS-MDBScan também usa as bases de dados rotulados chamadas *Training Dataset*, para auxiliar o treinamento da MLP e auxiliar o processo de classificação do KNN.

As bases de dados reais são compostas por amostras de dados obtidos de partidas do StarCraft, e a *Training Dataset* é composta por amostras de outras dez partidas do StarCraft, maiores detalhes são apresentados na Seção 6.4.5.1.

6.4.5.1 Bases de dados reais usadas nos experimentos com SDS-MDBScan

Nesta subseção são apresentadas informações sobre as onze bases de dados reais utilizadas nos experimentos com o SDS-MDBScan.

As amostras que compõem as bases de dados reais foram obtidas de partidas do StarCraft entre um jogador humano e um jogador automático representado pelo UAlbertaBot (CHURCHILL; BURO, 2011; CHURCHILL; SAFFIDINE; BURO, 2012).

As amostras nas bases reais são representadas por 8 atributos definidos empiricamente. Sendo que para criar uma amostra de dados em uma base real, cada atributo tem seu valor acumulado por um período que equivale a quinze frames do jogo. Quando se atinge este período, uma amostra é registrada na base e os atributos são zerados para a geração de uma nova amostra de dados. Então, uma amostra de dados sumariza os dados a cada quinze frames, e esse processo se repete até a conclusão da partida.

As bases reais apresentam mudança do tipo gradual/recorrente devido as características do próprio jogo StarCraft, onde as ações levam um tempo para serem concluídas como, por exemplo: construção de novas estruturas, movimentação de unidades pelo mapa, ou mesmo, o combate às unidades inimigas.

As Tabelas 9 e 10 apresentam em detalhes as bases reais R1 e R4. Estas bases foram escolhidas aleatoriamente para exemplificar as bases reais em uma análise mais detalhada. Estas tabelas possuem as seguintes colunas: coluna *ChangeID* que enumera as mudanças de comportamento em cada base; o *Timestamp* indica o momento da mudança, apresentando a quantidade de amostras lidas do fluxo até esse momento da mudança; e a coluna *Semântica* indica a semântica que prevalece na sequência de dados a partir da mudança.

6.4.5.2 *Training Dataset* para os experimentos com o SDS-MDBScan

Assim como foi apresentado para o Semantic-MDBScan, foram utilizadas *Training Dataset* para treinar as redes MLP e auxiliar os classificadores KNN. Como foram definidas dezoito bases (sete artificiais e onze reais) para experimentos, então dezoito redes MLP foram treinadas usando para isso as suas correspondentes *Training Datasets*. O KNN também contará com uma *Training Dataset* para cada base de dados usada nos experimentos. As *Training Dataset* para as bases artificiais, são as mesmas usadas nos experimentos com o Semantic-MDBScan, apresentados na Seção 6.4.1.

As *Training Dataset* para as bases reais foram construídas com amostras de dez bases reais distintas daquela usada como base de teste. Então, cada uma das 11 bases reais será, em algum momento, usada como base de teste e as demais serão usadas para compor a *Training Dataset*. Nas bases reais, cada rótulo de comportamento será representado por 100 amostras nas *Training*

Tabela 9 – Descrição da base real R1

Base de dados	ChangeID	Timestamp	Semântica
Base de dados R1	1	160	Strategic Defensive
	2	167	Unknown
	3	1021	Aggressive Offensive
	4	1053	Unknown
	5	1062	Aggressive Offensive
	6	1075	Unknown
	7	1081	Aggressive Offensive
	8	1102	Unknown
	9	1115	Aggressive Offensive
	10	1125	Unknown
	11	1227	Aggressive Offensive
	12	1401	Unknown
	13	1463	Aggressive Offensive
	14	1489	Unknown
	15	1522	Aggressive Offensive
	16	1586	Unknown
	17	1681	Aggressive Offensive
	18	1761	Unknown
	19	1798	Aggressive Offensive
	20	1845	Unknown
	21	1886	Aggressive Offensive
	22	1965	Unknown
	23	2032	Aggressive Offensive
	24	2076	Unknown
	25	2100	Aggressive Offensive
	26	2148	Unknown
	27	2292	Aggressive Offensive
	28	2301	Unknown
	29	2547	Aggressive Offensive
	30	2670	Unknown

Datasets geradas para estas bases. Essa quantidade foi definida empiricamente com base na quantidade de amostras usadas para representar os comportamentos nas *Training Dataset* das bases artificiais.

Os resultados dos experimentos serão apresentados em duas análises: uma sobre os resultados com bases artificiais, descrita na Seção 6.4.5.3; a outra é sobre os resultados com as bases reais, descrita na Seção 6.4.5.4.

6.4.5.3 Análise dos experimentos do SDS-MDBScan com bases artificiais

A Tabela 11 apresenta os resultados dos experimentos com bases artificiais que avaliam a capacidade do SDS-MDBScan com KNN em atribuir um significado para as mudanças que ocorreram nas bases descritas na Tabela 6. A Tabela 12 apresenta os resultados com as mesmas

Tabela 10 – Descrição da base real R4

Base de dados	ChangeID	Timestamp	Semântica
Base de dados R4	1	177	Strategic Defensive
	2	184	Unknown
	3	447	Aggressive Offensive
	4	488	Unknown
	5	903	Aggressive Offensive
	6	944	Unknown
	7	1069	Aggressive Offensive
	8	1075	Unknown
	9	1081	Aggressive Offensive
	10	1085	Unknown
	11	1140	Defensive Aggressive
	12	1147	Unknown
	13	1221	Aggressive Offensive
	14	1239	Unknown

bases artificiais, mas produzidos pelo SDS-MDBScan com MLP para definir a semântica das mudanças.

Tabela 11 – Resultados de experimentos com bases artificiais: *SDS-MDBScan com KNN*

Tipo de mudança	Base de dados	Medidas				
		VP	FP	FN-B / D / S	VN	F1
Abrupta	**Base de dados 1 (9 mudanças)**	6	3	0 / 3 / 0	199988	0.66
	Base de dados 2 (3 mudanças)	2	0	1 / 0 / 0	3997	0.8
	Base de dados 3 (4 mudanças)	3	0	1 / 0 / 0	4996	0.85
	Base de dados 4 (4 mudanças)	3	1	1 / 0 / 0	4995	0.75
	Base de dados 5 (9 mudanças)	6	1	2 / 1 / 0	199990	0.75
	Média de F1-score (Abrupta)					**0.76**
Gradual	**Base de dados 6 (3 mudanças)**	1	0	2 / 0 / 0	3997	0.5
	Base de dados 7 (3 mudanças)	2	9	0 / 1 / 0	79988	0.28
	Média de F1-score (Gradual)					**0.39**

Analisando os resultados obtidos pelos experimentos com as bases artificiais e com mudanças abruptas/recorrentes (bases 1, 2, 3, 4 e 5), o valor de F1 varia entre 0,66 a 0,85. Considerando os resultados obtidos pelos experimentos com bases artificiais e mudanças do tipo gradual/recorrente (bases 6 e 7), o valor de F1 foi de 0,28 para a base 7 (precisão igual a 0,18 e revocação igual a 0,66) e 0,5 para a base 6 (precisão igual a 1 e revocação igual a 0,3). É importante destacar que o SDS-MDBScan com KNN ou MLP apresentaram os mesmos resultados pois o aspecto que poderia variar é a atribuição semântica representada pela medida FN-S, contudo não houve nenhuma ocorrência de erro na definição da semântica de uma mudança. Mais detalhes sobre as detecções nas bases artificiais podem ser observados nas Figuras 26, 27, 28, 29, 30, 31 e 32.

Analisando os resultados obtidos com experimentos sobre as bases 6 e 7, é observável uma

Tabela 12 – Resultados dos experimentos com bases artificiais: *SDS-MDBScan com MLP*

Tipo de mudança	**Base de dados**	**Medidas**				
		VP	**FP**	**FN-B / D / S**	**VN**	**F1**
Abrupta	**Base de dados 1 (9 mudanças)**	6	3	0 / 3 / 0	199988	0.66
	Base de dados 2 (3 mudanças)	2	0	1 / 0 / 0	3997	0.8
	Base de dados 3 (4 mudanças)	3	0	1 / 0 / 0	4996	0.85
	Base de dados 4 (4 mudanças)	3	1	1 / 0 / 0	4995	0.75
	Base de dados 5 (9 mudanças)	6	1	2 / 1 / 0	199990	0.75
	Média de F1-score (Abrupta)					**0.76**
Gradual	**Base de dados 6 (3 mudanças)**	1	0	2 / 0 / 0	3997	0.5
	Base de dados 7 (3 mudanças)	2	9	0 / 1 / 0	79988	0.28
	Média de F1-score (Gradual)					**0.39**

redução no valor de F1 se comparados aos valores dessa medida obtidos nos experimentos com as bases compostas por mudanças abruptas/recorrentes. Essa redução no valor de F1 é explicado pelo cenário desafiador de mudanças graduais/recorrentes presentes nas bases 6 e 7. No caso da base 7, em que a quantidade de FPs foi muito alta, destaca-se como um dificultador o alto valor de m (quantidade de amostras entre duas mudanças consecutivas), que nesta base era igual a 20000, e que consequentemente, gera um período de transição de mudanças mais longo, dificultando as detecções, por neste período não prevalecer nenhuma tendência de comportamento, sendo composta por amostras de dados geradas por diferentes distribuições.

Os resultados com as bases artificiais 6 e 7 mostraram que também não ocorreu erro na definição da semântica da mudança, consequentemente o SDS-MDBScan com KNN ou MLP apresentaram os mesmos resultados, mas agora usando as bases artificiais com mudanças graduais. Maiores detalhes sobre as detecções nas bases 6 e 7 são apresentadas nas Figuras 31 e 32.

Com o SDS-MDBScan foram feitos experimentos com as mesmas bases artificiais usadas pelo Semantic-MDBScan, o que torna possível uma comparação direta entre as abordagens. Os resultados do Semantic-MDBScan com KNN estão descritos na Tabela 7 e os resultados do Semantic-MDBScan com MLP estão na Tabela 8.

Nesta comparação direta entre o Semantic-MDBScan e o SDS-MDBScan observa-se que o valor de F1 do SDS-MDBScan apresentou melhores resultados em todos os experimentos com exceção dos resultados com a base 2, onde as duas abordagens tiveram valores iguais com a medida F1. Esta comparação evidencia que os critérios usados pelo SDS-MDBScan, para declarar uma mudança de comportamento e atribuir uma semântica à mudança, contribuíram para esses melhores resultados, visto que os demais procedimentos, como os da etapa de agrupamento, são comuns entre as duas abordagens.

Também é observável que o SDS-MDBScan teve redução de 100% das ocorrências de FN-S nas bases artificiais, se comparado ao resultado obtido pelo Semantic-MDBScan que tinha apresentado 2 FN-S na abordagem com KNN e 1 FN-S na abordagem com MLP, ambos com a base 7.

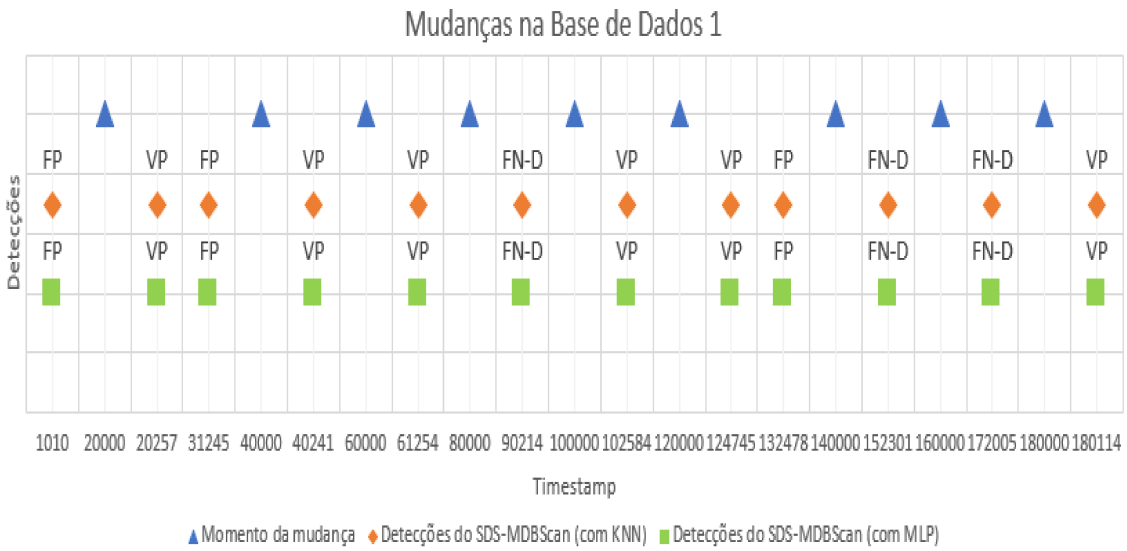


Figura 26 – Detecções do SDS-MDBScan na Base de dados 1.

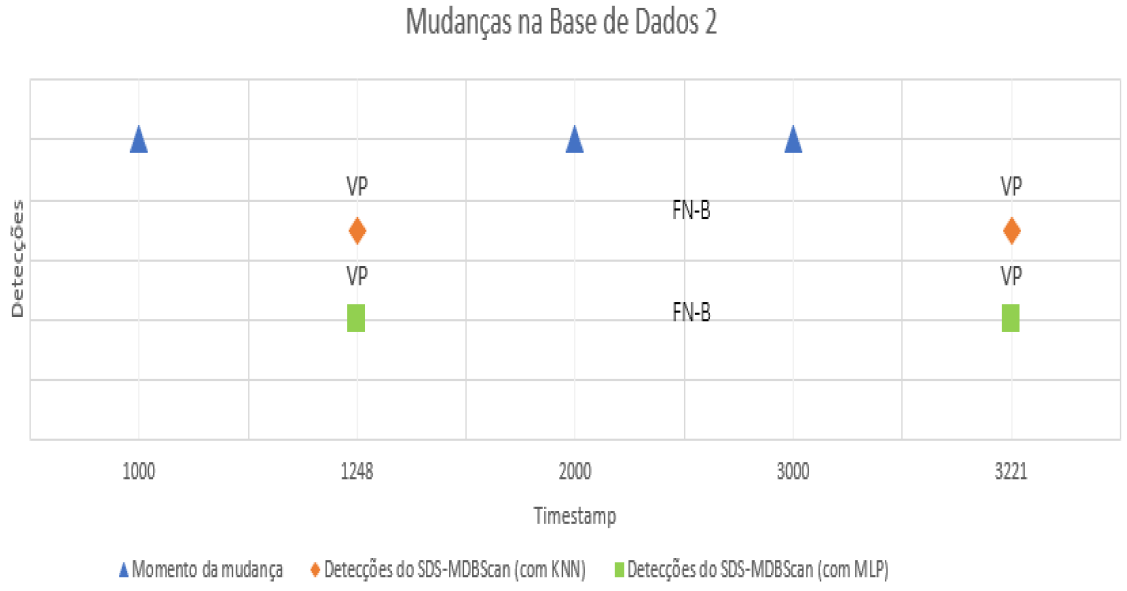


Figura 27 – Detecções do SDS-MDBScan na Base de dados 2.

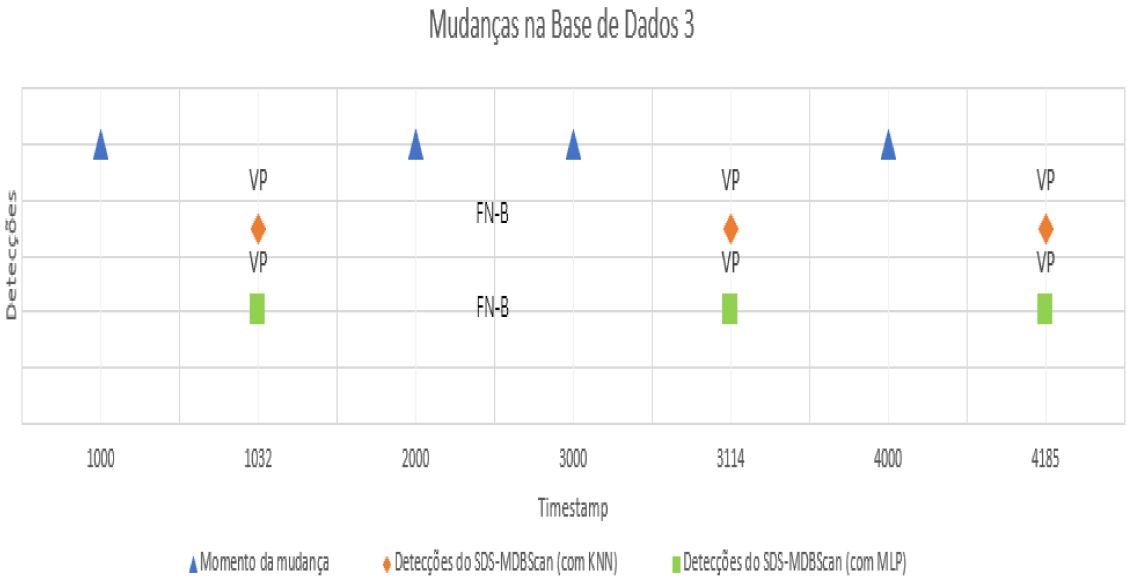


Figura 28 – Detecções do SDS-MDBScan na Base de dados 3.

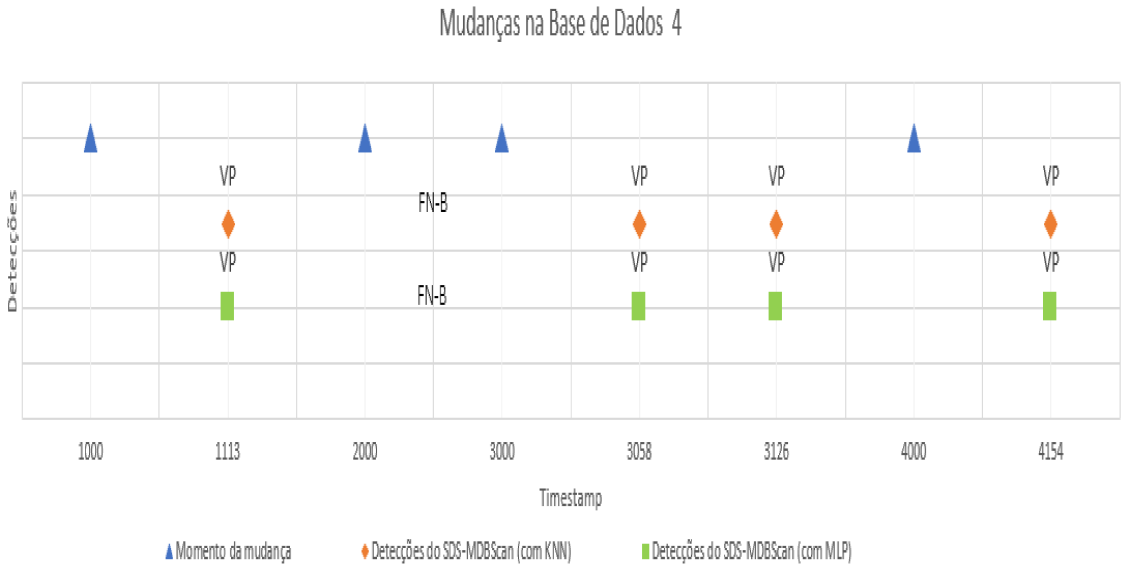


Figura 29 – Detecções do SDS-MDBScan na Base de dados 4.

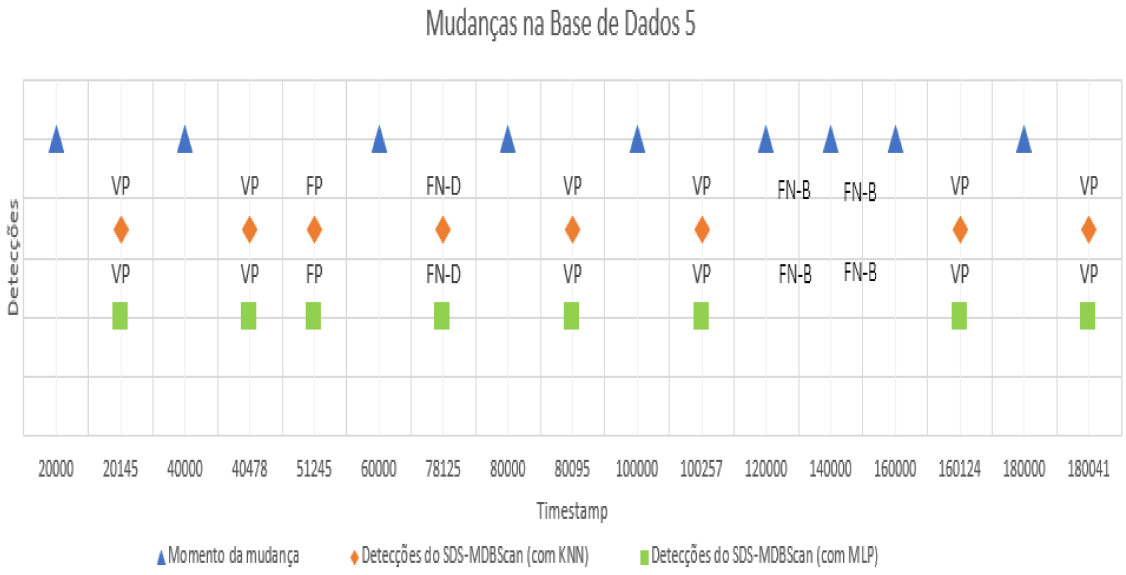


Figura 30 – Detecções do SDS-MDBScan na Base de dados 5.

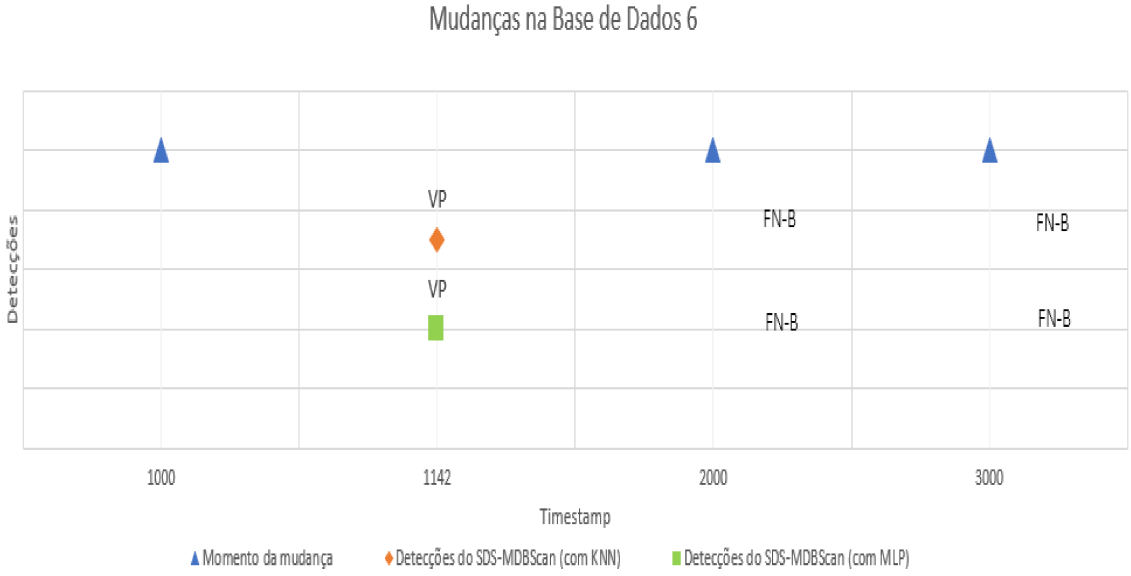


Figura 31 – Detecções do SDS-MDBScan na Base de dados 6.

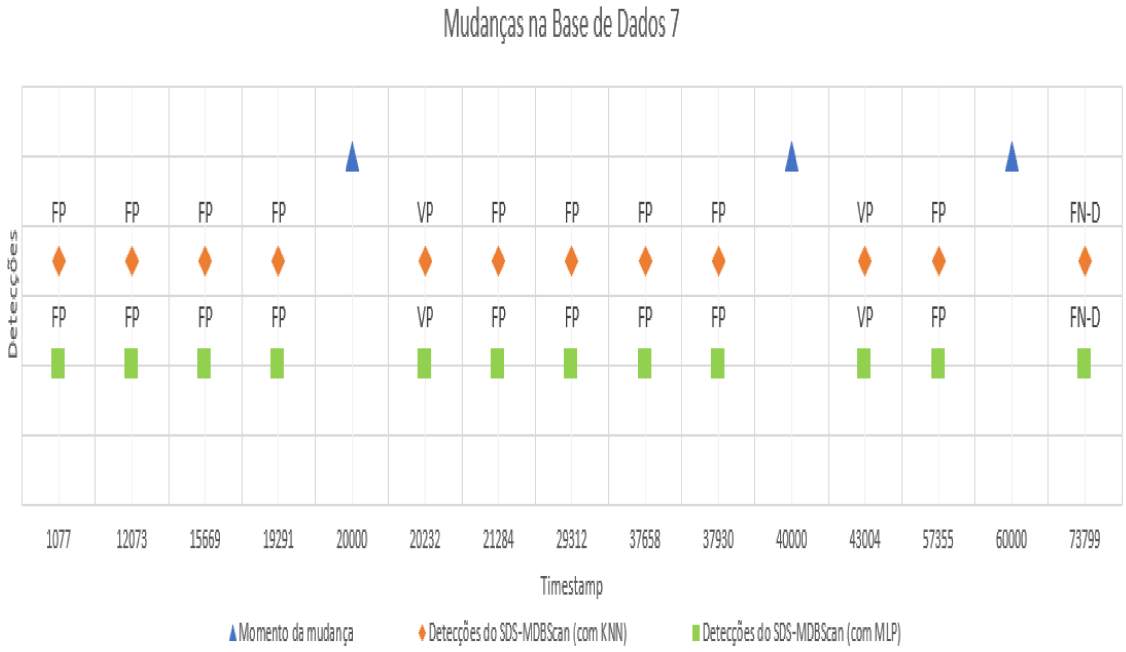


Figura 32 – Detecções do SDS-MDBScan na Base de dados 7.

O valor de F1 desses experimentos com bases artificiais foram usados no teste estatístico de Wilcoxon, comparando os resultados do Semantic-MDBScan e do SDS-MDBScan nas suas respectivas abordagens com KNN e MLP. Para aplicar o teste de Wilcoxon foi usada a ferramenta KEEL (ALCALÁ-FDEZ et al., 2011)).

A primeira hipótese nula considerada sobre o SDS-MDBScan é a seguinte: *O SDS-MDBScan com KNN apresenta resultados semelhantes aos obtidos pelo Semantic-MDBScan com KNN*. Os resultados obtidos são R^+ igual a 21,0 e R^- igual a 0, com um p-valor assintótico igual a 0,01787, que permite a rejeição da hipótese nula. Considerando as Tabelas 7 e 11, pode ser observado que as médias do *F1-score* do SDS-MDBScan com KNN são sempre maiores que as médias de *F1-score* do Semantic-MDBScan com KNN, permitindo a formulação da seguinte hipótese alternativa: *O SDS-MDBScan com KNN apresentou uma performance superior a apresentada pelo Semantic-MDBScan com KNN*. Então, como ocorreu a rejeição da hipótese nula, a hipótese alternativa pode ser afirmada.

A segunda hipótese nula considerada na comparação direta entre as abordagens do SDS-MDBScan é a seguinte: *O SDS-MDBScan com MLP apresenta resultados semelhantes aos obtidos pelo Semantic-MDBScan com MLP*. Os resultados obtidos são R^+ igual a 21,0 e R^- igual a 0, com um p-valor assintótico igual a 0,01787, que permite a rejeição da hipótese nula. Considerando as Tabelas 8 e 12, pode ser observado que as médias do *F1-score* do SDS-MDBScan com MLP são sempre maiores que as médias de *F1-score* do Semantic-MDBScan com MLP, permitindo a formulação da seguinte hipótese alternativa: *O SDS-MDBScan com MLP apresentou uma performance superior a apresentada pelo Semantic-MDBScan com MLP*. Então, como ocorreu a rejeição da hipótese nula, a hipótese alternativa pode ser afirmada.

Na seção seguinte são apresentados os resultados do SDS-MDBScan nos experimentos com as bases reais.

6.4.5.4 Análise dos experimentos do SDS-MDBScan com bases reais

Como descrito na Seção 6.4.5.1, as bases reais foram geradas a partir de dados de partidas do jogo StarCraft. Nestas bases temos mudanças dos tipos gradual/recorrente, que definem um cenário mais desafiador para os experimentos.

Diferentemente das bases artificiais, as bases reais não apresentam um valor constante de m como pode ser observado nas Tabelas 9 e 10. Isto impactará na análise da medida FN-D, visto que o atraso é definido em função do valor de m.

As bases de dados R1 e R4 foram escolhidas para uma análise detalhada (Figuras 33 e 34), por possuir quantidades bem diferentes de mudanças, 30 mudanças na base R1 e 14 na base R4. Os resultados dos experimentos com as demais bases estão apresentados nas Tabelas 13 e 14.

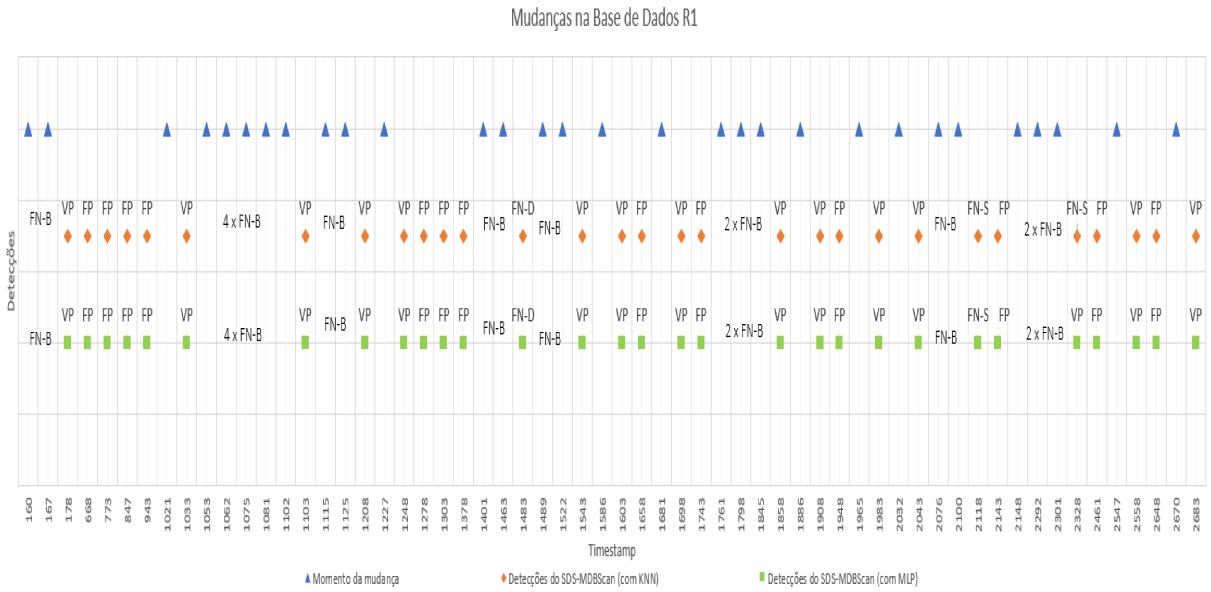


Figura 33 – Detecções do SDS-MDBScan na base de dados R1.

Considerando a ocorrência de FN-S, é interessante notar a relação entre VP e FN-S, visto que se não ocorresse o erro de detecção semântica, esta detecção seria um VP. Analisando em detalhes os resultados com as bases reais R1 e R4, o SDS-MDBScan com KNN teve, respectivamente, 2 e 1 FN-S, 14 e 6 VPs. O SDS-MDBScan com MLP teve 1 FN-S e 15 VPs com a base R1 e 1 FN-S e 6 VPs com a base R4 Maiores detalhes das detecções sobre as bases R1 e R4 estão apresentados nas Figuras 33 e 34.

Considerando os demais experimentos, as bases R7, R8, e R10, em ambas as abordagens do SDS-MDBScan, não tiveram nenhuma ocorrência de FN-S.

Porém, as bases R2, R3, R5, R6 e R9 apresentaram, respectivamente, as seguintes ocorrências de FN-S com a abordagem SDS-MDBScan com KNN, 1, 2, 2, 2 e 1. Enquanto que com a

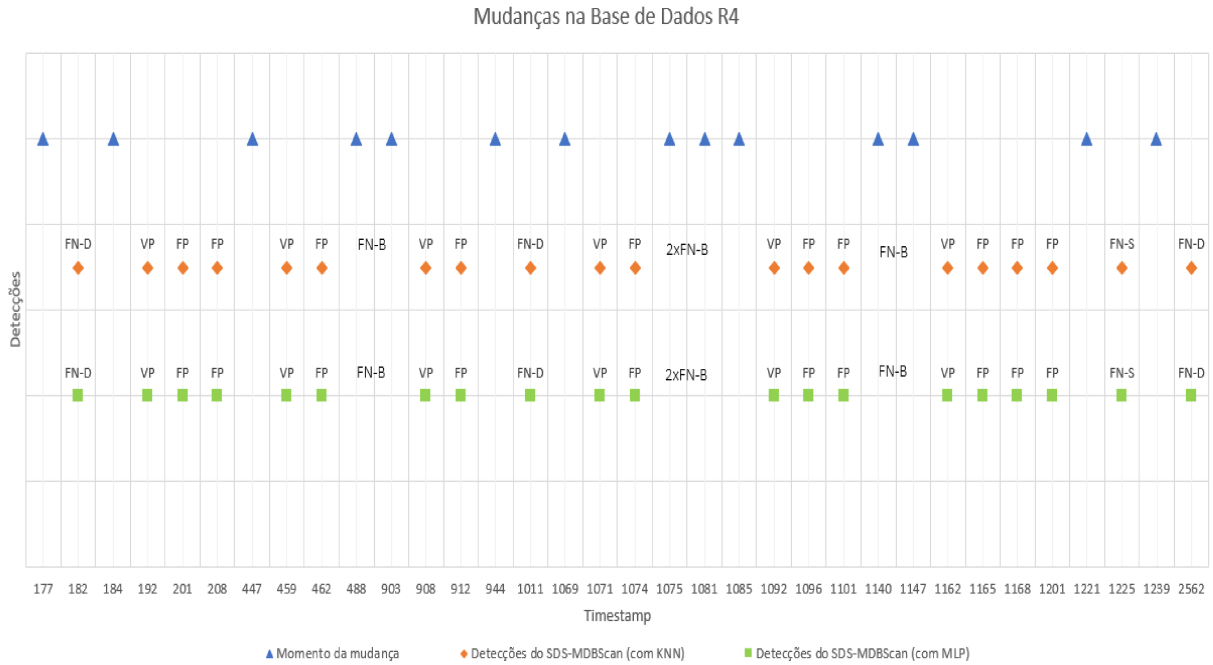


Figura 34 – Detecções do SDS-MDBScan na base de dados R4.

Tabela 13 – Resultados dos experimentos com bases reais: *SDS-MDBScan com KNN*

Tipo de mudança	**Base de dados**	**Medidas**				
		VP	**FP**	**FN-B / D / S**	**VN**	**F1**
Gradual	**Base de dados R1 (30 mudanças)**	14	13	8 / 6 / 2	2739	0.49
	Base de dados R2 (16 mudanças)	9	6	4 / 2 / 1	1465	0.58
	Base de dados R3 (18 mudanças)	8	7	5 / 3 / 2	1290	0.48
	Base de dados R4 (14 mudanças)	6	10	4 / 3 / 1	1349	0.4
	Base de dados R5 (38 mudanças)	17	13	13 / 6 / 2	6616	0.5
	Base de dados R6 (50 mudanças)	23	21	14 / 11 / 2	2868	0.49
	Base de dados R7 (26 mudanças)	16	11	6 / 4 / 0	1753	0.6
	Base de dados R8 (18 mudanças)	11	12	4 / 3 / 0	1411	0.53
	Base de dados R9 (20 mudanças)	11	9	5 / 3 / 1	2195	0.55
	Base de dados R10 (6 mudanças)	3	4	2 / 1 / 0	1406	0.46
	Base de dados R11 (34 mudanças)	16	14	12 / 4 / 2	2734	0.5
	Média do F1					**0.5**
	Desvio padrão do F1					**0.05**

Tabela 14 – Resultados dos experimentos com bases reais: *SDS-MDBScan com MLP*

Tipo de mudança	**Base de dados**	**Medidas**				
		VP	**FP**	**FN-B / D / S**	**VN**	**F1**
Gradual	**Base de dados R1 (30 mudanças)**	15	13	8 / 6 / 1	2739	0.51
	Base de dados R2 (16 mudanças)	9	6	4 / 2 / 1	1465	0.58
	Base de dados R3 (18 mudanças)	10	7	5 / 3 / 0	1290	0.57
	Base de dados R4 (14 mudanças)	6	10	4 /3 / 1	1349	0.4
	Base de dados R5 (38 mudanças)	17	13	13 / 6 / 2	6616	0.5
	Base de dados R6 (50 mudanças)	24	21	14 / 11 / 1	2868	0.5
	Base de dados R7 (26 mudanças)	16	11	6 / 4 / 0	1753	0.6
	Base de dados R8 (18 mudanças)	11	12	4 / 3 / 0	1411	0.53
	Base de dados R9 (20 mudanças)	12	9	5 / 3 / 0	2195	0.58
	Base de dados R10 (6 mudanças)	3	4	2 / 1 / 0	1406	0.46
	Base de dados R11 (34 mudanças)	17	14	12 / 4 / 1	2734	0.52
	Média do F1					**0.52**
	Desvio padrão do F1					**0.05**

abordagem SDS-MDBScan com MLP, os experimentos com estas mesmas bases apresentaram uma redução, na maioria dos casos, na medida FN-S, apresentando, respectivamente, as quantidades 1, 0, 2, 1 e 0. As únicas bases que mantiveram a mesma quantidade de FN-S, nas duas abordagens do SDS-MDBScan, foram as bases R2 e R5.

Lembrando que a principal contribuição do SDS-MDBScan é atribuir a uma mudança uma semântica que seja relevante estatisticamente, então, a pequena quantidade de FN-S nos experimentos demonstram que esta tarefa foi satisfeita.

Realizando uma análise geral, considerando os dados das Tabelas 13 e 14, o SDS-MDBScan com KNN apresentou uma média de F1 igual a 0,5 com desvio padrão de 0,05, a abordagem com MLP apresentou uma média de F1 igual a 0,52 e desvio padrão de 0,05, demonstrando resultados bem similares das duas abordagens nos experimentos sobre as 11 bases de dados utilizadas.

Com o propósito de analisar se existe uma diferença estatística entre os resultados produzidos pelo SDS-MDBScan com KNN e MLP sobre as 18 bases utilizadas, o teste de Wilcoxon foi aplicado usando a ferramenta KEEL.

A hipótese nula usada é a seguinte: *O SDS-MDBScan com MLP apresenta o mesmo desempenho que o SDS-MDBScan com KNN quanto a correta detecção de mudanças e a correta atribuição de semânticas a estas mudanças*. Os resultados obtidos são R^+ igual a 114 e R^- igual a 39, com um p-valor assintótico igual a 0,0703657, que não permite a rejeição da hipótese nula. Então, não é possível afirmar que alguma das versões do SDS-MDBScan seja melhor. Contudo, mesmo sendo um algoritmo mais simples que a MLP, o KNN se mostrou um classificador interessante para o problema analisado, devido as suas características de algoritmo incremental, e por não demandar treinamento como ocorre com a MLP.

Na seção a seguir é apresentada uma avaliação que envolve o uso de diferentes configura-

ções, relacionando tais alterações com medidas como VP, FP, entre outras.

6.4.6 Avaliação do SDS-MDBScan com diferentes configurações

Uma vez que os resultados do SDS-MDBScan com as bases artificiais e reais foram apresentadas, uma análise utilizando diferentes configurações e que afetam os experimentos de ambos os tipos de bases de dados será apresentada nesta seção. O número de novidades é um parâmetro que pode influenciar a quantidade de VP, FP, VN e FN.

As bases artificiais 4 e 5 foram usadas em uma análise baseada na variação do número mínimo de novidades para declarar uma mudança de comportamento. A escolha destas bases foi aleatória, visando analisar bases com quantidades diferentes de mudanças de comportamento, a base 4 possui quatro mudanças e a base 5 possui 9 mudanças. Nesta análise foram usadas as quantidades de 2, 4 e 6 novidades, como sendo o mínimo para indicar a ocorrência de uma mudança de comportamento utilizando o SDS-MDBScan com KNN. O tamanho da janela deslizante permaneceu constante.

Nesta análise, foi observado que quando a quantidade mínima de novidades é aumentada, a quantidade de FPs e VPs reduz, uma vez que fica mais rígido o critério para declarar uma mudança de comportamento, afetando qualquer detecção, seja verdadeira ou falsa. Consequentemente, o número de FNs e VNs aumentam. Contudo, quando a quantidade mínima de novidades é reduzida, a quantidade de FPs e VPs sofre elevação, uma vez que fica menos restritivo indicar a ocorrência de uma mudança de comportamento. O número de FNs e VNs sofrem redução neste cenário. Esta análise pode ser observada, em detalhes, na Tabela 15.

Tabela 15 – Resultado do experimento: Variando a quantidade mínima de novidades

Quantidade de novidades	**Base de dados**	**Medidas**			
		VP	**FP**	**FN**	**VN**
2 novidades	**Base de dados 4 (4 mudanças)**	3	1	1	4995
	Base de dados 5 (9 mudanças)	6	1	3	199990
4 novidades	**Base de dados 4 (3 mudanças)**	2	0	2	4996
	Base de dados 5 (9 mudanças)	4	1	5	199990
6 novidades	**Base de dados 4 (3 mudanças)**	0	0	3	4997
	Base de dados 5 (9 mudanças)	2	0	7	199991

6.4.7 Tempo de execução do algoritmo

Nesta seção são apresentados os tempos de execução dos experimentos envolvendo o SDS-MDBScan com KNN e MLP para efeito de comparação. Este tempo de execução não inclui o tempo de treinamento das redes MLP, ele só representa o tempo total de processamento das amostras de dados das bases dos experimentos.

Estes experimentos foram executados em um computador com 16 GB de memória RAM, utilizando o processador Intel(R) Core(TM) i7-8565U. O código do programa, as bases de

dados e as *Training Dataset* estavam armazenados em um solid-state drive (SSD) ao longo de toda a execução.

A Tabela 16 apresenta os tempos de execução dos experimentos com as bases artificiais e reais. Nesta tabela há uma comparação direta entre os tempos do SDS-MDBScan com KNN e do SDS-MDBScan com MLP, sendo observável que a abordagem com MLP apresentou um desempenho melhor do que a abordagem com KNN, com uma média de tempo de execução de 336,543s contra a média de 394,755s obtido pela abordagem com KNN.

Para demonstrar a relevância estatística desta comparação dos tempos de execução, o teste de Wilcoxon (DERRAC et al., 2011) foi aplicado sobre os dados apresentados na Tabela 16, sendo adotada a seguinte hipótese nula: *O SDS-MDBScan com MLP apresenta o mesmo desempenho, em termos de tempo de execução, que a abordagem do SDS-MDBScan com KNN*. Os resultados obtidos são R^+ igual a 171 e R^- igual a 0, com um p-valor assintótico igual a 0.000153, que permite a rejeição da hipótese nula. Baseado na média do tempo de execução apresentado na Tabela 16, a seguinte hipótese alternativa foi elaborada: *O SDS-MDBScan com MLP apresentou uma performance superior, em termos de tempo de execução, que a abordagem SDS-MDBScan com KNN*. Uma vez que a hipótese nula foi rejeitada, a hipótese alternativa pode ser afirmada.

Tabela 16 – Tempo de execução dos experimentos com SDS-MDBScan

	Base de dados	**Tempo de execução do SDS-MDBScan com KNN**	**Tempo de execução do SDS-MDBScan com MLP**
Artificial	**Base de dados 1 (9 mudanças)**	2710,784s	2309,418s
	Base de dados 2 (3 mudanças)	54,216s	46,188s
	Base de dados 3 (4 mudanças)	67,770s	57,735s
	Base de dados 4 (4 mudanças)	67.770s	57.735s
	Base de dados 5 (9 mudanças)	2710,784s	2309,418s
	Base de dados 6 (3 mudanças)	54,216s	46,188s
	Base de dados 7 (3 mudanças)	1084,313s	923,767s
Real	**Base de dados R1 (30 mudanças)**	37,707s	32,213s
	Base de dados R2 (16 mudanças)	20,155s	17,218s
	Base de dados R3 (18 mudanças)	17,823s	15,226s
	Base de dados R4 (14 mudanças)	18,610s	15,898s
	Base de dados R5 (38 mudanças)	90,364s	77,198s
	Base de dados R6 (50 mudanças)	39,835s	34,031s
	Base de dados R7 (26 mudanças)	24,262s	20,727s
	Base de dados R8 (18 mudanças)	19,531s	16,685s
	Base de dados R9 (20 mudanças)	30,144s	25,752s
	Base de dados R10 (6 mudanças)	19,192s	16,396s
	Base de dados R11 (34 mudanças)	38,124s	35,984s
Média do tempo de execução		**394,755s**	**336,543s**

Então, considerando o fator tempo de execução, a abordagem com MLP apresentou melho-

res resultados, contudo, é importante destacar que as redes MLP demandam treinamento que é uma etapa importante e não foi considerada nesta avaliação temporal. Além disso, em um cenário dinâmico, onde os dados podem, por exemplo, apresentar mudanças de conceito, isso implicará em uma queda no desempenho do classificador MLP, demandando um novo período de treinamento e talvez uma reestruturação da rede, incorporando mais neurônios ou camadas. Enquanto que o algoritmo KNN, já possui a característica de ser incremental, tornando-o uma abordagem adequada para cenários dinâmicos como os contemplados nesta pesquisa. Além disso, há os resultados apresentados na Seção 6.4.5.4 que foram favoráveis tanto para a abordagem MLP quanto para a KNN, o algoritmo KNN se torna bastante adequado para o cenário de jogos RTS. Então, considerando as propriedades de um algoritmo incremental um fator decisivo, a abordagem SDS-MDBScan com KNN foi a escolhida para ser incorporada ao motor do StarCraft, cujos resultados são apresentados no próximo capítulo.

6.5 Considerações finais

Pelos resultados dos experimentos com o Semantic-MDBScan na etapa 3 é observável que o objetivo de desenvolver uma técnica capaz de identificar mudanças de comportamento e de atribuir um significado a estas mudanças já foi atingido com sucesso, pelo desenvolvimento desta primeira abordagem. Além disso, independentemente do algoritmo de classificação usado com Semantic-MDBScan, MLP ou KNN, e do cenário em que ocorrem as mudanças que podem ser abruptas ou recorrentes, o Semantic-MDBScan se mostrou um algoritmo adequado para lidar com problemas do mundo real.

Contudo, o desenvolvimento do SDS-MDBScan, ainda na etapa 3, conseguiu aprimorar os resultados obtidos com Semantic-MDBScan, reduzindo erros de atribuição de semântica às mudanças, bem como a ocorrência de FP e FN-S, uma vez que o SDS-MDBScan utiliza critérios mais rígidos para declarar a ocorrência de uma mudança, e também, mais confiáveis para determinar a semântica da mudança. O SDS-MDBScan também foi testado em bases reais com cenários de teste mais desafiadores com mudanças graduais, apresentado bons resultados. Estes resultados serviram para validar a seguinte hipótese, apresentada na Seção 1.3: “A acurácia do M-DBScan pode ser aumentada por meio da criação de abordagens alternativas de detecção de mudanças que expandam o referido algoritmo nos seguintes aspectos: habilidade para identificar o significado das mudanças detectadas; uso de um novo critério estatístico, baseado na reincidência de um mesmo significado em uma sequência de novidades, como lastro para apontar novas mudanças de comportamento.”

As abordagens do SDS-MDBScan com KNN ou MLP, não apresentaram diferenças significativas quanto ao seu desempenho. Entretanto, existem fatores que pesam a favor da abordagem com o KNN, como a característica de algoritmo incremental, que o torna fácil de se adaptar a cenários dinâmicos, além de não ter a necessidade de um novo treinamento. Enquanto que a abordagem com o MLP, diante das mudanças de um cenário dinâmico, demandaria um novo

treinamento da rede neural, e talvez uma reestruturação da rede, incorporando novos neurônios ou novas camadas à rede. Por isso, o SDS-MDBScan com KNN é a versão escolhida para a incorporação do módulo semântico ao motor do jogo StarCraft, na próxima etapa da pesquisa apresentada no capítulo 7.

CAPÍTULO 7

Etapa 4: Incorporando o SDS-MDBScan ao StarCraft

Nesta etapa 4 da pesquisa tem-se o objetivo de implementar um agente que opera em ambientes de disputa e dotado das seguintes habilidades: detectar as ocorrências de mudança de comportamento de seus oponentes; identificar os significados de tais mudanças; e usar tais significados como informação adicional para nortear suas tomadas de decisão.

O jogo RTS StarCraft foi o escolhido como estudo de caso, porque em jogos RTS, a habilidade de detectar mudanças no comportamento do adversário é essencial para garantir o sucesso do agente (YANNAKAKIS; MARAGOUDAKIS, 2005; VALLIM et al., 2013). Por exemplo, quando o agente percebe que seu adversário se tornou mais agressivo, ele pode alterar o seu estilo de jogo executando ações mais defensivas.

Nesta etapa 4 será apresentado o desenvolvimento de um novo agente StarCraft cuja arquitetura conta com os seguintes módulos principais: 1) O algoritmo SDS-MDBScan; 2) Um motor de tomada de decisão composto por vários conjuntos de regras indicando ações a serem selecionadas em função do cenário corrente de jogo. O processo de desenvolvimento do novo agente está descrito na Seção 7.1.

Em síntese, em tal arquitetura, os significados identificados pelo SDS-MDBScan para as mudanças detectadas sobre o comportamento do adversário são usados pelo motor de tomada de decisão, como informação adicional a ser considerada no momento de selecionar um conjunto de ações a serem executadas.

Com isso, será possível que o módulo de decisão do StarCraft funcione em conformidade com o que foi identificado como sendo a semântica de comportamento do adversário. O SDS-MDBScan foi aplicado, pois conforme apresentado nos experimentos da etapa 3 (Seção 6.4), ele apresentou melhores resultados que o Semantic-MDBScan. Além disso, foi utilizado diferentes conjuntos de atributos com o SDS-MDBScan aplicado ao StarCraft, sendo que este procedimento foi apresentado na etapa 2 (capítulo 5), mas sem ser usado, até então, associado ao processo de tomada de decisão de um agente.

O agente SDS-StarCraft foi avaliado por meio de partidas contra diferentes adversários

digitais, visando avaliar os ganhos que cada inovação, presente neste agente, possa trazer. Os experimentos realizados estão descritos na Seção 7.2.

7.1 Desenvolvimento da etapa 4

O principal objetivo desta etapa é melhorar o desempenho de um agente jogador de StarCraft por meio de informações adicionais de mudança de comportamento do jogador oponente.

Para tanto, implementa-se aqui um novo agente jogador cuja arquitetura incorpora o algoritmo SDS-MDBScan ao jogador automático de StarCraft, chamado UAlbertaBot (Seção 2.1). No caso, tal algoritmo tem como objetivo fornecer ao agente informações semânticas sobre o comportamento do seu adversário. A partir de tais informações, o agente poderá reagir em conformidade com aquilo que é observável no cenário de jogo, sendo tal cenário descrito por meio de atributos que permitem uma interpretação sobre as tendências das ações de seu oponente.

Assim sendo, a arquitetura aqui proposta conta com dois módulos de tomada de decisão: um primeiro, denominado *Módulo de execução de regras padrões do StarCraft*, composto pelas regras originais disponibilizadas pela BWAPI. Este módulo será responsável pela escolha de ações quando nenhuma mudança de comportamento no adversário tiver sido identificada ou ocorrer uma mudança de comportamento cuja semântica é desconhecida; e um segundo módulo, denominado *Módulo de execução de regras contextualizadas ao comportamento*, proposto no presente trabalho, e que será responsável pela tomada de decisão nas demais situações.

A Figura 35 apresenta, em uma visão geral, o fluxo que ilustra como o SDS-MDBScan foi incorporado ao motor do jogo usado pelo agente jogador do StarCraft.

Ao longo de uma partida do StarCraft, para cada amostra de dado obtida, o seguinte fluxo será executado: o *Módulo Extrator de Features* será ativado (seta #1) com o objetivo de gerar a amostra de dado que sumariza o cenário corrente de jogo, sendo tal amostra apresentada como entrada para o módulo SDS-MDBScan (seta #2). O módulo SDS-MDBScan funcionará de acordo com o que foi apresentado na Seção 6.3, sendo que a saída deste módulo (seta #3) indicará se houve ou não uma mudança de comportamento. No caso em que for indicada uma mudança de comportamento (seta #4), o sistema identificará o significado (semântica) do novo comportamento. Neste caso, a semântica será usada pelo *Módulo de execução de regras contextualizadas ao comportamento* para selecionar, dentre os conjuntos de regras disponíveis no motor de tomada de decisão proposto neste trabalho, aquele que corresponde à melhor sequência de ações a ser executada na situação corrente de jogo, levando em consideração o significado do novo comportamento. Contudo, caso nenhuma mudança de comportamento tenha sido detectada (seta #5) ou, então, tiver sido detectada uma mudança de comportamento, mas a sua semântica não pertence ao conjunto dos comportamentos (significados) conhecidos, o *Módulo de execução de regras padrões do StarCraft* será acionado. Em ambas as situações, ou seja, tanto no caso de o conjunto de ações escolhido tiver sido definido pelo *Módulo de execução de regras contextualizadas ao comportamento* (seta #6) ou pelo *Módulo de execução de regras*

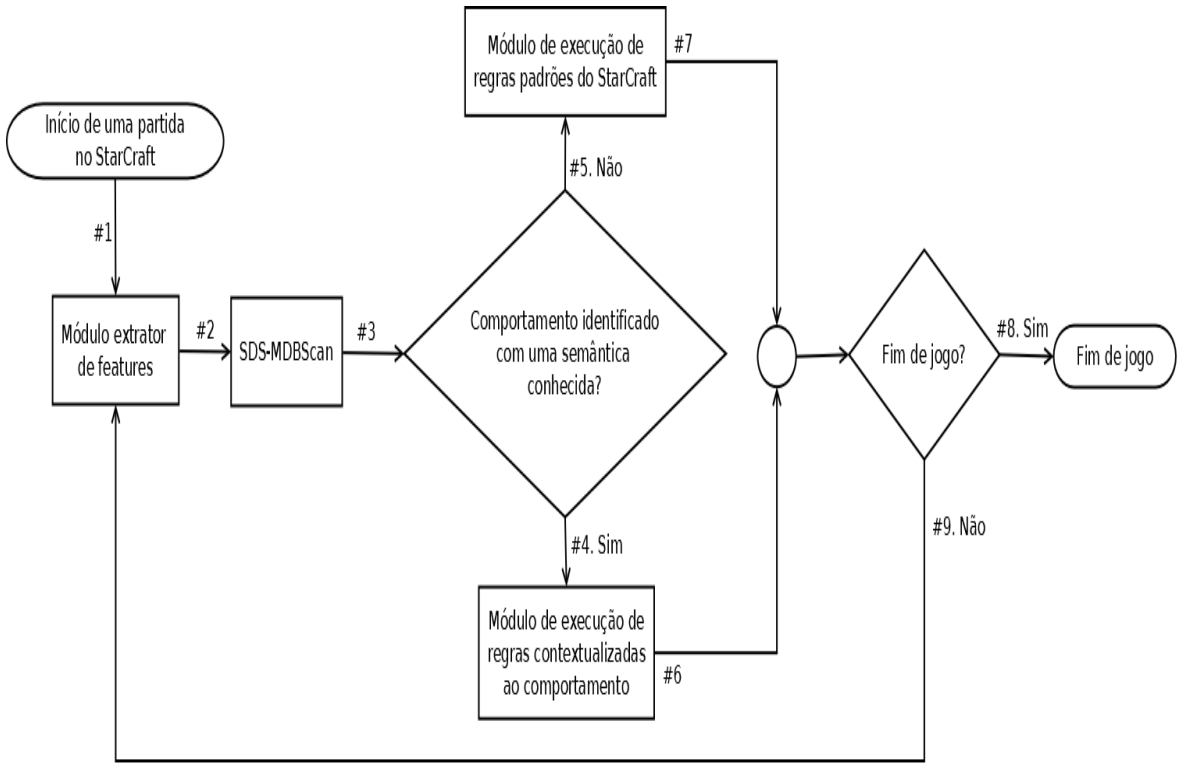


Figura 35 – Fluxograma do StarCraft com SDS-MDBScan.

padrões do StarCraft (seta #7), o sistema verifica se chegou no fim da partida. Caso afirmativo (seta #8), o jogo é encerrado, senão (seta #9), o fluxo se reinicia com o acionamento do *Módulo Extrator de Features* para gerar uma nova amostra de dado que representa o novo cenário corrente de jogo.

Para incorporar o SDS-MDBScan ao agente UAlbertaBot como foi apresentado na Figura 35 é preciso definir os seguintes elementos: as *features* utilizadas para representar o cenário de jogo; os comportamentos do adversário a serem considerados; os novos conjuntos de ações a serem executadas em função de cada comportamento detectado (correspondem às acões do *Módulo de execução de regras contextualizadas ao comportamento*); e, por fim, onde, no código-fonte do motor do jogo estas alterações deverão ser implantadas.

As próximas seções apresentam em maiores detalhes tal processo de incorporação do SDS-MDBScan ao StarCraft.

7.1.1 Representação do cenário de jogo por meio de atributos

As amostras de dados usadas para representar os cenários do jogo foram obtidas de partidas do StarCraft. Estas amostras de dados são representadas por 8 atributos relevantes, selecionados empiricamente, e obtidas da seguinte maneira: o valor dos 8 atributos são continuamente extraídos ao longo de toda partida a cada intervalo de 15 frames, ou seja, esses 8 atributos vão sumarizar os dados referentes a este intervalo. Os valores de cada atributo são zerados antes de começar uma nova leitura de dados referente aos 15 frames seguintes a última leitura.

Os cenários de jogo apresentam mudanças do tipo gradual/recorrente devido as características do jogo StarCraft, onde as ações não são concluídas imediatamente, como a construção de novas estruturas, a movimentação de unidades pelo mapa, ou mesmo, o combate entre unidades inimigas.

Os 8 atributos usados para representar o cenário de jogo são os seguintes:

- *Power Damage of agent's army* (A_1): Indica o poder de ataque do exército do agente;
- *Power Damage of enemy's army* (A_2): Indica o poder de ataque das unidades do exército do adversário que foram vistas por alguma unidade do agente;
- *Last agent's deaths* (A_3): Indica o número de mortes das unidades do exército do agente;
- *Agent's units In enemy's base* (A_4): Indica o número de unidades do agente em bases do adversário;
- *Enemies in agent's base* (A_5): Indica o número de unidades do adversário que foram vistas por unidades do agente nas bases do agente;
- *Previous agent's units in enemy's base* (A_6): Indica o número prévio de unidades do agente em bases do adversário. Corresponde ao valor anterior do atributo A_4;
- *Previous enemies in agent's base* (A_7): Indica o número prévio de unidades do adversário que foram vistas por unidades do agente nas bases do agente. Corresponde ao valor anterior do atributo A_5;
- *Enemy units in support activities* (A_8): Indica o número de unidades inimigas vistas por unidades do agente realizando atividades como coleta de minerais ou construção de estruturas.

Os atributos usados na etapa 2 (capítulo 5) não foram usados na incorporação do SDS-MDBScan no StraCraft, pois nesta etapa 2 foram concebidos atributos, quando ainda não havia a necessidade de representar comportamentos. Assim, nesta etapa 4, para se trabalhar com comportamentos que sejam bem distintos entre si, permitindo definir reações do agente diante da identificação de cada comportamento, novos atributos foram necessários. O processo de definição dos comportamentos está descrito a seguir, na Seção 7.1.2.

7.1.2 Definindo um conjunto de comportamentos

O conjunto de comportamentos do adversário foi inicialmente definido de forma empírica, com base em características inerentes ao StarCraft (representadas por meio de atributos específicos do cenário de jogo) e, também, por meio de um guia de estratégias para jogadores do modo *multiplayer* (FARKAS, 2002). Neste guia, há uma série de orientações para jogadores, que se baseiam em vários aspectos, tais como as particularidades das raças usadas na partida como, por

exemplo, os fatores que as tornam mais fracas ou mais fortes, além dos objetivos específicos no jogo como, por exemplo, a conquista de uma região do mapa. Como esta etapa da pesquisa tem o objetivo de criar um jogador automático mais genérico, o conjunto de comportamentos foi definido de forma a representar as característica gerais das raças, evitando, assim, concentrar-se em aspectos particulares de uma raça.

Dessa forma, as informações presentes no referido guia foram muito úteis para definir comportamentos distintos e representativos da natureza do jogo. Os rótulos dos comportamentos foram uma criação do autor, não estando presente neste guia.

Cada comportamento precisa ser associado a um dado conjunto de ações, de modo que o agente esteja apto a decidir como agir em situações em que um comportamento for identificado. Dessa maneira, os comportamentos inicialmente definidos foram os seguintes:

- ❏ Agressivo defensivo: Este comportamento é caracterizado pelo ataque do adversário às unidades do agente, e a localização desset confronto é a base do adversário.

- ❏ Agressivo ofensivo: Este comportamento é caracterizado pelo ataque do adversário às unidades do agente, e este ataque ocorre na base do agente.

- ❏ Defensivo imediato: Este comportamento é caracterizado pelo recuo feito pelo adversário, retirando as suas unidades da região onde ocorre um confronto.

- ❏ Defensivo estratégico: Este comportamento é caracterizado pela concentração do adversário na realização de tarefas dentro de sua base, como a construção de novas estruturas e a coleta de materiais.

A rotulação desse conjunto de comportamentos ocorre por meio de uma árvore de decisão cujos ramos são definidos por meio dos intervalos de valores que os atributos (apresentados na Seção 7.1.1) assumem em função dos comportamentos que representam, conforme ilustrado na Figura 36. Tais intervalos foram empiricamente estabelecidos em função das características do jogo. O rótulo *Unknown* representa situações que não apontam para nenhum dos comportamentos considerados. Dessa maneira, a árvore de decisão foi criada para funcionar como um especialista capaz de definir um comportamento com base nos valores dos atributos utilizados. A árvore é usada, por exemplo, na geração das bases de amostras rotuladas (*Training Datasets*) que serão utilizadas pelos algoritmos de classificação (KNN e MLP), que integram o módulo de atribuição de semântica do comportamento, tanto presente no Semantic-MDBScan quanto no SDS-MDBScan.

Na seção a seguir, está descrito o processo de criação das ações contextualizadas, as quais o agente deverá executar caso um certo comportamento seja identificado.

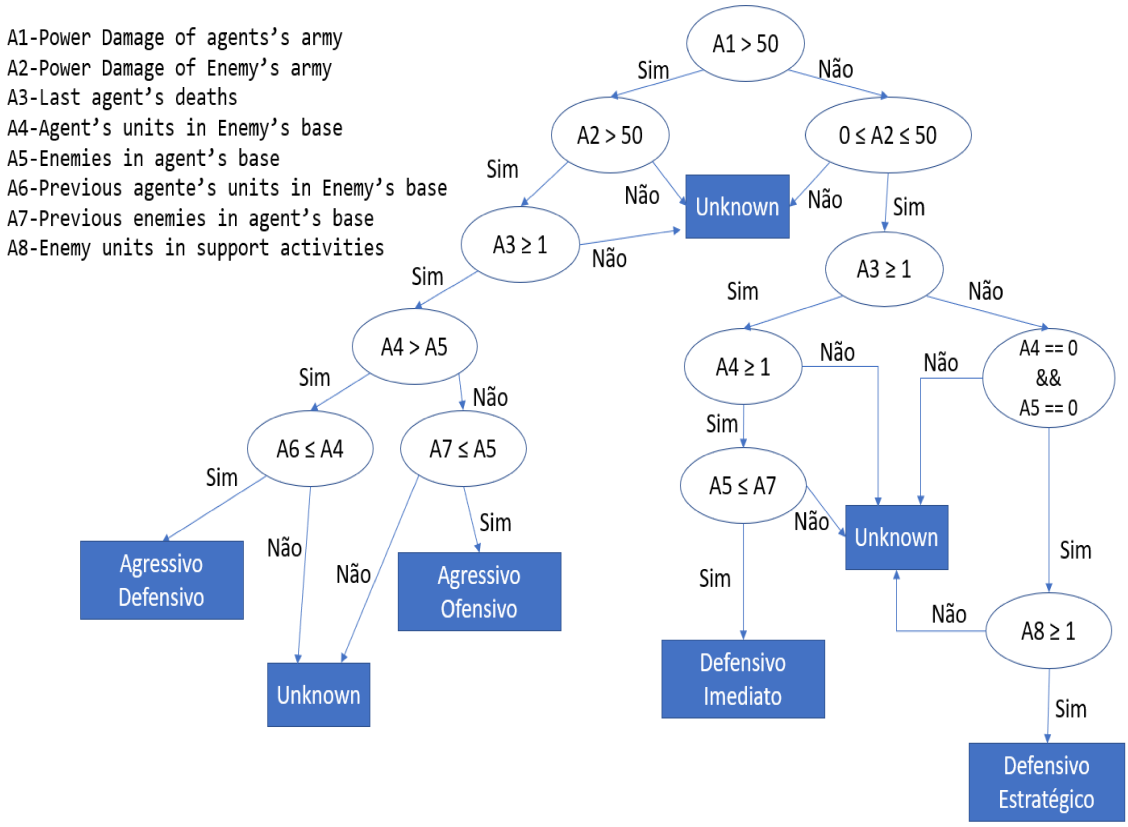


Figura 36 – Árvore de decisão para o conjunto inicial de 4 comportamentos.

Fonte: Elaborada pelo autor.

7.1.3 Ações contextualizadas a cada comportamento

Considerando todos os comportamentos apresentados na Seção 7.1.2, foi preciso definir as ações contextualizadas que devem ser executadas quando determinada semântica vinculada a uma mudança for identificada. Estas ações são apresentadas na Tabela 17. É importante salientar que cada uma dessas reações corresponde ao conjunto de ações contextualizadas que devem ser executadas, quando mudanças são identificadas sobre o comportamento do adversário.

7.1.4 Alterações aplicadas ao agente UAlbertaBot do StarCraft

A fim de acoplar adequadamente o SDS-MDBScan ao UAlbertaBot foi preciso identificar, no código do motor, onde seria possível obter amostras de dados que representassem a dinâmica de variação do cenário de jogo. Também foi necessário identificar o local no código, onde deveriam ser implementadas as ações contextualizadas a cada comportamento, ou seja, aquelas que deverão ser executadas pelo agente cada vez que for detectada uma mudança de comportamento do oponente.

Tal análise permitiu concluir que as classes do código do jogo de onde as amostras, que refletem o cenário de jogo, poderiam ser recuperadas, e onde as ações contextualizadas a cada

Tabela 17 – Ações contextualizadas para cada comportamento

Comportamento Identificado	Reação do agente
Agressivo Defensivo	As unidades mais próximas da base do adversário serão movidas para defesa das unidades sob ataque.
Agressivo Ofensivo	As unidades fora da base do agente retornarão para a base, para defender as unidades sob ataque.
Defensivo Imediato	As unidades do agente deverão seguir as unidades do adversário, mantendo a ofensiva do ataque.
Defensivo Estratégico	As unidades do agente deverão atacar as unidades do adversário concentradas em tarefas como extração de recursos ou construção de novas estruturas.

comportamento poderiam ser incorporadas, são as seguintes:

- A classe *GameCommander*, onde ocorre a inicialização de vários elementos do jogo que existirão ao longo de toda a partida como os gerenciadores de combate, mapa, construções, entre outros. Assim, a classe *GameCommander* se torna apropriada para obter informações do contexto do jogo desde o início da partida até a sua conclusão.

- A classe *CombatCommander*, que está vinculada à dinâmica dos confrontos existentes no jogo, o que é essencial para gerar eventos que funcionarão como reações frente ao comportamento identificado. Na classe *CombatCommander* é possível fazer alterações nos esquadrões que estão realizando algum ataque, alguma defesa, ou mesmo atribuir uma função para unidades que estão inativas. Logo, a classe *CombatCommander* se torna apropriada para implementar o conjunto de ações contextualizadas de cada comportamento.

É importante ressaltar que ambas as classes citadas acima são implementadas originalmente no próprio agente de base, o UAlbertaBot.

O uso das classes *GameCommander* e *CombatCommander* com o objetivo de incorporar o SDS-MDBScan ao jogo StarCraft, são apresentados no fluxograma da Figura 37. Este fluxograma é uma visão focada sobre a ordem em que tais classes interagem com o SDS-MDBScan. Este fluxograma não tem o objetivo de apresentar o acionamento da globalidade das classes do agente UAlbertaBot.

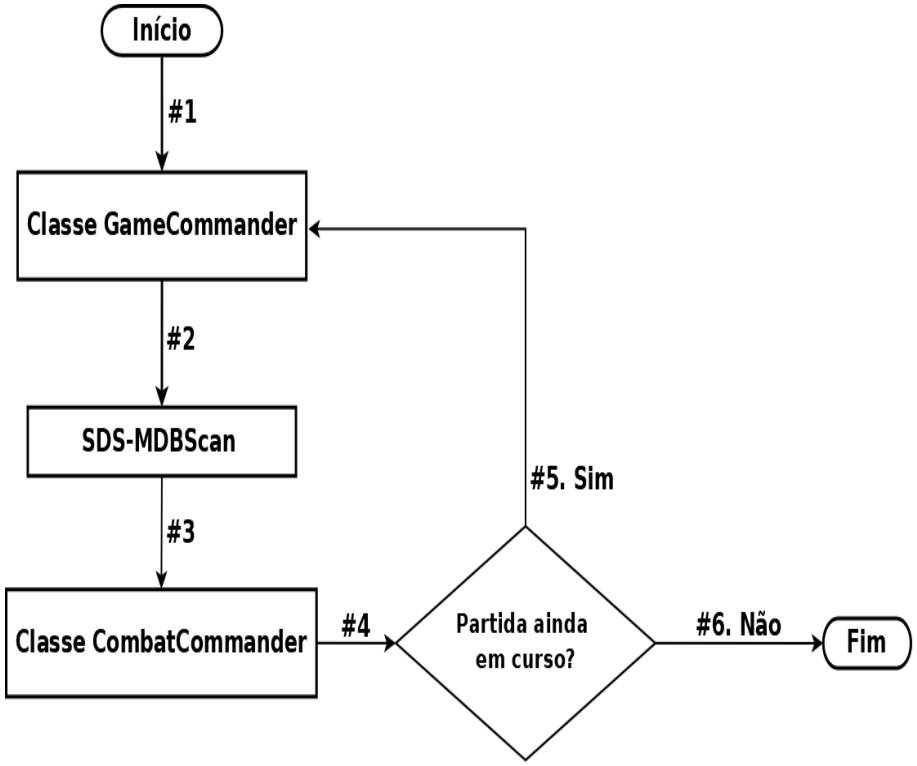


Figura 37 – Combinando o SDS-MDBScan ao motor StarCraft por meio das classe GameCommander e CombatCommander.

A transição #1 do fluxograma representa a busca por uma nova amostra de dados a ser produzida na classe *GameCommander*. Esta amostra de dados representa o cenário de jogo e servirá como entrada para o SDS-MDBScan (seta #2), que poderá indicar uma mudança de comportamento ou não, mas em ambas as situações o próximo módulo acionado é representado pela classe *CombatCommander* (seta #3). Se uma mudança de comportamento não foi identificada, o *CombatCommander* executará as ações padrões já existentes no motor do jogo, mas caso uma mudança de comportamento foi identificada, serão executados comandos que corresponderam as ações contextualizadas do agente (detalhado na Seção 7.1.3). Será feita, então, uma verificação se a partida ainda está em curso (seta #4), se sim (seta #5), então o processo se repete, buscando uma nova amostra de dado que represente o cenário de jogo. Caso a partida está encerrada (seta #6), o uso do SDS-MDBScan também se encerra.

Na seção seguinte será descrita a integração do que foi desenvolvido na etapa 2 (capítulo 5), que envolve o uso de conjuntos distintos de atributos para diferentes momentos do jogo, com o SDS-MDBScan integrado no jogo StarCraft.

7.1.5 Uso de diferentes conjuntos de atributos com o SDS-MDBScan no StarCraft

Conforme apresentado na etapa 2, foram usados diferentes conjuntos de atributos, que melhor descrevem diferentes estágios do jogo, como uma forma de aumentar a acurácia na de-

tecção de mudanças de comportamento. Então, para realizar esta combinação dos diferentes conjuntos de atributos, propostos na etapa 2, com a integração do SDS-MDBScan ao Starcraft, será preciso definir quais atributos utilizados na etapa 4 representarão os diferentes estágios do jogo. Assim, a definição dos atributos em função dos estágios foi a seguinte:

- *Estágio de formação da batalha* (EFB) será representado pelos seguintes atributos:
  - Power Damage of agent's army (A_1);
  - Power Damage of enemy's army (A_2);
  - Agent's units In enemy's base (A_4);
  - Enemies in agent's base (A_5).
- *Estágio intermediário da batalha* (EIB) e o *Estágio de conclusão da batalha* (ECB) serão ambos representados pelo conjunto completo de atributos, lembrando que o conjunto de atributos usados na etapa 4 é menor do que o originalmente usado na etapa 2. Essa redução no conjunto de atributos visa garantir uma plena distinção entre os comportamentos usados para representar o adversário. Então, os atributos dos estágios EIB e ECB são os seguintes:
  - Power Damage of agent's army (A_1);
  - Power Damage of enemy's army (A_2);
  - Last agent's deaths (A_3);
  - Agent's units In enemy's base (A_4);
  - Enemies in agent's base (A_5);
  - Previous agent's units in enemy's base (A_6);
  - Previous enemies in agent's base (A_7);
  - Enemy units in support activities (A_8).

Como os estágios EIB e ECB utilizaram o mesmo conjunto de atributos, não haverá diferença entre as amostras de dados nestes estágios, isso permite usar uma única CM para os estágios EIB e ECB, visto que não comprometerá o processo de atribuição de uma amostra a um *micro-cluster*, que ocorre com base na distância Euclidiana. Então, será possível usar duas CMs, uma para o EFB e uma outra para EIB e ECB.

Os detalhes de todos os experimentos realizados na etapa 4 estão descritos na Seção 7.2.

7.2 Experimentos da etapa 4

Os experimentos descritos nesta seção visam avaliar a incorporação do SDS-MDBScan ao jogo StarCraft, mensurando o desempenho de tal agente. Para tanto, implementa-se o novo sistema jogador apresentado na Figura 37, referenciado, a partir daqui, como SDS-StarCraft.

Assim sendo, os experimentos serão conduzidos de forma a avaliar o quão a identificação do significado de uma modificação comportamental do oponente contribui para melhorar a qualidade da tomada de decisão do SDS-StarCraft com relação ao agente StarCraft padrão, que é o agente jogador UAlbertaBot apresentado na Seção 2.1.

Para tanto, dois cenários de teste serão executados: o primeiro (Seção 7.2.2) tem como objetivo avaliar a eficácia do conjunto de semânticas de comportamento inicialmente definido na Seção 7.1.2 e, a partir dessa análise, verificar se haveria alternativas mais adequadas para tal conjunto; o segundo cenário (Seção 7.2.3) visa avaliar o impacto que a abordagem aqui proposta propicia a um sistema jogador em termos de desempenho em partidas disputadas, efetuando tal avaliação a partir dos conjuntos de comportamentos investigados. Os parâmetros usados nos experimentos estão apresentados na Seção 7.2.1.

7.2.1 Configurações dos experimentos do SDS-MDBScan no StarCraft

Os valores dos parâmetros do processo de agrupamento e da entropia espacial usados com o algoritmo SDS-MDBScan foram definidos com base nos experimentos apresentados na Seção 6.4.3. Lembrando que nestes experimentos só há mudanças do tipo gradual/recorrente, e só foi usada entropia espacial, por ter apresentado melhores resultados nos experimentos anteriores. Os parâmetros são os seguintes:

- Parâmetros da etapa de agrupamento: $\mu = 10$; $\beta = 0{,}105$; $\varepsilon = 0{,}45$; $\lambda = 0{,}001$, sendo que μ é quantidade mínima de pontos para ser considerado um *micro-cluster*; ε é o limite de raio para o *micro-cluster*; β é o limiar de outlier; e λ é o fator de decaimento.

- Parâmetros da entropia espacial: quantidade mínima de novidades para uma mudança igual a 4; tamanho da janela deslizante igual a 20. Outros parâmetros: $\eta_s = 0{,}05$; $\gamma = 0{,}5$; $\delta = 0{,}02$; $\theta = 2$, senso que η_t é um peso usado para controlar a intensidade da atualização de cada probabilidade; γ e δ são os pesos usados no cálculo do limiar da entropia; e θ é uma constante que corresponde ao número de desvio padrões usados para definir uma distribuição normal para os valores de entropia.

Para o KNN, o valor de K foi definido como 5, devido aos mesmos experimentos apresentados na Seção 6.4.3.

A *TrainingDataset* usada nos experimentos foi construída seguindo a descrição apresentada na Seção 6.4.5.2.

7.2.2 Investigando alternativas de conjunto de comportamentos

O primeiro desafio ao incorporar o SDS-MDBScan ao StarCraft é definir um conjunto de comportamentos que realmente represente o estilo de jogo do adversário, provendo informações que sejam úteis para nortear a reação do agente diante de tais comportamentos.

Para avaliar a relevância do conjunto de comportamentos apresentados na Seção 7.1.2, foi definida a seguinte avaliação: Será contabilizada a quantidade de vezes que cada comportamento foi acionado em uma partida (ativando o *Módulo de execução de regras contextualizadas ao comportamento*), e também a quantidade de vezes que alguma ação padrão do StarCraft foi acionada (ativando o *Módulo de execução de regras padrão do StarCraft*). Este último caso pode ocorrer quando nenhuma mudança de comportamento foi detectada, ou, quando, alguma mudança de comportamento foi detectada, contudo, seu significado não se encaixa em algum dos comportamentos definidos, representando a situação de comportamento *Unknown*. Então, em função de um total de ações ativadas em uma partida, é possível analisar a representatividade de cada comportamento, considerando o número total de ações ativadas. Para esta análise de relevância de comportamentos, foram usadas 10 partidas em que o agente foi vitorioso e outras 10 partidas em que o agente foi derrotado.

A Figura 38 apresenta a média dos resultados para o grupo de partidas em que o SDS-StarCraft foi vitorioso, enquanto que a Figura 39 apresenta a média dos resultados para o grupo de partidas em que tal agente foi derrotado. Em ambos os casos, o SDS-StarCraft operou com o conjunto de 4 comportamentos inicialmente definido.

Analisando as Figuras 38 e 39 é observável que a média do percentual de acionamento das ações padrões do StarCraft, vinculadas ao *Módulo de execução de regras padrões do StarCraft* é maior entre as partidas em que o agente SDS-StarCraft foi derrotado do que entre as partidas em que o agente venceu, 68% e 58% respectivamente.

Logo, constatou-se que o desempenho do agente melhora a medida em que a quantidade de decisões tomadas pelo *Módulo de execução de regras contextualizadas ao comportamento*, presente no SDS-StarCraft, aumenta em relação à quantidade das ações tomadas pelo *Módulo de execução de regras padrão do StarCraft*, que também está presente no SDS-StarCraft e foi herdado do agente que serviu de base para o seu desenvolvimento, o UAlbertaBot (apresentado na Seção 2.1).

Tal resultado indica que uma boa estratégia para melhorar o desempenho do agente jogador, envolve a inserção de novas semânticas ao conjunto de comportamentos inicialmente proposto (com 4 comportamentos). Essa conclusão provém do seguinte fato: tal inserção provocaria uma redução no número de rótulos *Unknown* da árvore de decisão (Figura 36), reduzindo também, por consequência, o número de decisões a serem tomadas pelo *Módulo de execução de regras padrão do StarCraft*.

Contudo, para ampliar o conjunto de comportamentos deve-se continuar definindo somente comportamentos com características bem distintas entre si, e por isso, algum conjunto de ações no jogo pode não ser representado. Essa medida é importante para garantir a distinção entre os comportamentos, reduzindo a chance de uma reação incorreta do agente, se o comportamento identificado for muito parecido com algum outro, e devido a esta semelhança, um conjunto incorreto de ações pode ser acionado.

Além disso, para não aumentar a quantidade de comportamentos de maneira caótica, in-

Figura 38 – Percentual de ativação de ações de comportamentos em partidas com vitória - Uso de 4 comportamentos.

corporando e criando novos comportamentos que não estariam perfeitamente representados nos dados, com o simples propósito de aumentar a ativação de ações contextualizadas. Foi feita uma análise sobre dados de uma partida, buscando identificar a quantidade ideal de comportamentos.

Nesta análise, um algoritmo de agrupamento foi usado, pois amostras de dados vinculadas a um mesmo comportamento apresentam semelhanças, que em um processo de agrupamento resultaria nestas amostras pertencerem a um mesmo grupo. Logo, cria-se uma relação direta da quantidade de grupos com a quantidade de comportamentos.

Dessa maneira, amostras de dados de uma partida (construídas como explicado na Seção 7.1.1) foram submetidas ao algoritmo de agrupamento K-means (MACQUEEN, 1967), que é um algoritmo popular na literatura e que atende a necessidade desta análise, que é explorar diferentes quantidades de grupos buscando um valor ideal. Contudo, ainda é preciso definir uma forma de mensurar a qualidade do agrupamento, para saber se ela é melhor ou pior que a

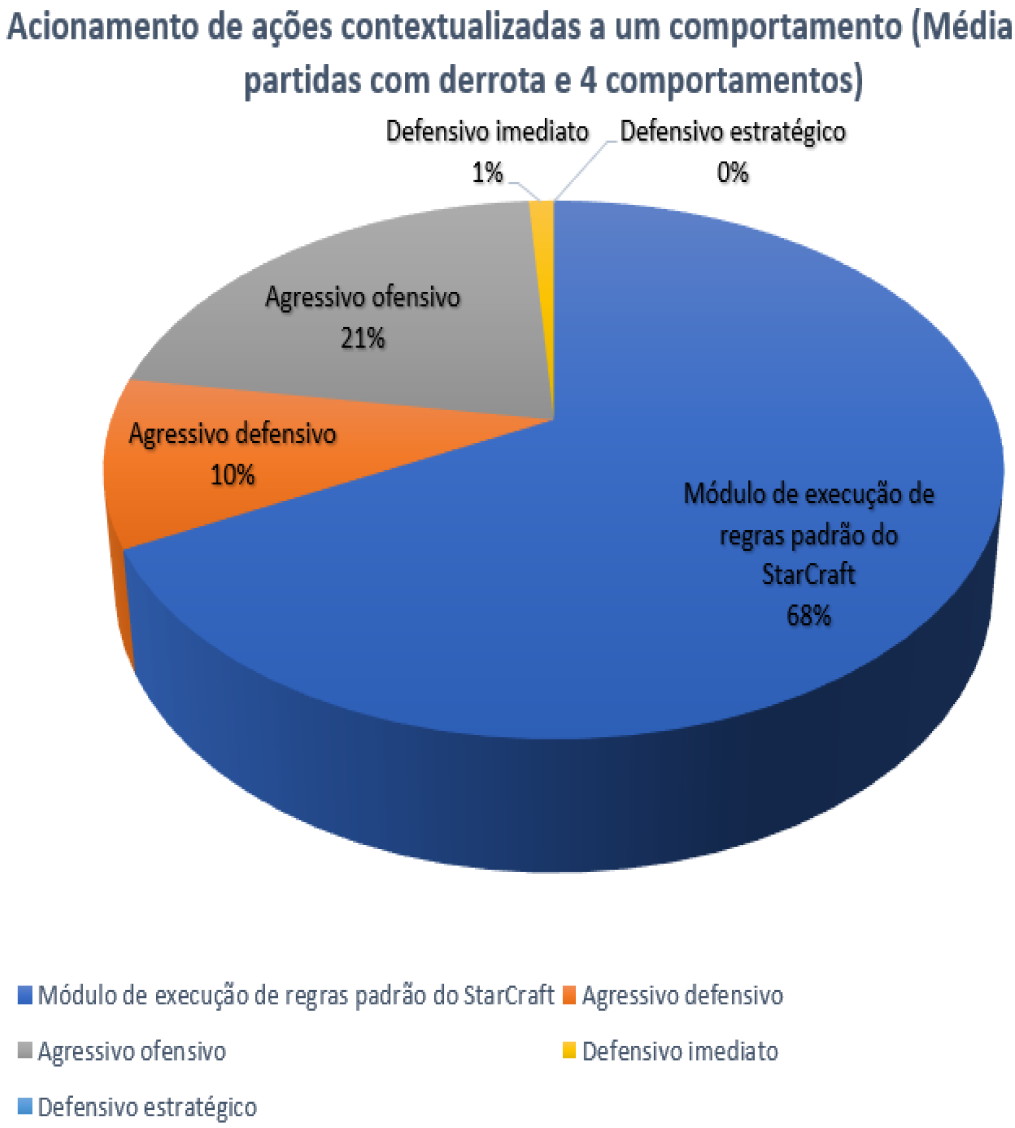


Figura 39 – Percentual de ativação de ações de comportamentos em partidas com derrota - Uso de 4 comportamentos.

já utilizada. Para isto, foi usada a medida de silhueta, que demonstra a coesão entre as amostras do mesmo grupo, comparado aos outros grupos.

No resultado da análise foi encontrado como melhor resultado um total de seis grupos. Considerando a existência de uma relação direta da quantidade de comportamentos e grupos, foram definidos mais dois comportamentos, também rotulados por meio de uma árvore de decisão ilustrada na Figura 40.

Os dois novos comportamentos são os seguintes:

- ❏ Ofensivo extremo: Este comportamento é caracterizado pelo ataque imediato do oponente devido ao fato de ele não dispor de um grande exército para sua defesa.
- ❏ Defensivo extremo: Este comportamento é caracterizado por um recuo das unidades do adversário em situações em que as unidades do agente estão quase morrendo.

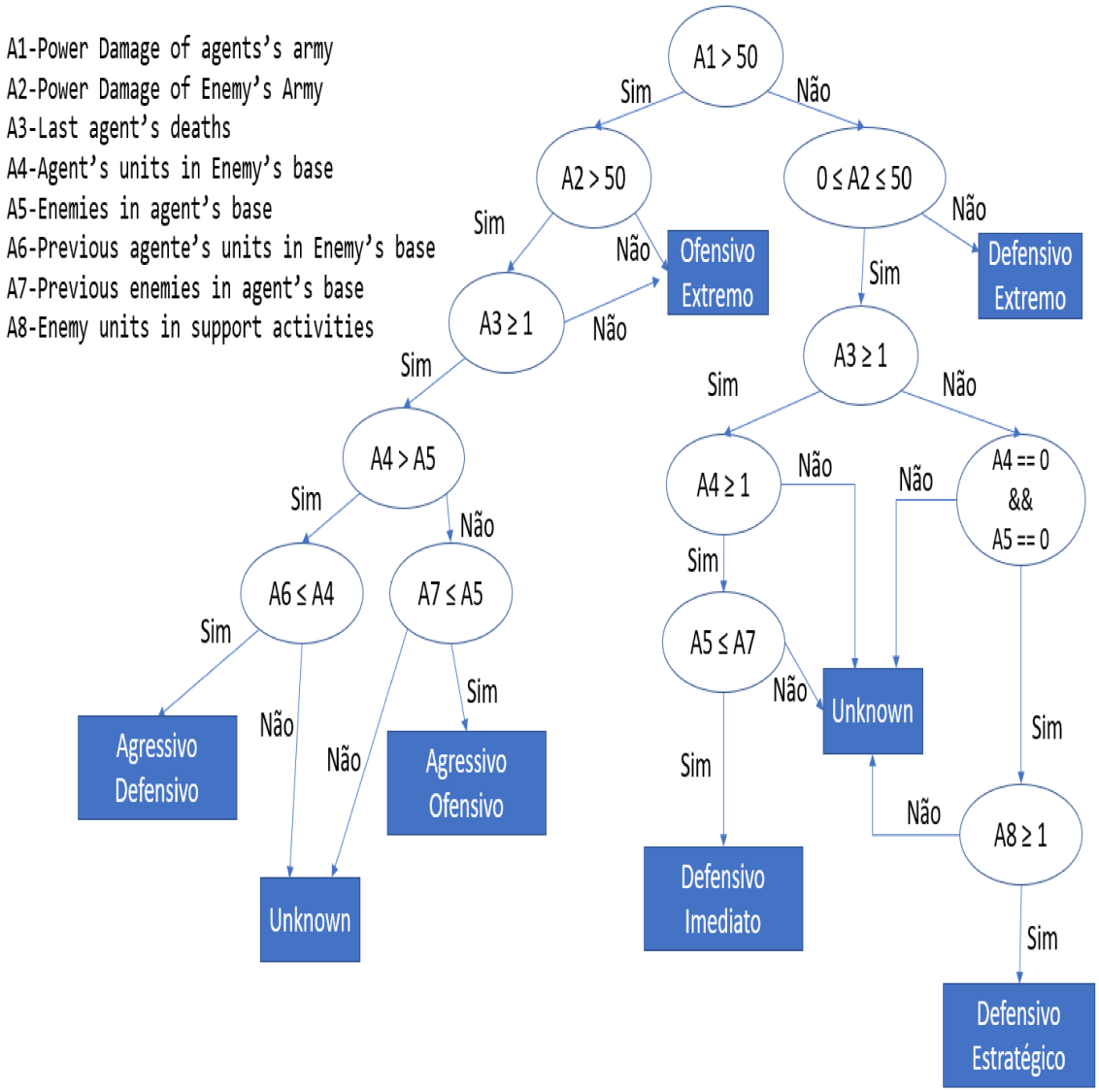


Figura 40 – Árvore de decisão com classificação de 6 comportamentos.

Fonte: Elaborada pelo autor.

As reações que o agente SDS-StarCraft deve ter, caso algum dos novos comportamentos for identificado, estão apresentadas na Tabela 18.

Conforme mencionado anteriormente, o aumento no número de comportamentos tem como consequência natural a redução de rótulos "*Unknown*" na árvore de decisão apresentada na Figura 36, construída a partir de 4 comportamentos. Essa redução é observável na nova árvore de decisão, apresentada na Figura 40, construída com base em 6 comportamentos.

Repetindo a análise de relevância de comportamento, mas agora com seis comportamentos. Nesta análise, também foram usadas 10 partidas com vitórias e 10 partidas com derrotas do agente SDS-StarCraft. Analisando os resultados, observa-se que a média do percentual de ativação de alguma ação padrão do motor do StarCraft, nas partidas com vitórias, caiu de 58% para 49%, enquanto que na média, entre as partidas com derrota, esse valor caiu de 68% para

Tabela 18 – Ações contextualizadas para os novos comportamentos

Comportamento Identificado	Reação do agente
Ofensivo Extremo	As unidades do agente deverão recuar para se protegerem, fugindo do confronto.
Defensivo Extremo	As unidades com maior nível de vida deverão perseguir e manter o ataque contra o adversário.

64%. Como esta análise ocorre sobre o conjunto total de ações, formada por ações padrão do StarCraft e ações contextualizadas a um comportamento, essa queda indica que foi possível, com a alteração para seis comportamentos, aumentar o acionamento das ações contextualizadas aos comportamentos, durante as partidas do agente SDS-StarCraft, como pode ser observado nos resultados apresentados nas Figuras 41 e 42.

Na próxima seção são apresentados os resultados dos experimentos, onde o agente SDS-StarCraft joga com diferentes adversários digitais.

7.2.3 Avaliando o agente SDS-StarCraft

Este segundo cenário de teste tem o objetivo de avaliar o desempenho do SDS-StarCraft por meio de resultados obtidos em partidas realizadas em tempo real no próprio ambiente do jogo. Tal avaliação será feita por meio de duas análises distintas: inicialmente, será avaliada a taxa de vitórias; na segunda análise, o desempenho será medido por meio da pontuação *kill score*, que reflete o desempenho do agente quanto a quantidade de mortes que ele provocou no exército do adversário.

A primeira análise será conduzida em três etapas: a primeira (Seção 7.2.3.1) avalia a taxa de vitória em partidas que têm como competidores o SDS-StarCraft (4 comportamentos) e o UAlbertaBot (agente StarCraft sem SDS-MDBScan); a segunda etapa (Seção 7.2.3.2) avalia a taxa de vitória em partidas em que o SDS-StarCraft (6 comportamentos) enfrenta o SDS-StarCraft (4 comportamentos); a terceira etapa (Seção 7.2.3.3) avalia o SDS-StarCraft (6 comportamentos) associado ao que foi feito na etapa 2 da pesquisa, que é utilizar conjuntos de atributos adaptáveis ao estágio do jogo. Então, o SDS-StarCraft (6 comportamentos com atributos adaptáveis) enfrentará o SDS-StarCraft (4 comportamentos). Foi feita esta escolha de adversário, pois na segunda etapa, o SDS-StarCraft (6 comportamentos) enfrentou o SDS-StarCraft (4 comportamentos), logo será possível uma comparação direta, identificando os possíveis ganhos do agente quando ele utiliza atributos adaptáveis ao estágio da partida.

A segunda análise (Seção 7.2.3.4), como já mencionado, será feita pela avaliação do *kill score* obtido por cada agente em partidas que envolvem o UAlbertaBot, o SDS-StarCraft (4 comportamentos) e o SDS-StarCraft (6 comportamentos) com ou sem atributos adaptáveis.

Figura 41 – Percentual de ativação das ações de comportamentos em partidas com vitória - Uso de 6 comportamentos.

7.2.3.1 Experimentos com o conjunto de 4 comportamentos

Para analisar o desempenho do agente SDS-StarCraft, utilizando o conjunto inicial de quatro comportamentos, foram executadas quarenta partidas onde o SDS-StarCraft enfrentou o agente jogador UAlbertaBot, que não usa SDS-MDBScan.

No gráfico apresentado na Figura 43 observa-se que o SDS-StarCraft (4 comportamentos) venceu 77% das partidas disputadas, demonstrando que o uso da semântica para nortear as tomadas de decisão do agente foi um fator que favoreceu muito o agente, visto que a base de desenvolvimento dele foi o seu adversário, o agente UAlbertaBot. Usando o UAlbertaBot como uma base de comparação, o SDS-StarCraft (6 comportamentos), em 40 partidas, obteve uma taxa de vitória de 92%. Por outro lado, o SDS-StarCraft (6 comportamento e atributos adaptáveis), também enfrentando o UAlbertaBot em 40 partidas, obteve uma taxa de vitória de

Figura 42 – Percentual de ativação das ações de comportamentos em partidas com derrota - Uso de 6 comportamentos

97%.

Na seção a seguir serão apresentados os resultados do agente SDS-StarCraft com 6 comportamentos, onde ele enfrenta a sua versão com 4 comportamentos.

7.2.3.2 Experimentos com o conjunto de 6 comportamentos

Para analisar o desempenho do agente SDS-StarCraft utilizando o conjunto estendido de seis comportamentos, também foram executadas quarenta novas partidas onde o agente SDS-StarCraft (6 comportamentos) enfrentou o agente SDS-StarCraft (4 comportamentos).

No gráfico apresentado na Figura 44 observa-se que o agente SDS-StarCraft (6 comportamentos) venceu 87% das partidas disputadas contra o agente SDS-StarCraft (4 comportamentos). Este resultado reforça o que já tinha sido observado nos experimentos apresentados

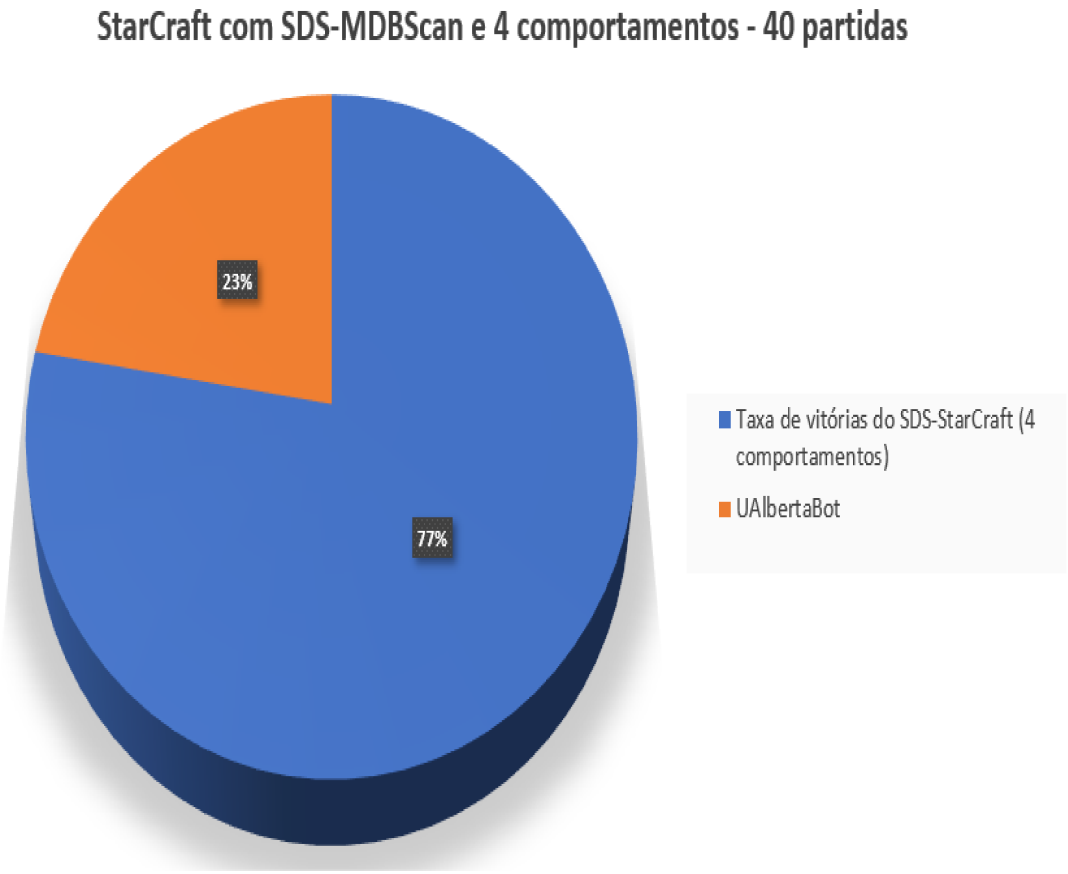


Figura 43 – Percentual de vitórias em partidas do StarCraft: SDS-MDBScan com 4 comportamentos.

na Seção 7.2.2, onde se constatou que as ações contextualizadas realmente permitiram que o agente reagisse de forma adequada ao estilo de jogo do adversário, refletindo nos resultados das partidas.

Na próxima seção são apresentados os resultados obtidos com os experimentos, que visa combinar o que foi desenvolvido na etapa 2 (capítulo 5) com o agente SDS-StarCraft.

7.2.3.3 Experimentos utilizando SDS-MDBScan com atributos para distintos estágios do jogo StarCraft

Para analisar o desempenho do agente SDS-StarCraft com seis comportamentos e utilizando um conjunto variável de atributos, que se adapta em função do estágio do jogo, como foi apresentado na Seção 7.1.5, foram executadas quarenta partidas entre o SDS-StarCraft (6 comportamentos e atributos adaptáveis) e o agente SDS-StarCraft (4 comportamentos), que não utiliza atributos adaptáveis.

No gráfico apresentado na Figura 45 observa-se que o agente SDS-StarCraft (6 comportamentos e atributos adaptáveis) venceu 90% das partidas disputadas contra o agente SDS-StarCraft (4 comportamentos), representando um ganho de 3% na taxa de vitória do agente, se

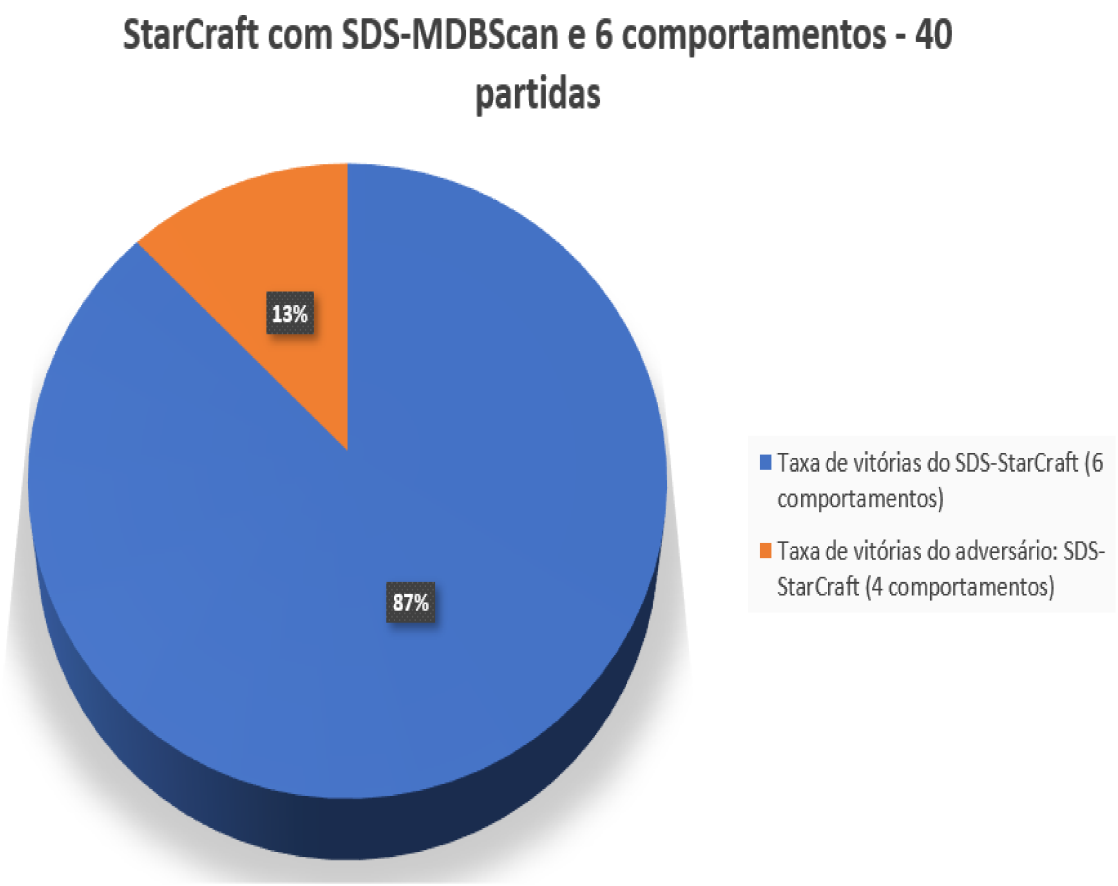


Figura 44 – Percentual de vitórias em partidas do StarCraft: SDS-MDBScan com 6 comportamentos.

comparado ao resultado da sua versão sem atributos adaptáveis, conforme apresentado na Seção 7.2.3.2.

O uso de atributos adaptáveis havia mostrado com os resultados da etapa 2 (Seção 5.2.1), que o uso de uma representação adequada via atributos, aumenta a quantidade de VPs nas detecções de mudanças de comportamento. Logo, aplicar essa abordagem a incorporação do SDS-MDBScan ao StarCraft evita que o motor do jogo cometa o erro de ativar alguma reação do agente que não se encaixa ao contexto corrente do jogo. Este tipo de erro prejudicaria o agente diretamente, pois ele pode, por exemplo, se tornar mais agressivo quando deveria estar mais defensivo, e isto poderia causar perdas na composição do seu exército, que talvez não seja recuperável até o fim de uma partida.

Na seção seguinte será feita uma análise sobre o desempenho do agente SDS-StarCraft quanto a pontuação *kill score* obtida nas partidas realizadas.

7.2.3.4 Analisando a pontuação *kill score*

O jogo StarCraft fornece uma pontuação chamada *kill score*, capaz de indicar o desempenho do jogador quanto as mortes que ele provocou no exército do adversário. Com esta pontuação, é

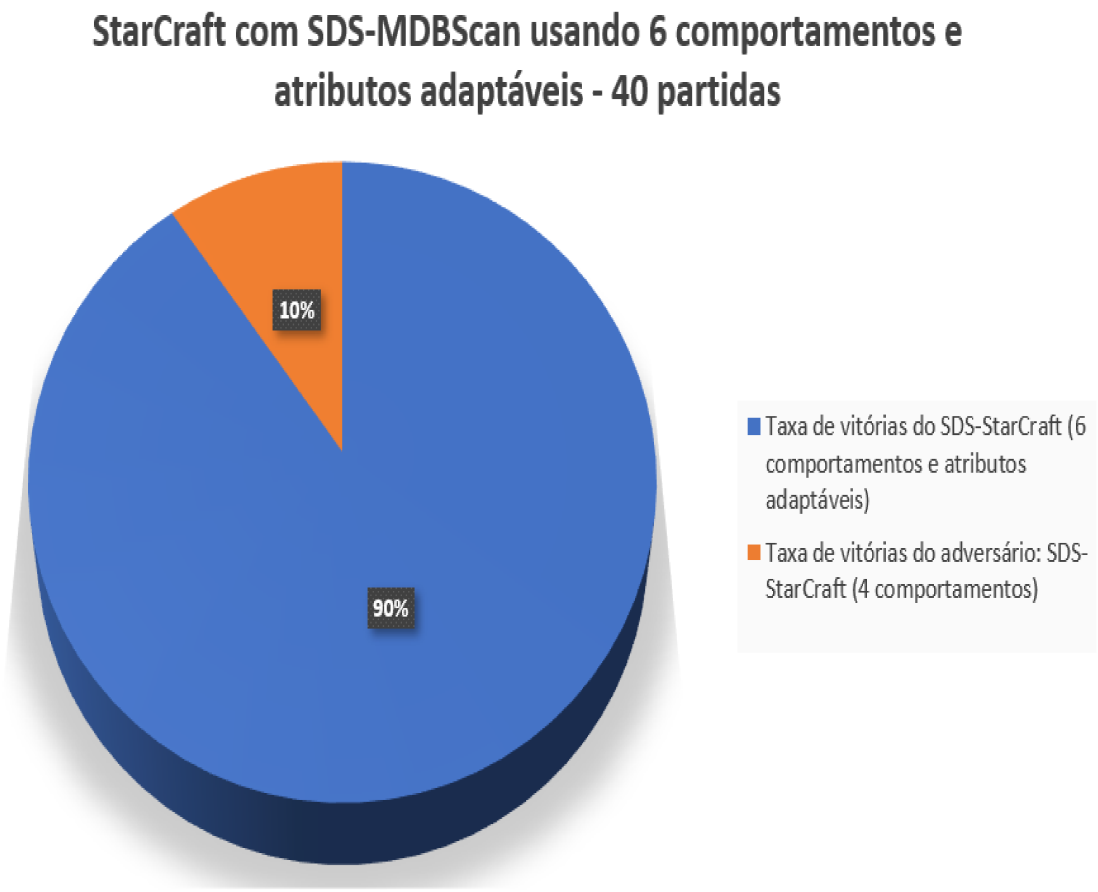


Figura 45 – Percentual de vitórias em partidas do StarCraft: SDS-MDBScan com 6 comportamentos e conjunto de atributos adaptáveis por estágio de jogo.

possível saber se o jogador foi mais agressivo nos confrontos, independentemente do resultado final da partida, que simplesmente indica se o jogador foi o vencedor ou o perdedor.

Na Figura 46 é possível observar ganhos do agente na média da pontuação *kill score*. Nesta figura observa-se, que em média o agente SDS-StarCraft (4 comportamentos) teve um *kill score* de 4660 contra o valor médio de 1966,75 obtido pelo agente UAlbertbot, que não utiliza SDS-MDBScan.

Considerando os ganhos do agente SDS-StarCraft (6 comportamentos) na pontuação (*kill score*), observa-se na Figura 46, que em média, o SDS-StarCraft (6 comportamentos) teve um *kill score* de 9944,3, que é consideravelmente maior do que o valor médio de 4660 obtido pelo agente SDS-StarCraft (4 comportamentos), quando ele enfrentou um adversário mais fraco, que foi o agente UAlbertaBot.

Considerando o agente SDS-StarCraft (6 comportamentos e atributos adaptáveis), os ganhos na pontuação *kill score* também apresentou uma elevação, onde este agente, em média, teve um *kill score* de 11507,73 (conforme apresentado na Figura 46), que é maior que a média de 9944,3, obtida pelo agente SDS-StarCraft (6 comportamentos), quando enfrentou o mesmo adversário, o agente SDS-StarCraft (4 comportamentos).

O aumento na média do *kill score* é um indício de que, independente do resultado da partida,

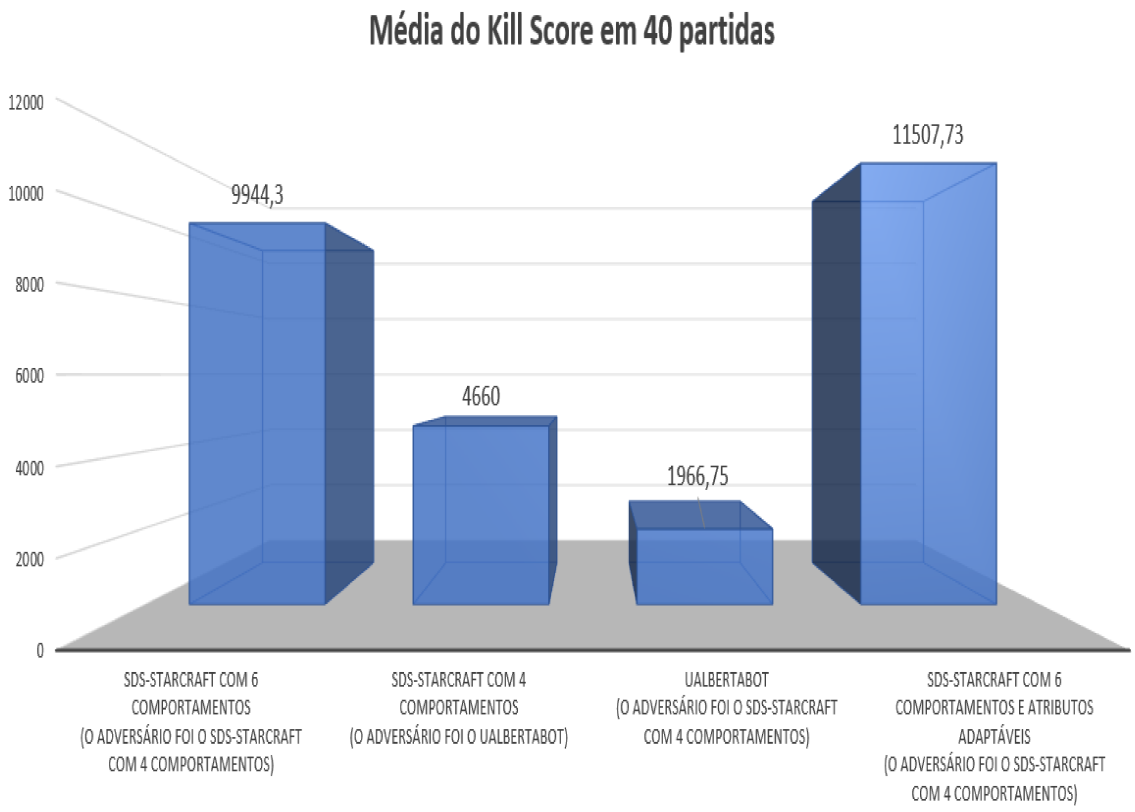


Figura 46 – Comparando a média do kill score em partidas do StarCraft com SDS-MDBScan.

o agente SDS-StarCraft está mais competitivo e mais agressivo. Então, mesmo que, eventualmente, o agente seja derrotado no final da partida, ele conseguiu elevar a pontuação em um critério onde os comportamentos definidos para o problema realmente atuam, que é o confronto direto das unidades.

A fim de mensurar estatisticamente o ganho que o uso do SDS-MDBScan pode ter proporcionado ao agente do StarCraft, em todas as suas versões analisadas, o teste de Wilcoxon foi aplicado, usando a ferramenta KEEL (ALCALÁ-FDEZ et al., 2011). As três situações onde foi usado o teste de Wilcoxon são as seguintes: na primeira compara-se os valores de *kill score* obtidos pelo agente SDS-StarCraft (4 comportamentos) com os valores obtidos pelo agente UAlbertaBot, que não utiliza SDS-MDBScan; na segunda comparação utiliza-se os valores de *kill score* obtidos pelo agente SDS-StarCraft (6 comportamentos) e os valores obtidos pelo agente SDS-StarCraft (4 comportamentos); na terceira comparação utiliza-se os valores de *kill score* obtidos pelo agente SDS-StarCraft (6 comportamentos e atributos adaptáveis) e os valores obtidos pelo agente SDS-StarCraft (6 comportamentos).

Então, considerando a primeira situação analisada, foi definida a seguinte hipótese nula: *O agente SDS-StarCraft (4 comportamentos) tem um desempenho similar ao obtido pelo agente UAlbertaBot*. Com base na média do *kill score* apresentado na Figura 46, observa-se que o agente SDS-StarCraft (4 comportamentos) apresenta uma média superior ao agente UAlbertaBot, possibilitando a definição de uma hipótese alternativa: *O agente SDS-StarCraft (4 compor-*

tamentos) tem um desempenho superior ao obtido pelo agente UAlbertaBot. Com a aplicação do teste de Wilcoxon, tem-se R^+ igual a 598,5 e R^- igual a 221,5, com p-valor assintótico igual a 0,010413, que permite rejeitar a hipótese nula e afirmar a hipótese alternativa.

Analisando a segunda situação, foi definida a seguinte hipótese nula: *O agente SDS-StarCraft (6 comportamentos) tem um desempenho similar ao obtido pelo agente SDS-StarCraft (4 comportamentos)*. Com base na média do *kill score* apresentado na Figura 46, observa-se que o agente SDS-StarCraft (6 comportamentos) apresenta uma média superior ao agente SDS-StarCraft (4 comportamentos), possibilitando a definição de uma hipótese alternativa: *O agente SDS-StarCraft (6 comportamentos) tem um desempenho superior ao obtido pelo agente SDS-StarCraft (4 comportamentos)*. Com a aplicação do teste de Wilcoxon, tem-se R^+ igual a 703 e R^- igual a 117, com p-valor assintótico igual a 0,000069, que permite rejeitar a hipótese nula e afirmar a hipótese alternativa.

Analisando a terceira situação, foi definida a seguinte hipótese nula: *O agente SDS-StarCraft (6 comportamentos) e o agente SDS-StarCraft (6 comportamentos e atributos adaptáveis) possuem desempenho semelhantes*. Com base na média do *kill score*, que é favorável ao agente SDS-StarCraft (6 comportamentos e atributos adaptáveis) é possível definir a seguinte hipótese alternativa: *O agente SDS-StarCraft (6 comportamentos e atributos adaptáveis) possui desempenho superior a do agente SDS-StarCraft (6 comportamentos)*. Com a aplicação do teste de Wilcoxon, tem-se R^+ igual a 571,5 e R^- igual a 248,5, com p-valor assintótico igual a 0,026781, que permite rejeitar a hipótese nula e afirmar a hipótese alternativa.

Logo, o agente SDS-StarCraft (4 comportamentos) apresentou ganhos quando comparado com o agente UAlbertaBot. Além disso, o conjunto estendido de comportamentos (6 comportamentos) ampliou os ganhos do agente SDS-StarCraft (4 comportamentos), que são observáveis tanto em vitórias quanto na pontuação *kill score*. O uso de atributos adaptáveis com o agente SDS-StarCraft (6 comportamentos e atributos adaptáveis) reconfirmou os resultados obtidos na etapa 2, proporcionando ganhos ao agente tanto em termo de vitória, quanto em pontuação *kill score*.

7.3 Considerações finais

Com a informação semântica produzida na etapa 3 (capítulo 6), foi possível executar a etapa 4 da pesquisa, que consiste em utilizar a semântica da mudança de comportamento para aprimorar o processo de tomada de decisão do motor do jogo StarCraft. O SDS-MDBScan com KNN foi a versão usada nesta etapa, devido as propriedades de algoritmo incremental que o KNN apresenta e pelo SDS-MDBScan ter apresentado melhores resultados que o Semantic-MDBScan. Esta incorporação do SDS-MDBScan ao motor do jogo StarCraft permitiu a criação do agente SDS-StarCraft.

Antes de implantar o SDS-MDBScan foi preciso mapear a tendência das ações do adversário com base nos significados de comportamentos, de tal modo que o agente pudesse reagir em

conformidade com tais mudanças. Logo, foi preciso mapear um conjunto de ações para cada semântica de comportamento utilizada. Inicialmente, foi definido um conjunto de 4 comportamentos, sendo que cada comportamento possui suas próprias características, o que permite implementar seu conjunto de ações no contexto do jogo de maneira distinta.

Com o desenvolvimento do SDS-StarCraft, foram obtidos ótimos resultados com taxa de vitória de 77%, usando 4 comportamentos. Além disso, foi possível observar nos experimentos, que quanto mais eram acionadas as ações contextualizadas ao comportamento do adversário, melhores eram os resultados. Por isso, aplicando uma análise de agrupamento, onde cada grupo seria um comportamento, observou-se que seria adequado ampliar o conjunto de comportamentos usado, aumentando de 4 para 6 comportamentos. Com isto, além dos ganhos nas taxas de vitória, que foram de 87%, também ficou evidente ganhos na pontuação *kill score*, indicando que o agente se tornou mais ofensivo nas partidas.

Devido aos bons resultados observados com a etapa 2 da pesquisa, onde se utilizou conjuntos adaptáveis de atributos aos estágios de jogo. Esta ideia, foi, então, usada de maneira adaptada na etapa 4, junto ao agente SDS-StarCraft. Os resultados obtidos apresentaram melhoras, atingindo uma taxa de vitória de 90%, além do aumento na pontuação *kill score*.

Com os resultados desta quarta etapa, a última hipótese apresentada na Seção 1.3 foi validada: “Um agente cujo motor de tomada de decisão conta com um módulo capaz de identificar mudanças de comportamento, atribuir uma semântica a elas e usar tal significado para norteá-lo na escolha de suas ações, tem um melhor desempenho.”.

No capítulo seguinte é feita uma conclusão de toda a pesquisa, indicando as contribuições alcançadas com os resultados obtidos, dentre elas as produções bibliográficas.

CAPÍTULO 8

Conclusão

Este trabalho apresentou uma nova abordagem para o desenvolvimento de agentes aptos a tomar decisões, em tempo real, em cenários competitivos de fluxo contínuo de dados onde um oponente age para minimizar seu sucesso nas conclusões de suas tarefas. Tal abordagem dota tais agentes da capacidade de detectar mudanças nos comportamentos de seus adversários, interpretando tais mudanças e usando seus significados para norteá-lo no sentido de reagir em conformidade com o que foi detectado.

O jogo de RTS StarCraft foi escolhido como cenário de estudo. O M-DBScan foi usado como algoritmo de base para os dois novos detectores de mudanças de comportamento em cenário de fluxo contínuo de dados aqui propostos e implementados - denominados Semantic-MDBScan e SDS-MDBScan - ambos estendendo o M-DBScan pela habilidade de identificar o significado das mudanças encontradas. O SDS-MDBScan, além disso, também conta com um processo de detecção de mudanças aprimorado, por meio de análises estatísticas, que lhe confere uma acurácia no processo de detecção de mudanças maior do que a do M-DBScan e do Semantic-MDBScan.

A título de concretizar e validar a abordagem proposta, implementou-se o agente jogador de StarCraft denominado SDS-StarCraft, criado com base no agente UAlbertaBot, cujo motor de tomada de decisão é constituído pelas regras padrão do StarCraft. No caso, a arquitetura do novo agente proposto expande a do agente base nos seguintes aspectos: 1) Inclusão de um módulo de detecção e atribuição de significado às mudanças de comportamento (sendo incorporado o SDS-MDBScan); 2) Inclusão de novas regras no motor padrão de tomada de decisão de forma que, sempre que for detectada uma mudança de comportamento do oponente e for identificado um significado para tal mudança, essas regras definem um conjunto de ações que o agente deve executar a fim de reagir de modo condizente com aquela mudança.

Os experimentos conduzidos na primeira etapa desta pesquisa comprovaram a viabilidade de se usar o M-DBScan como algoritmo de base para identificar mudanças de comportamento no cenário de jogos RTS.

Os experimentos da segunda etapa confirmaram o ganho de acurácia na detecção de mudanças no comportamento do adversário obtida com o uso de diferentes conjuntos de atributos, em

diferentes momentos de uma partida de StarCraft.

Os experimentos feitos na terceira etapa ratificaram os avanços obtidos pelos algoritmos Semantic-MDBScan e SDS-MDBScan, aqui propostos, em relação ao MDBScan, além de evidenciarem a superioridade do SDS-MDBScan sobre o Semantic-MDBScan.

Tanto o Semantic-MDBScan quanto SDS-MDBScan utilizaram, de maneiras distintas e satisfatórias, os algoritmos de classificação KNN e MLP na tarefa de definir a semântica da mudança de comportamento. Contudo, o KNN, em virtude de sua natureza incremental (não presente no MLP), mostrou-se mais adequado para o cenários de problemas dinâmicos como o de jogos. Por isso, o algoritmo SDS-MDBScan com KNN foi o escolhido para a implantação do módulo semântico do SDS-StarCraft.

Os experimentos executados na quarta etapa da pesquisa, em que o SDS-StarCraft enfrenta o UAlbertaBot, confirmaram o expressivo aumento de desempenho que a abordagem aqui proposta conferiu ao primeiro. De fato, o SDS-StarCraft, implementado com 6 tipos distintos de comportamento do oponente, obteve 87% de vitórias nos torneios realizados, além de ter alcançado um significativo aumento na pontuação *kill score* - que reflete o potencial ofensivo do agente - mesmo nas partidas em que ele não foi o vencedor. Ainda na quarta etapa, provou-se que a taxa de vitória do agente saltou para 90% nos experimentos em que o SDS-StarCraft operou com 6 comportamentos e diferentes conjuntos de atributos para fases distintas de jogo. Finalmente, os experimentos da quarta etapa mostraram um melhor desempenho do agente nas partidas onde as tomadas de decisão baseadas no significado de mudança de comportamento foram mais frequentes, que é uma das contribuições desta pesquisa.

8.1 Validação das hipóteses via experimentos

Resume-se aqui onde cada uma das hipóteses levantadas na subseção 1.3 foi validada no decorrer das pesquisas conduzidas.

Os experimentos da primeira etapa da pesquisa (Seção 4.2), demonstraram que o uso de um conjunto adequado de atributos na representação dos cenários de jogo permitiu que o M-DBScan mapeasse, com boa acurácia, as mudanças de comportamento do oponente do agente, tanto utilizando a entropia espacial quanto a temporal. Validou-se, assim, a primeira hipótese apresentada.

A segunda hipótese diz que o uso de diferentes conjuntos de atributos, para diferentes momentos do jogo, proporcionam ganhos de acurácia no processo de detecção de mudanças no comportamento do adversário (efetuado pelo M-DBScan), e ela foi validada com os experimentos da Etapa 2 da pesquisa (Seção 5.2).

Os resultados apresentados pelas abordagens Semantic-MDBScan e SDS-MDBScan na Etapa 3 da pesquisa (Seção 6.4) corroboraram a hipótese de que a acurácia do M-DBScan pode ser aumentada por meio de diferentes abordagens de detecção de mudanças que, além de incorporarem a habilidade de identificar o significado das mudanças detectadas, no caso

do SDS-MDBScan, usa-se um critério estatístico para detectar tais mudanças, baseando-se na relevância estatística de uma semântica em uma sequência de dados.

Por fim, as taxas de vitória e a pontuação *kill score* obtidas pelo agente SDS-StarCraft nos experimentos da Etapa 4 (Seção 7.2) comprovaram o teor da quarta hipótese, a qual prevê a significativa melhora de desempenho obtida por um agente que opera em ambiente competitivo, contando com a seguinte arquitetura: um módulo capaz de mapear as mudanças de comportamento de seu oponente e de atribuir um significado a elas; um modulo de tomada de decisão que use tal significado como parâmetro para norteá-lo na maneira adequada com que deve reagir a tais mudanças.

8.2 Trabalhos Futuros

Como trabalhos futuros os autores pretendem estender a aplicação da abordagem aqui proposta para outros tipos de problemas em que o mapeamento da mudança de comportamento de outros agentes, bem como a identificação do significado dessas mudanças, possam auxiliar na tomada de decisão do agente concebido para lidar com esses desafios. Como exemplo dessas aplicações adicionais, citam-se:

- ❏ Avaliação da tendência do mercado financeiro em função das conjunturas;
- ❏ Detecção de fraudes;
- ❏ Uso de agentes jogadores para estimular as habilidades cognitivas de portadores de síndromes diversas (no caso, mapeando o perfil de comportamento desses pacientes, detectando suas fragilidades e adaptando seu motor de tomada de decisão de forma a auxiliá-los a melhor escolher suas ações);
- ❏ Detectar e ajustar o nível de dificuldade de um jogo digital, com base nas mudanças de comportamento que um jogador pode apresentar durante uma partida;
- ❏ Identificar a mudança de opinião de uma pessoa com base nas informações que ela fornece ou que são identificadas automaticamente;
- ❏ Dentre outros;

Esses cenários adicionais, passíveis de investigação, servem para demonstrar a relevância da pesquisa aqui desenvolvida, bem como de seu potencial de generalização, podendo ser aplicada a problemas de diferentes áreas. Para tanto, considerando as diversas variantes associadas à natureza de cada um desses problemas, tais investigações demandariam a definição das seguintes informações: Os comportamentos relevantes para cada problema; Os atributos e seus valores correspondentes que permitirão identificar e distinguir cada comportamento; As reações que deverão ser executadas considerando o comportamento identificado. Diante disto, destaca-se a relevância do especialista na tarefa de adaptação da técnica para outros problemas.

Em futuros estudos, os autores pretendem também investigar e propor novas técnicas de detecção de mudanças de comportamento e de identificação do significado dessas mudanças baseadas em Aprendizagem Profunda. Além disso, os autores também intencionam testar outras medidas de entropia, bem como desenvolver uma versão semi-supervisionada do SDS-MDBScan, a qual otimizaria a tarefa de atribuir uma semântica para situações já conhecidas pelo algoritmo.

Referências

AL-NABI, D. L. A.; AHMED, S. S. Survey on classification algorithms for data mining: comparison and evaluation. **International Journal of Computer Engineering and Intelligent Systems**, Citeseer, v. 4, n. 8, p. 18–27, 2013.

ALCALÁ-FDEZ, J. et al. Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. **Journal of Multiple-Valued Logic & Soft Computing**, Citeseer, v. 17, 2011.

AMARAL, F. **Aprenda mineração de dados: teoria e prática**. [S.l.]: Alta Books Editora, 2016. v. 1.

____. **Introdução à Ciência de Dados: mineração de dados e big data**. [S.l.]: Alta Books Editora, 2016.

ARRIETA, A. B. et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. **Information fusion**, Elsevier, v. 58, p. 82–115, 2020. Disponível em: <https://doi.org/10.1016/j.inffus.2019.12.012>.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. **Computer networks**, Elsevier, v. 54, n. 15, p. 2787–2805, 2010. Disponível em: <https://doi.org/10.1016/j.comnet.2010.05.010>.

BABCOCK, B. et al. Models and issues in data stream systems. In: ACM. **Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems**. 2002. p. 1–16. Disponível em: <https://doi.org/10.1145/543613.543615>.

BALLINGER, C.; LIU, S.; LOUIS, S. J. Identifying players and predicting actions from rts game replays. 2014.

BIFET, A.; GAVALDA, R. Learning from time-changing data with adaptive windowing. In: SIAM. **Proceedings of the 2007 SIAM international conference on data mining**. 2007. p. 443–448. Disponível em: <https://doi.org/10.1137/1.9781611972771.42>.

BIFET, A. et al. MOA: massive online analysis. **J. Mach. Learn. Res.**, v. 11, p. 1601–1604, 2010. Disponível em: <http://portal.acm.org/citation.cfm?id=1859903>.

BODYANSKIY, Y.; TYSHCHENKO, O.; KOPALIANI, D. An evolving cascade neuro-fuzzy system for data stream fuzzy clustering. **International Journal of Computer Science and Mobile Computing (IJCSMC)**, v. 4, n. 9, p. 270–275, 2015. Disponível em:

<https://www.researchgate.net/publication/283505490_An_Evolving_Cascade_Neuro-Fuzzy_System_for_Data_Stream_Fuzzy_Clustering>.

CAO, F. et al. Density-based clustering over an evolving data stream with noise. In: SIAM. **Proceedings of the 2006 SIAM international conference on data mining**. 2006. p. 328–339. Disponível em: <https://doi.org/10.1137/1.9781611972764.29>.

CHARLES, D.; BLACK, M. Dynamic player modeling: A framework for player-centered digital games. In: **Proc. of the International Conference on Computer Games: Artificial Intelligence, Design and Education**. [S.l.: s.n.], 2004. p. 29–35.

CHEN, M.; MAO, S.; LIU, Y. Big data: A survey. **Mobile networks and applications**, Springer, v. 19, n. 2, p. 171–209, 2014. Disponível em: <https://doi.org/10.1007/s11036-013-0489-0>.

CHURCHILL, D. **GitHub UAlbertaBot**. 2016. Disponível em: <https://github.com/davechurchill/ualbertabot/wiki>.

____. **Design and Architecture**. 2020. Disponível em: <https://github.com/davechurchill/ualbertabot/wiki/Design-and-Architecture>.

CHURCHILL, D.; BURO, M. Build order optimization in starcraft. In: **AIIDE**. [s.n.], 2011. p. 14–19. Disponível em: <https://doi.org/10.1609/aiide.v7i1.12435>.

CHURCHILL, D.; SAFFIDINE, A.; BURO, M. Fast heuristic search for rts game combat scenarios. In: **AIIDE**. [s.n.], 2012. p. 112–117. Disponível em: <https://doi.org/10.1609/aiide.v8i1.12527>.

DACHOWICZ, A. et al. Mission engineering and design using real-time strategy games: An explainable ai approach. **Journal of Mechanical Design**, American Society of Mechanical Engineers Digital Collection, v. 144, n. 2, 2022. Disponível em: <https://doi.org/10.1115/1.4052841>.

DERRAC, J. et al. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. **Swarm and Evolutionary Computation**, Elsevier, v. 1, n. 1, p. 3–18, 2011. Disponível em: <https://doi.org/10.1016/j.swevo.2011.02.002>.

DUDANI, S. A. The distance-weighted k-nearest-neighbor rule. **IEEE Transactions on Systems, Man, and Cybernetics**, IEEE, n. 4, p. 325–327, 1976. Disponível em: <https://doi.org/10.1109/TSMC.1976.5408784>.

ENTERTAINMENT, B. Starcraft: Brood war. **PC. Level/area: Episode IV, mission**, v. 2, 1998.

ESTER, M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: **Kdd**. [S.l.: s.n.], 1996. v. 96, n. 34, p. 226–231.

FACELI, K. et al. **Inteligência Artificial: Uma abordagem de aprendizado de máquina**. [S.l.]: LTC, 2011.

FARIA, E. R.; GAMA, J.; CARVALHO, A. C. Novelty detection algorithm for data streams multi-class problems. In: ACM. **Proceedings of the 28th annual ACM symposium on applied computing**. 2013. p. 795–800. Disponível em: <https://doi.org/10.1145/2480362.2480515>.

FARIA, E. R. et al. Novelty detection in data streams. **Artificial Intelligence Review**, Springer, v. 45, n. 2, p. 235–269, 2016. Disponível em: <https://doi.org/10.1007/s10462-015-9444-8>.

FARKAS, B. **Prima's Official Strategy Guide: StarCraft**. [S.l.]: Prima Games, 2002.

FAWCETT, T.; PROVOST, F. Activity monitoring: Noticing interesting changes in behavior. In: **Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining**. [s.n.], 1999. p. 53–62. Disponível em: <https://doi.org/10.1145/312129.312195>.

GAMA, J. **Knowledge discovery from data streams**. CRC Press, 2010. Disponível em: <https://doi.org/10.1201/EBK1439826119>.

GAMA, J.; GABER, M. M. **Learning from data streams: processing techniques in sensor networks**. Springer, 2007. Disponível em: <https://doi.org/10.1007/3-540-73679-4>.

GAMA, J. et al. A survey on concept drift adaptation. **ACM computing surveys (CSUR)**, ACM New York, NY, USA, v. 46, n. 4, p. 1–37, 2014. Disponível em: <https://doi.org/10.1145/2523813>.

GOMES, H. M. et al. A survey on ensemble learning for data stream classification. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 50, n. 2, p. 1–36, 2017. Disponível em: <https://doi.org/10.1145/3054925>.

GUTIERREZ-GALAN, D. et al. Embedded neural network for real-time animal behavior classification. **Neurocomputing**, Elsevier, v. 272, p. 17–26, 2018. Disponível em: <https://doi.org/10.1016/j.neucom.2017.03.090>.

HAQUE, A.; KHAN, L.; BARON, M. Sand: Semi-supervised adaptive novel class detection and classification over data stream. In: **THIRTIETH AAAI Conference on Artificial Intelligence**. [s.n.], 2016. Disponível em: <https://doi.org/10.1609/aaai.v30i1.10283>.

HATAMIKHAH, N. et al. Concept drift detection via improved deep belief network. In: IEEE. **Electrical Engineering (ICEE), Iranian Conference on**. 2018. p. 1703–1707. Disponível em: <https://doi.org/10.1109/ICEE.2018.8472481>.

HAYKIN, S. **Neural networks and learning machines, 3/E**. [S.l.]: Pearson Education India, 2010.

HINTON, G. E.; OSINDERO, S.; TEH, Y.-W. A fast learning algorithm for deep belief nets. **Neural computation**, MIT Press, v. 18, n. 7, p. 1527–1554, 2006. Disponível em: <https://doi.org/10.1162/neco.2006.18.7.1527>.

HOTZ, L. et al. **Configuration knowledge representation and reasoning**. Tese (Doutorado) — Morgan Kaufmann Amsterdam, 2014. Disponível em: <https://doi.org/10.1016/B978-0-12-415817-7.00006-2>.

JALALI, L. et al. Human behavior analysis from smartphone data streams. In: SPRINGER. **International Workshop on Human Behavior Understanding**. [S.l.], 2016. p. 68–85.

JOHNSON, D.; WILES, J. Computer games with intelligence. In: IEEE. **10th IEEE International Conference on Fuzzy Systems.(Cat. No. 01CH37297)**. [S.l.], 2001. v. 3, p. 1355–1358.

JUSTESEN, N.; RISI, S. Continual online evolutionary planning for in-game build order adaptation in starcraft. In: ACM. **Proceedings of the Genetic and Evolutionary Computation Conference**. 2017. p. 187–194. Disponível em: <https://doi.org/10.1145/3071178.3071210>.

KRAWCZYK, B. et al. Ensemble learning for data stream analysis: A survey. **Information Fusion**, Elsevier, v. 37, p. 132–156, 2017. Disponível em: <https://doi.org/10.1016/j.inffus.2017.02.004>.

KURDZIOLEK, M. Explaining the unexplainable: Explainable ai (xai) for ux. 2022.

MACQUEEN, J. Classification and analysis of multivariate observations. In: **5th Berkeley Symp. Math. Statist. Probability**. [S.l.: s.n.], 1967. p. 281–297.

MUTHUKRISHNAN, S. et al. Data streams: Algorithms and applications. **Foundations and Trends® in Theoretical Computer Science**, Now Publishers, Inc., v. 1, n. 2, p. 117–236, 2005. Disponível em: <https://doi.org/10.1561/0400000002>.

NETO, H. C.; JULIA, R. M. S. Ace-rl-checkers: decision-making adaptability through integration of automatic case elicitation, reinforcement learning, and sequential pattern mining. **Knowledge and Information Systems**, Springer, v. 57, n. 3, p. 603–634, 2018. Disponível em: <https://doi.org/10.1007/s10115-018-1175-0>.

NETO, H. d. 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) — Universidade Federal de Uberlândia, 2016.

PAGE, E. S. Continuous inspection schemes. **Biometrika**, JSTOR, v. 41, n. 1/2, p. 100–115, 1954. Disponível em: <https://doi.org/10.1093/biomet/41.1-2.100>.

PENG, P. et al. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games. **arXiv preprint arXiv:1703.10069**, 2017.

READ, J.; PEREZ-CRUZ, F.; BIFET, A. Deep learning in partially-labeled data streams. In: ACM. **Proceedings of the 30th Annual ACM Symposium on Applied Computing**. 2015. p. 954–959. Disponível em: <https://doi.org/10.1145/2695664.2695871>.

RUSSELL, S. J.; NORVIG, P. **Artificial intelligence: a modern approach**. [S.l.]: Malaysia; Pearson Education Limited,, 2016.

SCHADD, F.; BAKKES, S.; SPRONCK, P. Opponent modeling in real-time strategy games. In: **GAMEON**. [S.l.: s.n.], 2007. p. 61–70.

SCHAEFFER, J.; HERIK, H. J. Van den. Games, computers, and artificial intelligence. **Artificial intelligence**, Elsevier, v. 134, n. 1-2, p. 1–7, 2002. Disponível em: <https://doi.org/10.1016/S0004-3702(01)00165-5>.

SEBASTIAO, R.; GAMA, J. A study on change detection methods. In: **Progress in artificial intelligence, 14th Portuguese conference on artificial intelligence, EPIA**. [S.l.: s.n.], 2009. p. 12–15.

SHAO, K.; ZHU, Y.; ZHAO, D. Cooperative reinforcement learning for multiple units combat in starcraft. In: IEEE. **Computational Intelligence (SSCI), 2017 IEEE Symposium Series on**. 2017. p. 1–6. Disponível em: <https://doi.org/10.1109/SSCI.2017.8280949>.

SILVER, D. et al. Mastering the game of go without human knowledge. **nature**, Nature Publishing Group, v. 550, n. 7676, p. 354–359, 2017. Disponível em: <https://doi.org/10.1038/nature24270>.

STANESCU, M.; ČERTICKỲ, M. Predicting opponent's production in real-time strategy games with answer set programming. **IEEE Transactions on Computational Intelligence and AI in Games**, IEEE, v. 8, n. 1, p. 89–94, 2014.

SYNNAEVE, G.; BESSIERE, P. et al. A dataset for starcraft ai & an example of armies clustering. In: **AIIDE Workshop on AI in Adversarial Real-time games**. [S.l.: s.n.], 2012. v. 2012.

TEAM, B. W. A. **Brood War API Team**. 2015. Disponível em: <https://github.com/bwapi/>.

The AlphaStar team. **AlphaStar: Mastering the Real-Time Strategy Game StarCraft II**. 2019. Disponível em: <https://www.deepmind.com/blog/alphastar-mastering-the-real-time-strategy-game-starcraft-ii>.

VALLIM, R. M. et al. Online behavior change detection in computer games. **Expert Systems with Applications**, Elsevier, v. 40, n. 16, p. 6258–6265, 2013. Disponível em: <https://doi.org/10.1016/j.eswa.2013.05.059>.

VALLIM, R. M. M. **Mineração de fluxos contínuos de dados para jogos de computador**. Tese (Doutorado) — Universidade de São Paulo, 2013.

VIEIRA, E.; JULIA, R. M. S.; FARIA, E. R. de. Mining data stream to detect behavior change in a real-time strategy game. In: **Proceedings of Machine Learning and Data Mining in Pattern Recognition**. [S.l.]: ibai publishing, 2019.

WEBER, B. G.; MATEAS, M. A data mining approach to strategy prediction. In: IEEE. **Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on**. 2009. p. 140–147. Disponível em: <https://doi.org/10.1109/CIG.2009.5286483>.

WITTEN, I. H. et al. **Data Mining: Practical machine learning tools and techniques**. [S.l.]: Morgan Kaufmann, 2016.

XIANG, S. et al. Dual-task semantic change detection for remote sensing images using the generative change field module. **Remote Sensing**, Multidisciplinary Digital Publishing Institute, v. 13, n. 16, p. 3336, 2021. Disponível em: <https://doi.org/10.3390/rs13163336>.

YANNAKAKIS, G. N.; MARAGOUDAKIS, M. Player modeling impact on player's entertainment in computer games. In: SPRINGER. **International Conference on User Modeling**. 2005. p. 74–78. Disponível em: <https://doi.org/10.1007/11527886_11>.

YANNAKAKIS, G. N. et al. Player modeling. In: SCHLOSS DAGSTUHL-LEIBNIZ-ZENTRUM FUER INFORMATIK. **Dagstuhl Follow-Ups**. [S.l.], 2013. v. 6.