Universidade Federal de Santa Maria
Centro de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Gustavo Bathu Paulus

CASOS E CLUSTERS NO DESENVOLVIMENTO DE POLÍTICAS DE REUSO PARA TOMADA DE DECISÃO EM JOGOS DE CARTAS

Santa Maria, RS
2020

CASOS E CLUSTERS NO DESENVOLVIMENTO DE POLÍTICAS DE REUSO PARA TOMADA DE DECISÃO EM JOGOS DE CARTAS

Dissertação apresentada ao Curso de Pós-Graduação em Ciência da computação da Universidade Federal de Santa Maria (UFSM), como requisito parcial para obtenção do título de **Mestre em Ciência da Computação.**

Orientador: Prof. Dr. Luís Alvaro de Lima Silva

Santa Maria, RS
2020

Paulus, Gustavo
CASOS E CLUSTERS NO DESENVOLVIMENTO DE POLÍTICAS DE REUSO PARA TOMADA DE DECISÃO EM JOGOS DE CARTAS / Gustavo Paulus.- 2020.
121 p.; 30 cm

Orientador: Luís Alvaro de Lima Silva
Dissertação (mestrado) - Universidade Federal de Santa Maria, Centro de Tecnologia, Programa de Pós-Graduação em Ciência da Computação , RS, 2020

1. Políticas de Reuso 2. Raciocínio Baseado em Casos 3. Clustering 4. Jogos de cartas 5. Jogo de Truco I. de Lima Silva, Luís Alvaro II. Título.

Sistema de geração automática de ficha catalográfica da UFSM. Dados fornecidos pelo autor(a). Sob supervisão da Direção da Divisão de Processos Técnicos da Biblioteca Central. Bibliotecária responsável Paula Schoenfeldt Patta CRB 10/1728.

Declaro, GUSTAVO PAULUS, para os devidos fins e sob as penas da lei, que a pesquisa constante neste trabalho de conclusão de curso (Dissertação) foi por mim elaborada e que as informações necessárias objeto de consulta em literatura e outras fontes estão devidamente referenciadas. Declaro, ainda, que este trabalho ou parte dele não foi apresentado anteriormente para obtenção de qualquer outro grau acadêmico, estando ciente de que a inveracidade da presente declaração poderá resultar na anulação da titulação pela Universidade, entre outras consequências legais.

Gustavo Bathu Paulus

CASOS E CLUSTERS NO DESENVOLVIMENTO DE POLÍTICAS DE REUSO PARA TOMADA DE DECISÃO EM JOGOS DE CARTAS

Dissertação apresentada ao Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Maria (UFSM), como requisito parcial para a obtenção do título de **Mestre em Ciência da Computação.**

Aprovado em 27 **de** Março **de** 2020**:**

Luís Alvaro de Lima Silva, Dr (UFSM)
(Presidente/Orientador)

Felipe Martins Muller, Dr (UFSM)

Rafael Heitor Bordini, Dr (PUCRS)

Santa Maria, RS
2020

DEDICATÓRIA

A todos que de alguma maneira contribuíram para a realização desta pesquisa. Dedico essa pesquisa principalmente a minha noiva Elviane, aos meus pais Eraldo e Ana, aos meus irmãos Rafael e Gabriel e a todas as pessoas que colaboraram para a realização deste trabalho.

AGRADECIMENTOS

Acredito que ninguém entra em nossa vida por acaso, todos tem uma proposta, todos deixam um pouquinho de si nas pessoas com quem convivem. Durante este tempo diversas pessoas deixaram muito de si comigo, são aprendizados que levarei para a vida inteira.

Primeiramente, agradeço a deus pelo dom da vida. Também gostaria de agradecer imensamente a minha noiva Elviane, obrigado pela sua paciência, pelos diversos sábados e domingos que ficou sentada ao meu lado enquanto eu trabalhava na pesquisa, obrigado por ter me acalmado nos momentos de nervosismo e ter ficado comigo em tempos difíceis.

Aos meus pais, pessoas humildes, mas com uma capacidade esplêndida. Obrigado por me ensinarem a correr atrás dos meus sonhos e por sempre me incentivarem a continuar lutando. Amo e tenho muito orgulho de vocês.

Ao meu irmão Rafael, muito obrigado por ter sido presença constante durante a etapa de coleta de casos e testes. As ideias que trocamos toda a vez que te chamava e pedia para testar algum dos agentes (desde a primeira versão feita em 2017) desenvolvidos contribuíram e muito na minha percepção e adaptação das técnicas para o cenário de tomada de decisão com visão parcial.

A todas as pessoas que se prontificaram a validar os agentes virtuais e trocaram ideias comigo a respeito da pesquisa. De forma especial, ao meu amigo Lucas Tarasconi, as ideias que trocamos proporcionaram compreensões que permitiram a evolução do Trabalho.

Aos meus colegas de sala (Daniel, Evaristo, Marcos e Ruan) obrigado pelos momentos compartilhados. Vocês foram presença constante durante esta pesquisa.

Aos Professores, Dr Rafael Bordini e Felipe Muller pelo aceite em participar desta banca.

Ao professor Dr Luís Alvaro de Lima Silva, por aceitar orientar-me nessa empreitada. A convivência com o Senhor me proporcionou evoluir como ser humano, agradeço por todos os conselhos que me deste.

Ao Professor Dr Joaquim Carvalho Assunção, por todas as dicas sempre muito ponderadas que permitiram a evolução da pesquisa. Também agraço pelos conselhos que me deste durante nossas reuniões.

Ao IFRS, obrigado pela oportunidade de fazer o mestrado. De forma especial aos colegas do IFRS-Campus Ibirubá que contribuíram para realização deste sonho.

Ser fiel aos nossos objetivos exige mais dedicação do que planejamento

RESUMO

Casos e Clusters no desenvolvimento de Políticas de Reuso para Tomada de Decisão em Jogos de Cartas

AUTOR: Gustavo Bathu Paulus
ORIENTADOR: Luís Alvaro de Lima Silva

Este trabalho investiga a combinação de casos e clusters aplicados ao reuso de ações de jogo (por exemplo, cartas jogadas, apostas efetuadas) presentes em casos recuperados de uma consulta feita a um agente de Raciocínio Baseado em Casos (CBR) projetado para jogar cartas. Com o suporte do algoritmo de agrupamento de dados K-MEANS, critérios de reuso são aplicados a grupos identificados nos casos presentes na base de casos onde as ações de jogo são relacionadas com o estado em que foram feitas. Inicialmente um critério é aplicado para escolher o grupo que deve ser acessado. Em seguida, outro critério de reuso utiliza os casos que pertencem ao cluster selecionado para determinar qual ação de jogo deve ser reusada como solução para a atual situação de jogo. Esse modelo de reuso em dois passos foi implementado em conjunto com diferentes combinações de critérios de reuso onde as ações e clusters foram escolhidos por meio do reuso da maioria, escolha da ação / cluster por métodos aleatórios onde as probabilidades de cada ação ser sorteadas são previamente computadas, ações / clusters com maior probabilidade de vitória, ou escolha do cluster ou grupo que proporciona maior ganho de pontos. Para avaliar essas propostas, agentes virtuais implementados com diferentes políticas de reuso foram submetidos a diferentes testes. Para a completa validação da proposta, este trabalho apresenta as abordagens utilizadas em todas as etapas do ciclo CBR (recuperação, reuso, revisão, retenção).

Palavras-chave: Políticas de Reuso. Raciocínio Baseado em Casos. Clustering. Jogos de cartas. Jogo de Truco.

ABSTRACT

Cases and Clusters applied at the development of Reuse Policies for Decision-Making in Card Games

AUTHOR: Gustavo Bathu Paulus
ADVISOR: Luís Alvaro de Lima Silva

This work investigates combinations of cases and clusters that use reusable gaming actions present in retrieved cases from queries made by virtual players. These queries are built using case-based reasoning (CBR) designed for playing cards action and bet actions. With the support of the K-MEANS data clustering algorithm, the past experiences are organized in groups where the combination of game-state and game-actions is considered. Initially, a criterion is applied to choose the group to be accessed; then, another reuse criterion uses the cases that belong to the selected cluster to determine which game-action should be reused as a solution to the current game situation. This two-step reuse model was implemented with different combinations of criteria in different steps. The criteria explored in this work are identified by majority rules reuse, choice of action/cluster by random methods where the probabilities of each action being classified are considered, actions/clusters most likely to win, or choice of cluster or group that provides the most points. To evaluate these combinations, the virtual agents were implemented with different reuse policies and were used in different tests. To validate the agents, this work shows approaches to all stages of the CBR cycle (retrieval, reuse, revise, retain).

Keywords: Reuse policies. Case Based Reasoning. Clustering. Card-Games. Truco-Game.

LISTA DE ILUSTRAÇÕES

FIGURAS

TABELAS

ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

ACPC	*Annual computer poker competition*
API	*Application programming interface*
CBR	*Case Base Reasoning*
IA	*Inteligência Artificial*
IBM	*International Business Machines*
K – NN	*K – Nearest Neighbor*
LDA	*Linear discriminant analysis*
MJ	*Majority rules*
MJC	*Majority rules with cluster*
MJCNPS	*Majority rules with cluster number points solution*
MJCPLS	*Majority rules with cluster probability lottery solution*
MJCPVS	*Majority rules with cluster probability victory solution*
MJCS	*Majority with cluster solution*
MJS	*Majority rules solution*
NP	*Number of points*
NPC	*Number of points with cluster*
NPCMJS	*Number of points with cluster majority rules solution*
NPCPLS	*Number of points with cluster probability lottery solution*
NPCS	*Number of points with cluster solution*
NPCPVS	*Number of points with cluster probability victory solution*
NPS	*Number of points solution*
PAM	*Partition around medoids*
PL	*Probability lottery*
PLC	*Probability lottery with cluster*
PLCMJS	*Probability lottery with cluster majority rules solution*
PLCNPS	*Probability lottery with cluster number points solution*
PLCPVS	*Probability lottery with cluster probability victory solution*
PLCS	*Probability lottery with cluster solution*
PLS	*Probability lottery solution*
PV	*Probability victory*
PVC	*Probability victory with cluster*
PVCMJS	*Probability victory with cluster majority rules solution*
PVCNPS	*Probability victory with cluster number points solution*

PVCPLS	*Probability victory with cluster probability lottery solution*
PVCS	*Probability victory with cluster solution*
PVS	*Probability victory solution*
RTS	*Real-time strategy*
WCSS	*Within-cluster sum of squares*

SUMÁRIO

1. INTRODUÇÃO

Jogos digitais continuam apresentando excelentes campos experimentais para a construção, teste e consequente evolução de técnicas de Inteligência Artificial - IA. Além de serem fáceis de entender, os problemas encontrados nestes jogos são muitas vezes similares a problemas presentes no mundo real. Prova disso é que, historicamente, diversas contribuições inicialmente propostas em cenários de jogos digitais possibilitaram o desenvolvimento de novas técnicas inteligentes, as quais ainda têm se mostrado efetivas na solução de problemas reais. Por exemplo, um sistema desenvolvido pela IBM inicialmente voltado para jogos hoje é utilizado para auxiliar na resolução de problemas em diferentes áreas como medicina e inteligência de negócios (Markoff, 2011) (Balan e Otto, 2017).

No cenário de jogos de cartas, conforme (Niklaus *et al.*, 2019), características como informações não explicitamente apresentadas (ou escondidas) associadas a aspectos estocásticos possibilitam simular realidades apresentadas em outras áreas de aplicação. Além disso, jogos com essas características apresentam grandes desafios de pesquisa para a área de IA aplicada (Sandven e Tessem, 2006; Luís Filipe e Reis, 2011; Backhus *et al.*, 2013; Teófilo e Reis, 2013; Ambekar *et al.*, 2015; Ventos *et al.*, 2017; Brown e Sandholm, 2019). Tecnicamente falando, conforme (Rubin e Watson, 2012), a distribuição aleatória e a visão parcial das cartas dos oponentes dificultam a criação de árvores de jogo, enfoque bastante explorado em diferentes tipos de aplicações de IA. Isso indica que situações reais de tomada de decisão são comuns nestes jogos, onde decisões precisam ser tomadas com base em estados de jogo com informações incompletas. Conforme descrito em (Canellas *et al.*, 2015) e em (Brown e Sandholm, 2019), é contínuo o desafio de criar técnicas de IA capazes de proporcionar boas decisões nestes tipos de problemas.

Em geral, técnicas de IA fundamentadas na análise de dados de jogos têm se mostrado promissoras na solução de diferentes problemas. Assim como investigado nesta dissertação, dados oriundos de logs de jogos de cartas gerados pelos mais variados perfis de jogadores, são capazes de guiar a tomada de decisão em novos problemas (Hooshyar *et al.*, 2018). Em uma abordagem de *lazy learning*, este trabalho explora esses dados de partidas de jogos disputadas entre diferentes tipos de jogadores via técnicas de Raciocínio Baseado em Casos (*Case-Based Reasoning* - CBR) (Lopez De Mantaras *et al.*, 2005). Entre outros motivos, CBR é uma técnica bastante consolidada em IA que demonstra como sistematicamente reutilizar experiências concretas de solução de problemas (ou casos) no apoio a tomada de decisão em novas situações. Além do mais, as soluções tomadas por sistemas CBR podem ser facilmente explicadas pelos casos utilizados para as decisões.

Nesta dissertação e em outros trabalhos da literatura (Sandven e Tessem, 2006; Rubin e Watson, 2007; Watson e Rubin, 2008; Rubin e Watson, 2009; Rubin e Watson, 2010; Rubin e Watson, 2012), mãos de jogos de cartas representadas e armazenadas no formato de casos, em bases de casos são utilizadas de diferentes formas para auxiliar na tomada de decisão em novos problemas. Para isso, políticas de reuso de soluções, são usadas na seleção e reuso de soluções gravadas em casos recuperados para consultas emitidas em sistemas CBR. Em (Rubin e Watson, 2010), por exemplo, soluções gravadas em casos passados são reusadas de três diferentes maneiras no jogo de Poker Texas Hold'em: 1) através do reuso da ação de jogo proposta pela maioria dos casos recuperados; 2) por meio de sorteio entre as ações de jogo contidas nos casos

recuperados onde as probabilidades de cada ação ser sorteada são previamente computadas e 3) pelo reuso de ações de jogo que proporcionaram melhores consequências (levaram a vitória, por exemplo) aos casos similares recuperados. Apesar da apresentação de agentes competitivos construídos a partir dessas políticas, tal como descrito em (Rubin e Watson, 2010), ainda há espaço para o desenvolvimento de novas técnicas de reuso de soluções neste domínio de problemas. Em particular, a forma com que essas políticas de reuso tradicionais de CBR são construídas, indica que o reuso de soluções passadas (por exemplo, ações de jogo) na resolução de novos problemas (novas jogadas em uma partida sendo disputada) se concentram nas soluções gravadas nos casos recuperados (Rubin e Watson, 2010). Como as características desses casos podem ser analisadas de diferentes formas para permitir organizar estes casos em diferentes grupos (ou clusters), estas políticas convencionais muitas vezes falham na consideração dos diferentes estados de jogo (por exemplo, início de uma rodada ou final de uma rodada) em que essas ações de jogo (por exemplo, a carta jogada ou a aposta realizada) foram tomadas. Na prática, o conhecimento capturado por pares (estados – ações) de jogo pode ser melhor utilizado em tarefas de tomada de decisão quando esses grupos de casos formados são explorados por tais políticas de reuso. De acordo com trabalhos voltados para a descoberta de conhecimento em logs de jogos (Gow *et al.*, 2012; Sifa *et al.*, 2013; Teófilo e Reis, 2013; Bauckhage *et al.*, 2015; Wehr e Denzinger, 2015; Lora *et al.*, 2016; Ohira *et al.*, 2016), é preciso analisar como casos armazenados em bases de casos podem ser organizados em grupos significativos de forma a identificar ações de jogo juntamente com os estados de jogo em que essas ações são executadas.

Para atacar esse problema, esta dissertação explora a identificação de padrões de jogo, encontrados em diferentes cenários de decisão existentes em logs de jogos gravados em bases de casos, as quais são geralmente formadas para orientar a tomada de decisão em sistemas CBR. Em uma área de pesquisa resultante da integração de técnicas de CBR e técnicas de agrupamento de dados (Li *et al.*, 2014; Khussainova *et al.*, 2015; Chen *et al.*, 2018; Lucca *et al.*, 2018), tal identificação de padrões de jogos é realizada a partir da exploração de algoritmos de *clustering (Bauckhage et al., 2015) (Jain et al., 1999).* Assim como demonstrado nesta dissertação, resultados de múltiplas execuções do algoritmo K – MEANS (Fukunaga e Narendra, 1975; Jain *et al.*, 1999) são explorados na análise e organização de casos contidos em bases de casos, e consequente utilização dos agrupamentos de casos formados na computação de novas políticas de reuso para sistemas CBR, as quais são usadas na implementação de agentes virtuais aplicados a jogos de cartas. Assim como investigado em trabalhos voltados para a descoberta de conhecimento (mineração de dados) em logs de jogos (Gow *et al.*, 2012; Sifa *et al.*, 2013; Teófilo e Reis, 2013; Bauckhage *et al.*, 2015; Wehr e Denzinger, 2015; Lora *et al.*, 2016; Ohira *et al.*, 2016), esse trabalho detalha como grupos de casos contidos na base de casos podem ser utilizados com a finalidade de construir políticas de reuso capazes de identificar os diferentes contextos de decisões de jogo existentes (ou táticas de jogo) nos casos recuperados para consultas emitidas em sistemas CBR. Nesse cenário, casos recuperados para uma consulta podem pertencer/representar diferentes estados de decisão no contexto de um jogo. Para atacar esse problema, grupos criados por algoritmos de *clustering* evitam que diferentes táticas de jogos sejam computadas de forma plana/uniforme (sem que sejam identificadas como táticas distintas) pelos critérios de reuso usados em sistemas CBR. Nesta dissertação, por exemplo, o algoritmo K-MEANS é utilizado para identificar diferentes conjuntos de pares estado-solução para os diferentes cenários de decisão existentes em um jogo denominado TRUCO (Winne, 2017). Em particular, o trabalho prático apresentado

nesta dissertação foi desenvolvido para apoiar a tomada de decisão neste jogo de cartas, o qual é muito popular nas regiões do sul da América do Sul.

Resultados preliminares obtidos nesta dissertação (Paulus *et al.*, 2019) demonstram que as novas políticas de reuso desenvolvidas podem ser melhoradas quando resultados de algoritmos de *clustering* são explorados. Tais resultados são inseridos em um modelo de reuso de soluções em dois passos explorado por um sistema CBR. O primeiro passo deste modelo envolve a recuperação dos casos mais similares para uma consulta dada. Neste passo, um critério de reuso é aplicado inicialmente para escolher um grupo dentre os grupos representados/presentes nos casos recuperados (este critério é identificado como critério de reuso extra cluster). Assim que um grupo é selecionado, os casos similares que pertencem ao grupo escolhido são filtrados. Então, o segundo passo deste modelo de reuso envolve aplicar um critério de reuso para escolher a solução presente nos casos que pertencem ao grupo escolhido no primeiro passo (este critério é identificado como critério de reuso intra cluster). Desta forma, esse modelo de reuso em dois passos é empregado na proposta e teste de novas políticas de reuso. Esta dissertação amplia significativamente o número de políticas de reuso apresentadas em (Paulus *et al.*, 2019), visto que diferentes combinações entre critérios de reuso podem ser utilizadas para seleção do cluster (critério de reuso extra cluster) e escolha da solução (critério de reuso intra cluster). Nesta investigação, os critérios utilizados para a proposição de novas políticas de reuso são:

a) Votação da maioria - *majority rules* - MJ: a escolha de clusters ou ações de jogo é apoiada pelo cluster/solução presente na maioria dos casos similares recuperados;
b) Sorteio baseado em probabilidade - *probability lottery* - PL: a escolha de clusters ou ações de jogo é proveniente de uma função de sorteio dependente de estimativas de probabilidade calculadas a partir de ações de jogo ou clusters presentes nos casos recuperados;
c) Probabilidade de vitória - *probability victory* - PV: o cálculo das chances de vitória para cada uma das diferentes ações de jogo registradas nos casos recuperados orienta a escolha dos clusters ou ações de jogo.
d) Quantidade de pontos ganhos – *number of points won* - NP: a quantidade de pontos conquistados (no jogo de Truco, pontos são disputados nas rodadas de jogo) apoia a escolha de clusters ou ações de jogo;

Em particular, esta dissertação propõe políticas de reuso resultantes da combinação destes diferentes critérios de reuso. Além disso, a dissertação detalha a implementação de uma técnica de aprendizado automático associada ao modelo que integra casos e clusters no processo de reuso de decisões passadas em CBR. Em resumo, a dissertação explora a combinação de algoritmos de *clustering* e CBR para criar agentes virtuais particularmente capazes de jogar Truco. A partir dessa pesquisa, políticas de reuso melhoradas através da combinação destes algoritmos são propostas e, por fim, além de validar tais políticas de reuso, apresenta uma análise do desempenho dos agentes jogadores de Truco construídos.

O trabalho está organizado da seguinte forma:

O Capítulo 2 apresenta o referencial teórico necessário para a construção e compreensão do trabalho desenvolvido. Neste Capítulo, além de explanar as técnicas exploradas e trabalhos relacionados que sustentam a relevância da pesquisa proposta. Também introduzimos os desafios apresentados por jogos de cartas onde não existe visão total dos estados de decisões.

O Capítulo 3 apresenta a abordagem utilizada na identificação dos comportamentos de jogo identificáveis a partir da análise de uma base de casos, representação de casos utilizada e modelo de recuperação de casos explorados na dissertação. Neste Capítulo, a organização da base de casos e o modelo utilizado para a etapa de recuperação são discutidos em detalhe.

O Capítulo 4 apresenta as abordagens propostas para a utilização das informações contidas nos casos mais similares para a solução de novos problemas. Neste Capítulo, as abordagens utilizadas para as etapas de reuso, revisão e retenção de casos são discutidas em detalhe.

O Capítulo 5 apresenta e discute os resultados de diferentes experimentos realizados. Além de introduzir a metodologia de testes utilizada, resultados de diferentes conjuntos de testes são discutidos individualmente.

O Capítulo 6 conclui o trabalho, além de apresentar sugestões de trabalhos futuros.

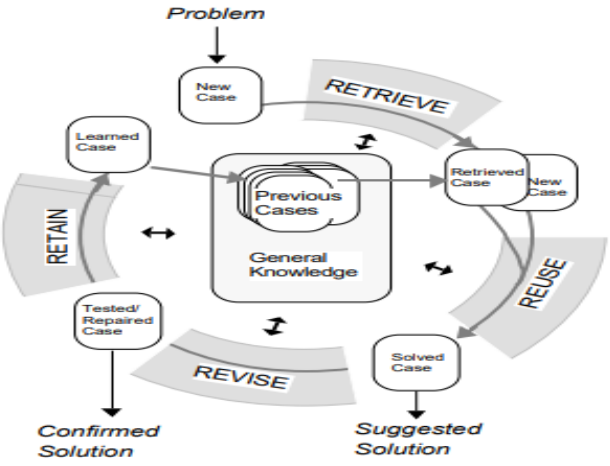
2. REVISÃO BIBLIOGRÁFICA

Este trabalho apresenta um modelo de decisão baseado em dados, para desafios proporcionados por jogos, onde decisões precisam ser tomadas com visão parcial do estado do jogo. Nossa pesquisa propõe um modelo de decisão híbrido através da integração de CBR com algoritmos de *clustering*. Nessa sessão, são apresentas as técnicas que dão suporte para as soluções propostas.

2.1. RACIOCÍNIO BASEADO EM CASOS

CBR permite que soluções utilizadas em experiências de solução de problemas passados sejam reaplicadas na solução de novos problemas. Em sistemas CBR, as coleções onde tais experiências ficam armazenadas em um formato de "casos" são conhecidas como base de casos. De acordo com (Richter e Weber, 2013), cada caso é composto por uma série de atributos capazes de representar tais experiências. Para permitir que experiências de solução de problemas sejam utilizadas para propor soluções em novos problemas, diferentes etapas de raciocínio podem ser realizadas. Em sistemas CBR, estas etapas compõem o ciclo CBR (Lopez De Mantaras *et al.*, 2005), o qual pode ser dividido em: recuperação, reuso, revisão e retenção. Figura 1 apresenta o Ciclo de CBR.

Figura 1 - Ciclo de Cbr



Fonte: (Aamodt e Plaza, 1994)

A etapa de recuperação é direcionada para a computação de similaridade e consequente determinação dos casos mais semelhantes ao problema apresentado ao sistema por meio de uma consulta. O algoritmo K – NN (k – *nearest neighbor*) (Fukunaga e Narendra, 1975) é frequentemente utilizado na recuperação dos casos mais similares armazenados na base de casos, onde essa recuperação é realizada através da utilização de funções de similaridade. Funções de similaridade podem ser divididas em local e global. Por meio das funções de similaridade locais é computada à similaridade em cada atributo usado na representação dos casos, com computações de similaridades

variadas de acordo com o tipo do atributo e seu objetivo na aplicação. Por isso, existe uma grande variedade de funções de similaridade locais com diferentes propostas. A função de similaridade global utiliza os resultados computados pelas similaridades locais para comparar a semelhança entre a consulta e os casos da base de casos de maneira ampla. A função de similaridade global utilizada para recuperar os casos na etapa recuperação dos algoritmos de CBR desenvolvidos nesta dissertação é descrita pela Equação (1):

$$G(q,c) = \sum_{f \in A} \frac{sim_f(q_f, c_f)}{|A|}$$

Na Equação (1), f representa a computação de similaridade local, q identifica a *query*, e A identifica a lista de atributos que são utilizados para a computação de similaridade global. Neste caso, |A| representa a quantidade de atributos utilizados na consulta. O resultado da similaridade global entre os casos é dado pela soma dos resultados das similaridades locais entre o caso de consulta e o caso armazenado na base de casos dividido pelo número de atributos utilizados nestas computações.

Uma maneira de computar a similaridade local entre atributos com valores categóricos (numéricos ou não) é simplesmente utilizar uma função *equal*. Outro exemplo de computação de similaridade local é dado pelo uso de uma função *interval (Recio-García et al., 2014)*, apresentada na Equação (2). Essa função permite calcular a similaridade entre atributos com valores numéricos através da subtração do atributo Qi do caso da consulta em relação ao atributo Ci do caso armazenado na base de casos. O resultado é normalizado pelo valor máximo do intervalo de possibilidades para o referido atributo e posteriormente subtraído de 1.

$$S_i = 1 - \left({}^{|Q_i - C_i|}/_{MAX\_VALUE} \right)$$

Uma vez que uma lista contendo os casos mais similares é recuperada da base de casos, esses casos devem ser utilizados para a escolha de uma solução para um problema corrente. Esta etapa seria extremamente simples se fosse possível construir uma base de casos que se armazena casos para todos os tipos de problemas no domínio de aplicação sendo atacado. Contudo, nem sempre é possível tomar a melhor decisão pelo reuso de soluções recuperadas a partir da análise de apenas um caso mais similar recuperado da base de casos, visto que são raros os problemas onde é possível encontrar um problema a ser resolvido que seja muito similar (idêntico) a um caso passado.

Além de ser responsável por escolher a solução que será utilizada na resolução de novos problemas, a etapa de reuso também adapta as soluções contidas nos casos recuperados para atender a necessidades de solução do novo problema. Por este motivo, podemos dizer que a etapa de reuso é composta por duas sub–tarefas: escolha da solução (responsabilidade das políticas de reuso) e adaptação da solução ao problema atual. Por exemplo, em um jogo de cartas, as cartas recebidas não necessariamente são as mesmas contidas nos casos similares recuperados. Quando existe a necessidade de escolher uma carta entre as cartas de uma mão corrente para jogar, essa adaptação deve ajustar as cartas jogadas indicadas nos casos similares recuperados para as cartas contidas na mão atual. De acordo com (Richter e Weber, 2013), um processo de adaptação pode iniciar durante a execução da política de reuso e concluir apenas no momento de retornar uma solução para o problema dado.

Diferentes critérios podem ser aplicados na escolha da solução a ser utilizada no novo problema. Neste caso, a etapa de reuso aplica critérios de seleção de soluções conforme políticas de reuso determinadas. É importante ressaltar que diferentes critérios utilizados nestas políticas podem resultar na obtenção de diferentes soluções a partir do mesmo conjunto de casos recuperados para uma consulta. Basicamente, é responsabilidade da etapa de reuso a escolha, com base nos casos recuperados, da solução mais relevante para o problema atual.

Como nesta dissertação propomos um modelo de reuso de soluções em dois passos, algumas definições se fazem necessárias:

- Um critério de reuso é utilizado para definir como a solução (ou um cluster) é selecionado (reuso da maioria, probabilidade loteria, quantidade de pontos, probabilidade vitória).
- Um modelo de reuso é utilizado para identificar a maneira que os critérios de reuso são aplicados (reuso em dois passos ou reuso convencional).

Neste trabalho, portanto, uma política de reuso é composta pela aplicação de critérios de reuso em um dos dois modelos de reuso utilizados.

Após a etapa de reuso ser concluída, a solução reusada é submetida à etapa de revisão. Na etapa de revisão é verificado se as soluções propostas são realmente eficazes para resolver o problema em que foram aplicadas. A etapa de revisão em CBR geralmente utiliza conhecimentos extras aos existentes nos casos para validar se a solução proposta pela etapa de reuso está correta. Apesar de utilizar conhecimentos extras aos casos, essa etapa pode ser feita de forma automatizada por meio da coleção e representação de regras de revisão (Unger, 2011). De acordo com (Richter e Weber, 2013), a etapa de revisão tem o objetivo de melhorar a qualidade das soluções propostas com base em experiências não disponíveis nos casos.

Após a solução proposta pela etapa de reuso ter sido submetida à etapa de revisão, esta nova experiência de solução de problemas pode ser armazenada na base de casos. Em geral, a construção de agentes inteligentes necessita que esse processo de aprendizado seja feito de forma continua, principalmente quando a base de casos inicial não tem uma boa cobertura dos tipos de problemas que o agente deve resolver. Em CBR, uma das principais etapas responsáveis por esse aprendizado é a etapa de retenção, a qual deve possuir critérios que levem a construção de bases de casos com qualidade, contendo uma grande cobertura de problemas e soluções. Em sistemas CBR, portanto, a tarefa de aprendizagem consiste em adicionar novas experiências de solução de problemas à base de casos (Richter e Weber, 2013). Por meio da exploração das bases de casos, políticas de reuso de soluções são capazes de propor soluções para novos problemas. Dessa forma, quanto maior a cobertura da base de casos, melhores podem ser as soluções propostas pelas políticas de reuso dos sistemas CBR.

Imitar soluções recuperadas de experiências de solução de problemas passados, é uma das principais características de sistemas CBR. Em geral, a retenção de experiências criadas a partir da execução do ciclo CBR descrito pode ser caracterizada como um tipo de aprendizagem por imitação. Essa forma de aprendizagem é colocada em prática quando as quatro etapas do ciclo CBR são implementadas (recuperação, reuso, revisão, retenção) (Ji *et al.*, 2018).

2.2. ALGORITMOS DE CLUSTERING

Clustering é uma técnica de aprendizado não-supervisionado capaz de identificar grupos em um determinado conjunto de dados. Estes algoritmos têm como principal objetivo investigar como organizar dados em grupos por meio da utilização de características naturais dos dados (Jain *et al.*, 1999).

Os algoritmos de *clustering* têm sido utilizados para uma vasta quantidade de situações. Os elementos que estão contidos em um mesmo grupo devem ser similares entre si e dissimilares dos que estão fora do grupo. Existe um grande número de paradigmas de algoritmos *clustering* com diferentes propostas encontradas na literatura (Hartigan, 1975).

Em geral, resultados de algoritmos de *clustering* são capazes de revelar informações muitas vezes implícitas na base de casos, onde tais informações podem ser exploradas na construção e aperfeiçoamento de funções de decisão de sistemas CBR. A aplicação de algoritmos de *clustering* em bases de casos permite a identificação dos grupos de casos existentes, e isso pode resultar em melhorias para as etapas do ciclo CBR. Por exemplo, a etapa de recuperação pode utilizar resultados de algoritmos de *clustering* para reduzir o custo computacional de consultas (Chen *et al.*, 2018), evitar que soluções equivocadas sejam utilizadas em recomendações (Khussainova *et al.*, 2015), ajustar os pesos de atributos em funções de similaridade (Lucca *et al.*, 2018), ou ainda para auxiliar na execução de atividades de manutenção de bases de casos (Smiti e Elouedi, 2014; Ayed *et al.*, 2017).

2.2.1. ALGORITMO K – MEANS

Diferentes algoritmos de *clustering* podem ser explorados no desenvolvimento de sistemas CBR. Por exemplo, em (Lucca *et al.*, 2018), instâncias de algoritmos particionais (incluindo o algoritmo K-MEANS), foram explorados; em (Chen *et al.*, 2018), o algoritmo K – MEANS foi explorado. De acordo com (Bauckhage *et al.*, 2015), o algoritmo K - MEANS tem sido explorado com maior frequência na análise de logs de jogos.

O algoritmo K - *MEANS* permite a criação de agrupamentos completos no qual todos os elementos são agrupados. Trata-se de um algoritmo particional baseado em centroides, onde como parâmetro é necessário informar o número de grupos K existentes nos dados. Este algoritmo utiliza o modelo de centroides no qual a noção de similaridade tem como medida a distância entre as instâncias e o centroide do grupo.

Por padrão a distância Euclidiana é utilizada na computação de similaridade, assim como apresentado na Equação (3):

$$d(Q, C) = \sqrt{\sum_{i}^{p} (q_{\mathrm{i}} - c_{\mathrm{i}})^2}$$

O algoritmo K – MEANS pode ser dividido em três passos, sendo o primeiro escolher os centroides (na implementação padrão são definidos aleatoriamente). Posteriormente, é calculada a distância do centroide em relação a cada um dos elementos. Por último, é feita a média de todos os pontos ligados ao centroide. Caso a média tenha alteração, um novo centroide é definido e é novamente necessário calcular a distância de todos os elementos em relação ao centroide até não existir mais alterações.

Diferentes critérios podem ser utilizados para auxiliar na identificação do número de grupos existentes em um dado conjunto de dados. Dentre as opções, o método *Elbow* apresenta-se como uma ferramenta para auxiliar nessa tarefa (Bholowalia e Kumar, 2014). Este método realiza testes com diferentes configurações no parâmetro K. Como resultado, um gráfico com diferentes resultados para cada número K testado pode ser apresentado. O gráfico *Elbow* pode ser utilizado para analisar a soma das distâncias dos itens de cada grupo com os respectivos centroides. Isso possibilita identificar o quão disperso estão os dados em relação aos centroides para cada um dos valores K testados. A computação das distâncias pode ser feita através da utilização do algoritmo WCSS (*within cluster sum of squares*) (Gow *et al.*, 2012). O algoritmo WCSS utilizado pelo método de *Elbow* pode ser representado pela Equação (4), onde P representa os pontos (instâncias) de dados existentes e C identifica os centroides dos grupos. Na equação (4) é apresentado um exemplo de uma configuração de dois grupos:

$$WCSS = \sum_{\mathrm{P_i} \in Cluster\ 1} \text{distância}(\mathrm{P_i}, \mathrm{C_i})^2 + \sum_{\mathrm{P_i} \in Cluster\ 2} \text{distância}(\mathrm{P_i}, \mathrm{C_i})^2$$

Este algoritmo pode ser dividido em quatro passos:

1. Computar as distâncias de cada elemento em relação ao centroide do grupo em que estão alocados.

2. Elevar ao quadrado os resultados das distâncias de cada um dos elementos em relação ao centroide do grupo que pertence.

3. Somar o quadrado das distâncias de todos os pontos de um dado cluster.

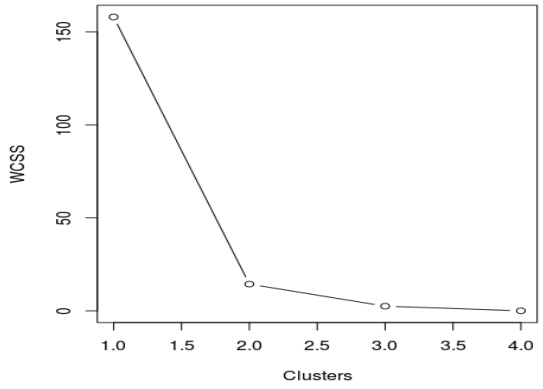
4. Somar os resultados de todos os clusters.

A tomada de decisão do número de grupos a ser usado com base no método de *Elbow* geralmente é feita através da análise de um gráfico. Neste gráfico, o eixo Y representa o resultado do WCSS e o eixo X o parâmetro K testado. Quanto maior forem os resultados do WCSS para cada uma das configurações de K testadas, mais dispersos estão os dados em relação aos centroides dos grupos que cada instância pertence. Porém, o excesso de grupos pode causar a criação de grupos desnecessários e dificultar a análise dos grupos de comportamentos existentes nos dados.

Geralmente, o número K utilizado pelo método *Elbow* é onde o resultado WCSS tem uma queda brusca em relação à configuração do K anterior e estabiliza no K

posterior. O nome do método é originado do fato de que o valor K (que geralmente é utilizado) estar em um ponto que comumente gera um formato de cotovelo causado pela queda do WCSS em relação ao resultado do WCSS do K anterior e estabilidade no K posterior. Como apresentado na imagem onde o valor K sugerido é 2:

Figura 2 - Exemplo do *plot Elbow*



Fonte: do autor.

Por fim, em (Paulus *et al.*, 2019), o qual descreve resultados preliminares obtidos na pesquisa desenvolvida nesta dissertação, foi introduzido um modelo de reuso de soluções em dois passos para sistemas CBR. Neste modelo, resultados de algoritmos de *clustering* possibilitaram a melhoria da acurácia da etapa de reuso de soluções utilizada por agentes CBR capazes de jogar Truco. Apesar destes resultados, de acordo com (Rubin e Watson, 2012), não são apenas as políticas de reuso que interferem no desempenho de agentes CBR aplicados a esses tipos de cenário de decisão. A partir disso, uma investigação sobre políticas de reuso, algoritmos de *clustering* aplicado em logs de jogos, integração entre CBR e *clustering* e aprendizagem de casos automática é desenvolvida nesta dissertação (ver trabalhos relacionados).

2.3. TRABALHOS RELACIONADOS

Esta seção revisa trabalhos relacionados sobre cada um dos principais tópicos da pesquisa desenvolvida nesta dissertação. A seção 2.3.1 revisa pesquisas de IA aplicadas a jogos de cartas. A seção 2.3.2 revisa trabalhos sobre políticas de reuso investigadas em aplicações similares a aplicação explorada neste trabalho. A seção 2.3.3 explora

pesquisas onde algoritmos de *clustering* são aplicados em dados de jogos. A revisão de trabalhos relacionados a integração entre CBR e *clustering* é realizada na seção 2.3.4. Devido à dificuldade de coletar casos em algumas aplicações, técnicas que permitem o aprendizado automático de casos foram necessárias no desenvolvimento deste trabalho. Por isso, a seção 2.3.5 investiga critérios da aprendizagem automática em sistemas CBR.

2.3.1. IA EM JOGOS DE CARTAS

A maioria dos jogos de cartas envolve uma alternância de propriedade de cartas entre os jogadores. Ao mesmo tempo, existem estados de jogo nos quais não se têm conhecimento das cartas que os oponentes possuem. Isso faz com que esse tipo de jogo seja desafiador visto que os jogadores devem ter a capacidade de inferir o estado do jogo por meio de informações incompletas (Yannakakis e Togelius, 2018). Além da visão parcial e elementos de chance, aspectos psicológicos e emocionais fazem de alguns jogos de cartas um relevante domínio para o avanço de técnicas de IA (Rubin e Watson, 2011; Brown e Sandholm, 2019).

Diferentes tipos de jogos têm sido utilizados em pesquisas de IA. Inicialmente, os esforços dos pesquisadores eram voltados a jogos onde existe visão total do jogo, tais como: xadrez, jogo de damas, gamão. Com a criação de agentes capazes de obter bons desempenhos nesse tipo de jogos, cenários de decisões onde existem informações imperfeitas têm se tornado alvo de pesquisas recentes (Ventos *et al.*, 2017) (Brown e Sandholm, 2019). A grande maioria dos jogos de cartas compartilham benefícios em comum, porém as diferentes particularidades existentes podem proporcionar contribuições distintas. Por exemplo, em (Ventos *et al.*, 2017) o jogo de Bridge foi investigado. Assim como o Truco, Bridge conta com diferentes etapas de decisões as quais permitem o desenvolvimento de técnicas onde habilidades distintas são exigidas. Isso é diferente do Poker Texas Hold'em, onde as ações de jogo exigem unicamente a capacidade de realizar apostas.

Diferentes abordagens são utilizadas na tentativa de criar técnicas de IA capazes de proporcionar boas decisões nesse tipo de ambiente. Em (Niklaus *et al.*, 2019), foi apresentada uma revisão de diferentes técnicas utilizadas para implementar a capacidade de decisão de agentes virtuais em jogos de cartas. De acordo com este trabalho, quatro métodos principais são geralmente utilizados para criar agentes autônomos em jogos de cartas, sendo eles:

- Métodos baseados em regras: nessa modalidade, são estipulados comportamentos para diferentes estados do jogo. Trata-se de uma abordagem que depende exclusivamente de um especialista de aplicação para a aquisição de conhecimento;
- Métodos de aprendizagem por reforço: esses algoritmos utilizam recompensas para incentivar comportamentos benéficos e penalizações para penalizar ações incoerentes;
- Métodos Monte Carlo: apesar do grande número de possibilidades, jogos de cartas têm um número finito de combinações. Por isso, métodos dessa classe de

métodos utilizam a aleatoriedade para resolver problemas que podem ser determinísticos;

- Algoritmos evolucionários: utilizam a teoria de que ações fortes podem se reproduzir enquanto as fracas tendem a serem extintas.

Além dessas abordagens, (Rubin e Watson, 2011) apresentou técnicas alternativas, onde CBR é citado como uma abordagem viável para realizar ações de apostas no jogo de Poker. Em (Rubin e Watson, 2012), por exemplo, foram apresentados características vitais para evoluir o desempenho de sistemas CBR aplicados ao jogo de Poker, dentre as quais estão:

- Representação de características;
- Métricas de similaridade;
- Representação de solução;
- Recuperação de casos;
- Políticas de reuso de soluções;
- Treinamento do sistema.

Em (Rubin e Watson, 2012), foram comparadas três diferentes abordagens para a criação do atributo responsável por representar a força de uma mão de jogo. Por possuir um atributo responsável por representar a sequência de apostas, uma função de similaridade ajustada para esse atributo foi construída, já que a utilização da similaridade local *equal* para esse atributo, apenas computaria similaridade quando a sequência de apostas fossem exatamente as mesmas. Como os autores tiveram acesso a uma grande quantidade de casos de jogos de Poker, com várias mãos repetidas, armazenar diferentes variações de soluções para uma mesma mão fez com que a base obtivesse proporções gigantescas com alto custo computacional. Dessa forma, foi criado um atributo vetor no modelo de representação de casos para representar todas as variações de soluções obtidas para uma mesma mão de jogo e outro para representar as consequências que cada uma das diferentes ações proporcionou. Já que um mesmo caso poderia conter diferentes soluções, os autores optaram por recuperar apenas o caso mais similar. Para escolher a ação que deveria ser feita no jogo, os resultados de três políticas de reuso apresentadas em (Rubin e Watson, 2010) foram utilizados. Por último, este trabalho descreve como a base de casos foi construída. De acordo com os autores, a base deveria conter, na medida do possível, informações completas. Por isso, grande quantidade de logs de jogos de Poker obtidos na competição ACPC (*Anual Computer Poker Competition*) foram utilizados no desenvolvimento do trabalho. Dessa forma, a base de casos utilizada ainda está sendo incrementada todo o ano, o que permite a evolução contínua dos agentes desenvolvidos. Utilizar logs de jogos para a construção de técnicas de IA para jogos de cartas não é uma exclusividade de pesquisas onde técnicas de CBR são utilizadas. Utilizando logs de partidas de Poker, (Luís Filipe e Reis, 2011) e (Ambekar *et al.*, 2015) aplicam mineração de dados em conjunto com diferentes classificadores. Dessa forma, os autores foram capazes de, respectivamente, criar um framework para, dada uma mão de jogo, gerar estratégias de jogo baseadas em comportamentos de jogadores humanos e antecipar a probabilidade de ganhar nesse jogo.

Assim como o jogo de Bridge e o jogo de Truco, o jogo de cartas *Wizard* apresenta diferentes desafios. A criação de um agente virtual para esse jogo foi investigada em (Backhus *et al.*, 2013) por meio da utilização da técnica de aprendizado por reforço. De acordo com os autores, em jogos de cartas é importante que as decisões sejam adaptadas para diferentes situações de jogo. Por isso, o agente por eles proposto foi capaz de utilizar estratégias ajustadas para diferentes estados do jogo. Para permitir

a constante evolução do agente desenvolvido, reforços positivos foram utilizados para incentivar ações de jogo corretas, enquanto que reforços negativos foram aplicados como punição para ações incorretas. Apesar de ser conceitualmente distinto do aprendizado por reforço, CBR também permite que consequências de ações de jogo sejam armazenadas na base de casos, permitindo que agentes virtuais automaticamente evoluam a qualidade de suas ações de jogo.

Em resumo, por se tratar de um cenário desafiador, diferentes trabalhos têm descrito abordagens na tentativa de criar agentes competentes para tomar decisões em jogos de cartas. O objetivo desta dissertação, neste caso, é contribuir para o desenvolvimento desta área de pesquisa.

2.3.2. POLÍTICAS DE REUSO EM SISTEMAS CBR

Diferentes etapas do ciclo CBR podem interferir no desempenho de sistemas CBR. Em particular, (Rubin e Watson, 2010) investigou as etapas de reuso e representação de soluções, onde tais aspectos influenciaram no desempenho de agentes virtuais aplicados ao jogo de Poker. Existe uma variedade de políticas de reuso que podem ser utilizadas para escolher a solução a ser selecionada nos casos recuperados de uma base de casos. Neste caso, (Rubin e Watson, 2010) compara o desempenho de três diferentes políticas de reuso aplicadas ao jogo de Poker:

a) reuso das ações de jogo que proporcionam o maior lucro de acordo com ações de jogo gravadas nos casos recuperados para uma consulta dada;

b) reuso de ações de jogo executadas conforme a maioria das ações gravadas nos casos recuperados para uma consulta dada;

c) ações de jogo escolhidas por meio de uma função de sorteio, onde tal função é baseada em estimativas de probabilidade que são calculadas a partir de ações de jogo presentes nos casos recuperados para uma consulta dada.

Um dos principais objetivos em jogos é tomar decisões que possibilitem os maiores ganhos ou evitar as maiores perdas. Por utilizar experiências de jogos passados, o critério utilizado na seleção de soluções gravadas em casos recuperados como resposta de consultas pode levar a tomada de diferentes decisões em situações de jogo atuais. Por exemplo, a solução (ou ação de jogo) utilizada pela maioria dos casos recuperados reflete a escolha da solução mais comumente utilizada no passado. Porém, a reprodução da solução mais comum pode tornar um jogador/agente previsível. Para contornar esse problema, (Rubin e Watson, 2010) utilizaram uma função de sorteio, onde essa função emprega as proporções de cada ação de jogo executada. Dessa forma, ações tomadas no jogo puderam ser mais imprevisíveis devido ao sorteio realizado. Mais ainda, em algumas situações de jogo, pode ser vantajoso conhecer os benefícios de uma ação de jogo particular a ser executada. Para testar essa ideia, um novo critério de reuso foi proposto, onde ocorre a imitação da ação de jogo que obteve os maiores ganhos.

Entre os critérios de reuso apresentados em (Rubin e Watson, 2010), o reuso da solução proposta pela maioria dos casos recuperados obteve o maior número de vitórias de acordo com testes realizados. Em contraste, a escolha da solução com os maiores ganhos (NP) obteve o menor número de vitórias. Conforme os autores, o melhor

desempenho da solução tomada pela maioria dos casos recuperados estava relacionado com o consenso das ações de jogo tomadas pelos jogadores.

Apesar de não ter sido explorado em (Rubin e Watson, 2010), o modelo de tomada de decisão mais tradicional utilizado em sistemas CBR é baseado no reuso da solução sugerida pelo caso mais similar recuperado. Esse tipo de tomada de decisão pode obter bons resultados quando os casos possuem uma qualidade atestada, tal como demonstrado em (Oh *et al.*, 2017). Neste trabalho, o uso da solução gravada no caso mais similar recuperado de uma base de casos totalmente construída por jogadores profissionais proporcionou bons resultados a um agente CBR desenvolvido para um jogo de RTS (*Real Time Strategy*). Porém, quando os casos contidos em bases de casos não possuírem uma qualidade atestada (situações onde casos são coletados a partir de jogadores com diferentes experiências de jogo, por exemplo), o reuso de uma solução presente no caso mais similar recuperado pode não garantir a execução de uma boa jogada.

Em (Sandven e Tessem, 2006), diferentes atributos foram utilizados para construir heurísticas com o objetivo de maximizar ganhos ou minimizar perdas no jogo de Poker. Neste trabalho, as ações de jogo executadas foram divididas em categorias de estratégias. Inicialmente, a política de reuso foi aplicada apenas para escolher a solução que proporcionou maior lucro dentre as ações de jogo que não faziam parte da estratégia identificada como blefe. Neste caso, blefes apenas foram aplicados quando eles demonstraram ser lucrativos e quando não existiam outras jogadas com resultado (saldo) positivo. Os benefícios que cada ação de jogo proporcionou foram verificados através do cálculo da média do saldo de ganhos e perdas dos casos recuperados que pertenciam a mesma categoria de estratégia. Nesta pesquisa, os autores consideraram apenas casos recuperados que obtiveram similaridade maior que 95% com a consulta dada. Em situações onde casos com essa similaridade mínima não eram recuperados, uma jogada aleatória era executada.

Criar agentes menos previsíveis não é um objetivo apenas associado a jogos com visão parcial. Em (Mozgovoy *et al.*, 2016), CBR foi utilizado para criar agentes capazes de jogar tênis no mesmo nível de jogadores humanos. Dessa forma, para adicionar imprevisibilidade nas decisões tomadas pelos agentes construídos, um critério de reuso que aplica a probabilidade de cada ação de jogo foi utilizado em uma função de sorteio. Similar critério também foi utilizado no jogo de Poker em (Rubin e Watson, 2007; Rubin e Watson, 2010). O uso desse critério de reuso no jogo de tênis foi especialmente útil pelo fato de que os diferentes personagens do jogo possuíam características variadas. Portanto, a forma de comportamento associada a cada um deles poderia acabar criando um padrão que facilitaria os adversários prever a forma de jogar do oponente. Neste caso, utilizar as ações de jogo em conjunto com uma função de sorteio permitiu adicionar imprevisibilidade ao agente, sem desconsiderar a forma de comportamento ideal para cada um dos personagens do jogo.

Nas pesquisas acima citadas, critérios de reuso foram diretamente aplicados na escolha da solução (ações de jogo) a ser usada na construção de agentes (jogadores/bots autônomos). Nesta dissertação, além de adaptarmos e expandirmos os critérios propostos em (Rubin e Watson, 2010), e ampliar a pesquisa apresentada em (Paulus *et al.*, 2019), também propomos novas políticas de reuso por meio da combinação entre diferentes critérios para a) a escolha de grupos de casos formados e b) ações de jogo gravadas em casos recuperados para consultas dadas.

2.3.3. CLUSTERING EM LOGS DE JOGOS

Algoritmos de *clustering* são frequentemente aplicados na identificação de padrões de jogos (Bauckhage *et al.*, 2015). Para isso, estes algoritmos utilizam funções de similaridade para identificar instâncias de dados que são similares entre si e dissimilares dos demais. Por possibilitar o reconhecimento de padrões, a complexidade dos dados pode ser reduzida, o que facilita a identificação de categorias existentes em dados de jogos (Sifa *et al.*, 2013). Conforme citado em (Bauckhage *et al.*, 2015), uma importante contribuição proporcionada por esses algoritmos é o aumento das possibilidades de coleta de dados contextuais relacionados aos comportamentos e táticas de jogo executados e capturados em logs de jogos.

A aplicação de algoritmos de *clustering* em dados de jogos pode ser feita de acordo com diferentes objetivos de análise. Em jogos de estratégia, por exemplo, pode ser importante investigar os grupos que representam sequências de estratégias existentes. Em (Wehr e Denzinger, 2015), o algoritmo de agrupamento de dados PAM (muito similar ao K-MEANS) foi aplicado com o objetivo de identificar sequências de ações que estão relacionadas com o sucesso de agentes aplicados a um jogo de estratégia baseado em turnos denominado *Battle for Wesnoth*. Os autores utilizaram agrupamento de dados para identificar grupos contendo sequências de ações que possibilitaram chegar a um estado vitorioso neste jogo. Em jogos sérios, onde o principal objetivo é desenvolver ou avaliar capacidades relevantes para o treinamento/aprendizado em situações do mundo real, algoritmos de *clustering* podem permitir a identificação de perfis de jogadores. Por exemplo, em (Ohira *et al.*, 2016), habilidades relacionadas a interações sociais são avaliadas através da construção de um ambiente *gameficado*. Após coletar dados de como os usuários interagiram neste ambiente sintético, os algoritmos K - *MEANS* e X – *MEANS* foram explorados para descobrir os perfis de jogadores existentes, visando explorar tais perfis descobertos para tornar mais efetivos os processos de ensino e aprendizagem. Da mesma maneira, grupos que representam comportamentos de jogadores podem ser utilizados para antecipar jogadas (predição de jogadas). (Teófilo e Reis, 2013) descrevem como algoritmos de agrupamento de dados podem ser utilizados para identificar tipos de estratégias presentes em logs de jogos de Poker. Após identificar o grupo que um oponente pertence, é possível utilizar o comportamento do grupo para predizer informações relacionadas ao estado de jogo do adversário. Em alguns casos, a identificação de tipos de comportamentos pode permitir que os jogos se adaptem ao perfil dos jogadores. Por exemplo, com o objetivo de tornar o jogo de Tetris mais cativante e fornecer aos jogadores desafios coerentes com suas habilidades, o algoritmo K - *MEANS* foi utilizado por (Lora *et al.*, 2016) para identificar, a partir de logs de jogos, os tipos de jogadores existentes. Neste caso, os perfis descobertos foram utilizados para adaptar a dificuldade do jogo de acordo com a habilidade de cada jogador. Na prática, após identificar a habilidade de cada jogador, a dificuldade do jogo foi ajustada de acordo com o perfil identificado. Os resultados apresentados comprovam que a utilização dos perfis descobertos pelos algoritmos de *clustering* foram capazes de, na maioria das vezes, proporcionar a correta adaptação do jogo ao nível do jogador.

Alguns algoritmos de *clustering* necessitam que o número de grupos existentes nos dados seja antecipadamente informado. Em (Gow *et al.*, 2012), o algoritmo K – *MEANS* foi aplicado em atributos pré – processados pela técnica de redução de dimensionalidade LDA (*linear discriminant analysis*). Neste trabalho, os autores foram

capazes de identificar diferentes estilos de jogos em dois estudos de casos distintos. Para ser possível a identificação do número de grupos existentes nos dados analisados, o algoritmo WCSS foi aplicado ao *plot elbow*.

Em resumo, a exploração de algoritmos de *clustering* em logs de jogos permite revelar diferentes padrões de comportamentos de jogo. Uma vez que tais padrões de jogos são identificados, essas informações podem contribuir de diferentes formas para a construção de agentes mais efetivos. Tal linha de pesquisa e desenvolvimento é explorada nesta dissertação.

2.3.4. A INTEGRAÇÃO DE CBR E CLUSTERING

O campo de pesquisa emergente da integração entre técnicas de CBR e de agrupamento de dados é principalmente direcionado na proposta de melhorias para a etapa de recuperação e manutenção de bases de casos. Entre outras contribuições, as tarefas de indexação de bases de casos desempenhadas por algoritmos de *clustering* permitem reduzir o tempo de resposta de sistemas CBR, bem como contribuir com o aumento da eficácia de soluções propostas por sistemas CBR.

Em sistemas onde uma grande quantidade de casos precisa ser armazenada em bases de casos, a recuperação dos casos mais similares pode ter um alto custo computacional (devido as computações de similaridades desenvolvidas entre o caso tomado como consulta e os demais casos armazenados na base de casos). Por isso, é comum a utilização de algoritmos de *clustering* para indexar tais bases de casos, e reduzir o número de computações de similaridade desenvolvidas por mecanismos de recuperação de casos. Em (Chen *et al.*, 2018), por exemplo, é descrito um modelo de recuperação em dois passos onde resultados de agrupamento de dados são usados para diminuir o tempo de computação de similaridades. Neste caso, estes algoritmos permitem organizar sub-bases de casos de acordo com os diferentes tipos de problemas presentes nos casos armazenados nestas bases. Nesse modelo, quando uma consulta é proposta, é computada à similaridade do caso atual com o centroide dos grupos de casos formados. Em seguida, a similaridade é medida apenas nos casos que pertencem ao grupo mais similar ao caso usado como consulta. Apesar da recuperação em dois passos ser geralmente utilizada para otimizar o tempo de processamento de consultas, existem pesquisas onde essa técnica é utilizada para aumentar a acurácia das recomendações apresentadas por sistemas CBR. Através da recuperação em dois passos, (Khussainova *et al.*, 2015) demonstram como melhorar a eficácia de um sistema CBR utilizado para

criar planos de tratamento para pacientes com câncer no cérebro. Neste trabalho, a utilização de resultados de algoritmos de *clustering* para indexar a base de casos foi capaz evitar que tratamentos incoerentes fossem sugeridos. Da mesma maneira, algoritmos de *clustering* foram utilizados em (Li *et al.*, 2014) para diminuir a recuperação de casos incorretos em bases contendo um número de problemas desbalanceado, assim viabilizando a construção de um sistema CBR utilizado para prever a falência de empresas. No trabalho em questão, foi aplicada uma estrutura semelhante à de recuperação em dois passos para evitar que a grande diferença entre a quantidade de experiências de sucesso e fracasso gravadas na base de casos interferisse na qualidade das previsões apresentadas. Em particular, essa pesquisa é relevante pelo fato de que pode ser difícil coletar casos onde exista um desbalanceamento entre experiências de sucesso e fracasso em diferentes aplicações de CBR. Por isso, em sistemas onde amostras positivas são mais abundantes, por exemplo, ao utilizar políticas de reuso convencionais (maioria ou caso mais similar, por exemplo), existe uma chance maior de que os casos recuperados representem experiências de solução de problemas que são essencialmente positivas, mesmo considerando que exista uma alta probabilidade da consulta resultar em uma experiência negativa. Por fim, em (Lucca *et al.*, 2018), o modelo de recuperação em dois passos também foi utilizado. Neste caso, a abordagem foi utilizada em uma pesquisa onde diferentes algoritmos de *clustering* viabilizaram a investigação semiautomática da relevância dos atributos usados na representação de casos e consequente construção de funções de similaridade ajustadas para os problemas de aplicação sendo considerados.

Em resumo, é comum nesses trabalhos a exploração de resultados de algoritmos de *clustering* para propor melhores técnicas de recuperação de casos para sistemas CBR. Neste contexto, em (Paulus *et al.*, 2019), foi proposto um modelo de reuso onde os resultados de algoritmos de *clustering* foram explorados na representação de soluções usadas na construção de um modelo de reuso para sistemas CBR. Lá, estes algoritmos foram utilizados na criação de um modelo de reuso em dois passos. O modelo de reuso em dois passos apresentado permitiu que o critério de reuso escolhesse um grupo contendo comportamentos de jogo nos casos recuperados. Após identificar o grupo a ser utilizado, o mesmo critério foi aplicado para a seleção da ação de jogo a ser tomada. Vale ressaltar que o trabalho desenvolvido em (Paulus *et al.*, 2019) permitiu analisar este modelo de reuso em dois passos em uma situação onde poucos casos estavam disponíveis na base de casos. Além disso, o trabalho apresentado em (Paulus *et al.*, 2019) analisou o modelo de reuso proposto por meio da aplicação do mesmo critério de reuso nas duas etapas de reuso propostas. O trabalho aqui apresentado permitiu combinar diferentes critérios de reuso em uma mesma política.

2.3.5. APRENDIZAGEM AUTOMÁTICA EM SISTEMAS CBR

Jogadores virtuais (agentes implementados) tradicionalmente têm suas capacidades de decisão melhoradas por meio da interferência de especialistas de aplicação humanos. Em contraste com esse método tradicional, agentes virtuais têm sido recentemente capazes de atingir desempenhos extraordinários através de treinamentos do tipo *self-play*. Nesse tipo de treinamento, decisões do próprio agente são utilizadas para aprender sem a interferência de jogadores humanos (Silver *et al.*, 2017). Em muitos problemas, essa técnica de treinamento baseada em *self-play* é utilizada em conjunto com a técnica de aprendizagem por reforço (Heinrich e Silver, 2014).

Em (Sandven e Tessem, 2006), a base de casos de um sistema CBR aplicado ao jogo de Poker foi inteiramente construída a partir de partidas disputadas na modalidade *self-play*. Por meio de técnicas de CBR, agentes desenvolvidos nesse trabalho foram treinados a partir da disputa de 50000 mãos de Poker. Quando a base de casos não tinha conhecimento necessário para a tomada de decisões de jogo nessas partidas, tais agentes utilizavam uma jogada aleatória. Nesse processo de treinamento, apesar de não explorar a modelagem de oponentes, não demorou muito para esses agentes fossem capazes de bater, de forma consistente, alguns de seus oponentes. Porém, devido ao treinamento ter sido efetuado com base em apenas um tipo de oponente, foi possível observar que somente um estilo de jogo dominante foi aprendido. Nesse caso, o agente tornou-se extremamente agressivo devido ao estilo conservador dos oponentes utilizados na etapa de treinamento. Além disso, uma mesma política de reuso foi explorada durante a formação da base de casos, o que contribuiu para o desenvolvimento de um único estilo de jogo usado pelos agentes desenvolvidos. Por fim, os agentes desenvolvidos foram testados contra agentes baseados em regras, onde a abordagem de treinamento usada se mostrou promissora apenas contra poucos desses oponentes. Porém, em partidas onde vários oponentes baseados em regras foram adicionados, estratégias muito agressivas acabaram ficando arriscadas e fizeram com que o agente deixasse de ter um bom desempenho. Abordagens alternativas a essa proposta poderiam utilizar diferentes pares de jogadores para a expansão da base de casos, permitindo construir uma base de casos contendo casos com estilos de jogos distintos.

Ciente das dificuldades de criar bases de casos com ampla cobertura de diferentes tipos de problemas no domínio de aplicação sendo atacado, (Floyd *et al.*, 2008) descreve como utilizar soluções contidas em casos similares recuperados para consultas dadas na criação de novos casos a serem retidos na base de casos. Nesse trabalho, quando agentes CBR foram explorados em campeonatos de futebol de robôs, esses agentes realizaram consultas usando o estado atual do jogo, onde a resposta dessas consultas indicava a ação de jogo que deveria ser executada. Dessa forma, pares (estado - ação) puderam ser retidos como novos casos na base de casos. Em geral, essa capacidade de criar novas experiências de solução de problemas de forma automática é uma importante característica da aprendizagem por imitação (Gómez-Martín *et al.*, 2005), o qual também pode ser explorado no desenvolvimento de sistemas CBR.

2.4. UM JOGO DE CARTAS USADO NO DESENVOLVIMENTO DA DISSERTAÇÃO: O JOGO DE TRUCO

As propostas apresentadas nesta dissertação foram implementadas e avaliadas no desenvolvimento de agentes CBR capazes de jogar Truco (Winne, 2017). Embora esta seção brevemente apresente características do jogo de Truco, as quais são relevantes para a compreensão do trabalho apresentado nesta dissertação, uma introdução mais detalhada deste jogo de cartas pode ser encontrada em (Winne, 2017).

Semelhante ao Poker, o Truco envolve desafios cognitivos relacionados com a visão parcial das cartas do oponente e a estocasticidade do sorteio das cartas. No entanto, diferente do Poker, o Truco possui várias etapas de tomada de decisão devido as múltiplas interações de jogo encontradas nas diferentes mãos disputadas. Neste caso, partidas de Truco são divididas em mãos onde dois (jogador mão, que é o primeiro a jogar, e jogador pé) ou mais adversários interagem. Em particular, esse trabalho explora partidas entre dois jogadores apenas.

O Truco utiliza 40 cartas de um baralho espanhol. Em cada mão de jogo, as cartas são embaralhadas e distribuídas, onde cada jogador recebe três cartas. Para a disputa de uma mão, cada um dos jogadores recebe três cartas. O jogo é dividido em várias mãos, onde ele geralmente termina quando um jogador atinge 30[1] pontos. Por ser um jogo baseado em turnos, os jogadores se revezam para jogar cartas e efetuar apostas na disputa das mãos, onde o vencedor de um turno inicia o próximo turno do jogo naquela mão. Para ganhar uma mão, o jogador precisa ganhar pelo menos dois dos três turnos da mão disputada. Para cada turno de uma mão, diferentes cenários de jogo são apresentados.

As principais ações deste jogo no Truco podem ser divididas em "jogar cartas" e "fazer apostas". Dada uma mão de jogo, jogadores têm o objetivo de largar na mesa uma carta maior do que a carta do seu oponente em no mínimo dois dos três turnos que constituem uma mão. As apostas são efetuadas com o objetivo de aumentar o número de pontos conquistados no jogo. Embora as apostas sejam geralmente feitas para aumentar os ganhos quando o jogador tem boas cartas, elas também podem ser feitas para forçar o oponente a desistir no caso de um blefe, e consequentemente permitir ganhar os pontos disputados em uma mão de jogo.

Existem duas modalidades de apostas possíveis em uma mão de Truco: o ENVIDO e o TRUCO (que possuí o mesmo nome do jogo). Enquanto que no TRUCO as apostas são baseadas na força relativa das cartas, o ENVIDO é baseado em pontuações provenientes da soma de cartas cujos naipes são iguais em uma dada mão. Para cada uma das apostas, os jogadores podem aceitar, negar ou aumentar as apostas realizadas. Na modalidade de apostas TRUCO, as apostas devem seguir um fluxo de interação entre os oponentes, onde não é possível começar com uma aposta máxima. No caso de um dos jogadores negar alguma das apostas, o jogador que fez a aposta negada pelo oponente recebe os pontos da última aposta aceita, ou um ponto, no caso da primeira aposta ter sido negada. Quando um dos jogadores vence a modalidade de aposta TRUCO, a mão atual é encerrada. A modalidade de apostas do tipo ENVIDO pode ser realizada somente antes da modalidade de apostas do tipo TRUCO iniciar. Diferente do TRUCO, o ENVIDO não encerra a mão sendo jogada. Mais ainda, o ENVIDO não precisa seguir uma ordem de apostas, por isso diferentes combinações de apostas podem existir. Para facilitar a compreensão, sempre que uma aposta é aumentada por algum jogador, um incremento na respectiva modalidade de aposta é considerado. A ideia é que toda vez que uma aposta é aumentada, um novo nível de disputa da mão corrente é

[1] Diferentes pontuações podem ser definidas por grupos de jogos

alcançado. Nesta dissertação, essa interação é descrita por níveis, onde o nível 1 representa apostas iniciais propostas por jogadores.

A modalidade de aposta ENVIDO é subdividida em dois tipos de apostas, são elas: ENVIDO (possuí o mesmo nome da modalidade de aposta) e FLOR. As pontuações consideradas na modalidade ENVIDO são criadas pela soma de pontuações de cartas que possuem o mesmo naipe. Quando apenas duas cartas possuem o mesmo naipe, o tipo da aposta também é identificado como ENVIDO. Porém, quando 3 cartas possuem o mesmo naipe o tipo de aposta é identificado como FLOR.

Similar ao jogo de Poker, os diferentes cenários de decisão do jogo de Truco revelam informações distintas sobre as cartas do oponente. Após cada carta ser jogada, mais completas são as informações disponíveis para orientar a escolha da próxima ação de jogo a ser tomada. O jogador que inicia uma mão, chamado de "jogador mão", tem visão apenas das cartas disponíveis na sua mão. O segundo a jogar, chamado de "jogador pé", pode ver a primeira carta do oponente a menos que o oponente faça uma aposta antes de jogar qualquer carta.

Como todos os jogos de cartas, o número de possibilidades de jogo no Truco pode ser quantificado. Apesar do elemento surpresa proporcionado pelo sorteio das cartas, as combinações de cartas possíveis são finitas. Em uma partida de mano (onde dois jogadores se desafiam), o primeiro jogador pode receber 9.880 mãos diferentes, enquanto o segundo jogador pode receber 7.770 mãos diferentes. Isso gera um total de 76.767.600 diferentes combinações de cartas que podem ser recebidas por dois jogadores. Essa quantidade de combinações já tornaria inviável a construção de árvores de jogos. Porém, esse número pode ser ainda maior se considerarmos que em cada combinação de cartas de uma mão podem existir diferentes ordens em que os jogadores realizam as jogadas e diversas possíveis interações de apostas. Por estar dividido nas modalidades de TRUCO (aposta do tipo TRUCO e ação de jogar cartas) e ENVIDO (apostas do tipo ENVIDO e FLOR), um mesmo conjunto de cartas pode possuir diferentes importâncias em cada uma das modalidades. Isto é, cartas com pontuação alta que possuem uma alta importância para modalidade do ENVIDO e com baixo valor de importância para apostas de TRUCO, por exemplo. Considerando cartas com diferentes naipes, as quais possuem o mesmo valor no jogo, e avaliando apenas combinações relevantes para a modalidade do TRUCO, um jogador pode ter 502 possíveis mãos de jogo. Enquanto na modalidade ENVIDO, existem 22 diferentes pontuações possíveis para combinações de duas cartas com o mesmo naipe. Além disso, existem 27 pontuações possíveis para mãos de jogo compostas por 3 cartas com o mesmo naipe. Em se tratando de pontuações neste jogo, Tabela 1 apresenta o ranking que as cartas pertencem para a modalidade TRUCO e a pontuação que proporciona na soma da modalidade ENVIDO. Tabela 2 apresenta um exemplo de como a soma dos pontos é realizada quando existem cartas com o mesmo naipe.

Tabela 1 - Ordem das cartas

Cartas	**Ranking para o TRUCO**	**Pontos ENVIDO**
1 de Espada	1	1
1 de Bastos	2	1
7 de Espada	3	7

7 de Ouro	4	7
Todos 3s	5	3
Todos 2s	6	2
1 de Copas e 1 de Ouro	7	1
Todos os 12s	8	0
Todas as 11s	9	0
Todos os 10s	10	0
7 de Bastos e 7 de Copas	11	7
Todos 6s	12	6
Todos 5s	13	5
Todos 4s	14	4

Fonte: do autor

Tabela 2 – Exemplo de soma de pontuações para a modalidade de apostas ENVIDO

Combinação de cartas	Pontuação
7 e 6 do mesmo naipe	7 + 6 + 20 = 33
7 e 5 do mesmo naipe	7 + 5 + 20 = 32
7 e 4 do mesmo naipe	7 + 4 + 20 = 31
7 e 2 do mesmo naipe	7 + 2 + 20 = 29
7 e 1 do mesmo naipe	7 + 1 + 20 = 28
7 e 10 ou 7 e 11 ou 7 e 12 do mesmo naipe	7 + 0 + 20 =27
7 e outras cartas de naipes diferentes	7

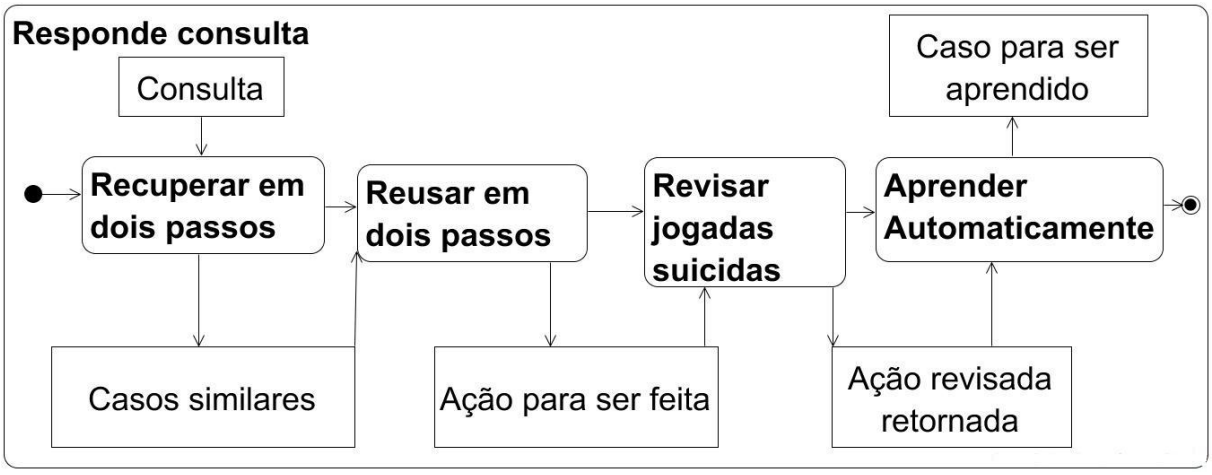
Fonte: adaptado de (Winne, 2017).

Em resumo, quando comparado com outros jogos de cartas, um dos principais desafios do Truco está relacionado com as diferentes modalidades de apostas (por exemplo, apostas do tipo ENVIDO e apostas do tipo TRUCO) em uma mesma disputa de mão no jogo. Apesar de proporcionar desafios computacionais semelhantes aos existentes no Poker, diferentes tipos de decisões precisam ser combinados no jogo de Truco.

3. ABORDAGENS PROPOSTAS PARA IDENTIFICAÇÃO DE COMPORTAMENTOS DE JOGO, RECUPERAÇÃO E ORGANIZAÇÃO DA BASE DE CASOS

O desenvolvimento de políticas de reuso é o principal foco de pesquisa conduzida nesta dissertação. Apesar disso, este trabalho também apresenta um conjunto de técnicas utilizadas para apoiar o desenvolvimento das etapas do ciclo de CBR, as quais sustentaram a implementação de agentes capazes de eficientemente jogar Truco. De forma resumida, as principais implementações desenvolvidas nesta dissertação podem ser divididas em: recuperação em dois passos, reuso em dois passos, revisão de jogadas suicidas, e aprendizagem com ajuste automático de grupos contendo casos da base de casos (ver Figura 3).

Figura 3 - Diagrama de atividades descrevendo a abordagem CBR proposta



Fonte: do autor.

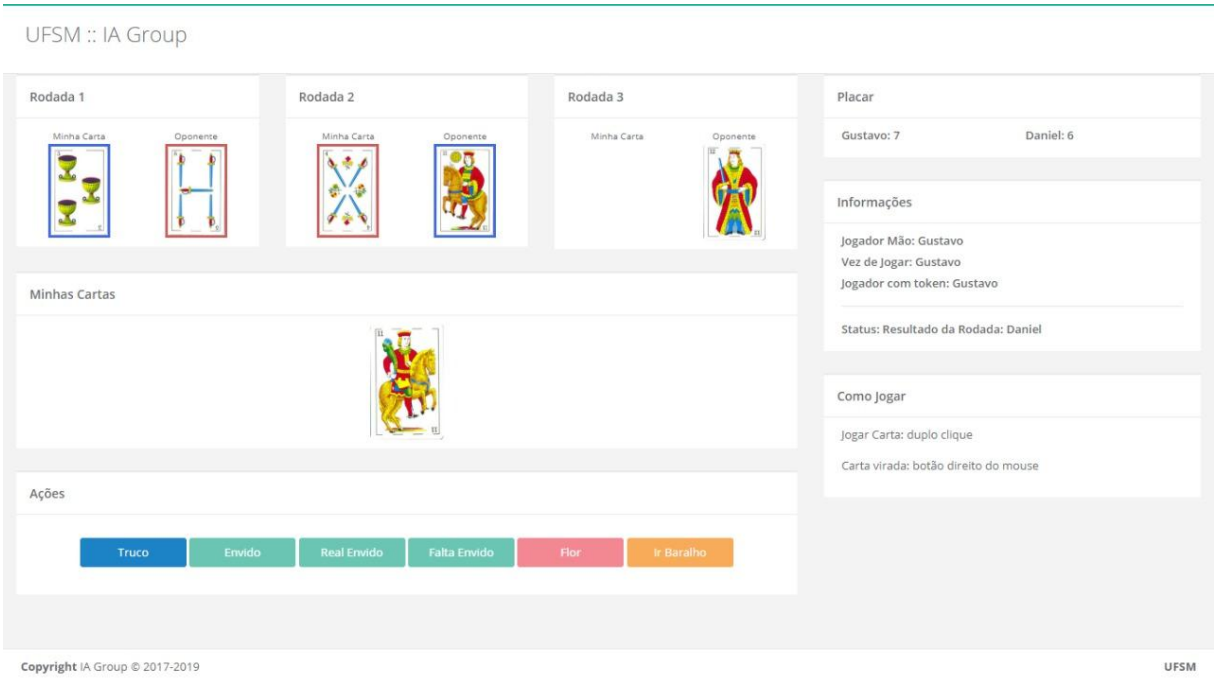
Em resumo, ao receber uma consulta, é realizada a computação de similaridade entre o caso tomado como consulta e os grupos de casos formados para indexar a base de casos (e reduzir o tempo necessário para obter respostas para as consultas realizadas no sistema). Ao ser identificado o grupo mais similar ao caso usado como consulta, é computada a similaridade com todos os casos contidos no grupo mais similar (enquanto outros casos pertencentes a outros grupos não são usados). Após retornar os casos mais similares recuperados, decisões de jogo tomadas no passado nestes casos são reusadas através de um modelo de reuso em dois passos. Para computar esse reuso, a abordagem

proposta utiliza o algoritmo K –*MEANS* para identificar e explorar os diferentes cenários de decisão contidos nos casos armazenados na base de casos. Dessa forma, um critério de reuso é aplicado na escolha de um desses grupos de casos, dentre os grupos representados pelos casos recuperados. Após isso, outro critério de reuso é aplicado na escolha da solução/ação de jogo que deve ser reusada de acordo com os casos mais similares recuperados. Por existirem casos obtidos em jogos disputados por jogadores com diferentes níveis de experiência, algumas jogadas armazenadas nestes casos recuperados foram executadas apenas em situações muito particulares, onde tais ações de jogo são difíceis de serem corretamente reusadas. Na etapa de revisão, portanto, é analisado se o modelo de reuso acabou não selecionando/sugerindo uma ação de jogo considerada "suicida", isto é, retornado uma ação com alto risco de fracasso. Por fim, critérios de aprendizagem são utilizados para analisar se a experiência de jogo corrente (sendo construída com o uso das técnicas desenvolvidas) deve ser armazenada como um novo caso na base de casos. Quando novas experiências são adicionadas na base de casos, novas combinações entre estados e soluções podem ser acrescentadas nesta base. Por isso, foi necessário propor critérios que analisaram, em tempo de execução, se os grupos de casos existentes e usados pelos algoritmos não foram alterados. Dessa forma, na ocorrência de alterações nesses grupos, principalmente no número de grupos, conforme análise realizada a partir do algoritmo WCSS, o algoritmo K – *means* é reexecutado e os agrupamentos de casos obtidos são automaticamente utilizados (substituindo os agrupamentos anteriormente sendo utilizados), sem precisar interromper a execução do processo de treinamento sendo executado.

3.1. IMPLEMENTAÇÃO CBR

Um sistema web foi desenvolvido e utilizado para coletar partidas de Truco permitindo popular a base de casos com informações detalhadas sobre mãos de jogo disputadas por diferentes pares de jogadores humanos (exemplo de coleta de casos entre humanos apresentado na Figura 4). Dessa forma, em conjunto com outros integrantes do grupo de pesquisa onde esta dissertação está inserida, uma base de casos inicial contendo logs completos de 3195 mãos de Truco foi produzida (base de casos tomada como *baseline* nos testes realizados nesta dissertação). Os casos presentes nesta base de casos foram representados como pares atributo-valor, conforme descrito na Tabela 3.

Figura 4 – Exemplo de coletas de casos



Fonte: Grupo de pesquisa.

Tabela 3 – Exemplos de atributos utilizados na representação de mãos de Truco

Atributos	**Descrição**	**Função de Similaridade Local Utilizada**
Cartas recebidas (codificação apresentada na Tabela 7)	Cartas recebidas pelos jogadores são representadas nos casos como atributos numéricos. Cartas são codificadas em uma escala numérica não linear variando de 1 até 52. De acordo com (Winne, 2017), os valores nessa escala capturam a importância relativa das	INTERVAL(52)

	cartas no jogo de Truco.	
Jogadores que iniciam jogando em cada uma das mãos	Jogador 1 ou Jogador 2.	EQUAL
Cartas jogadas em cada turno de uma mão	As três cartas recebidas pelos jogadores são codificadas como carta alta, média e baixa. Este atributo captura qual carta foi jogada em cada um dos 3 turnos de uma mão	INTERVAL(52)
Jogador que ganhou cada um dos turnos de uma mão	Jogador 1 ou Jogador 2	EQUAL
Apostas feitas pelos jogadores	Representa as apostas que cada um dos jogadores realizou	EQUAL
Quantidades de pontos ganhos / perdidos	Quantidade de pontos ganhos ou perdidos em cada uma das diferentes ações de jogo executadas na mão	INTERVAL(pontuação máxima que pode ser conquistada na modalidade de apostas em que a computação de similaridade é aplicada)

Quantidade de pontos disponíveis na mão	Soma da pontuação de cartas com o mesmo naipe de acordo com as regras de pontuação do ENVIDO	EQUAL
Grupo que o caso pertence para cada uma das decisões de jogo possíveis de serem tomadas em uma mão	Cluster que o caso pertence para cada uma das decisões de jogo presentes na disputa de uma mão de Truco. Atributo utilizado para recuperar os casos pertencentes aos grupos formados.	EQUAL

Fonte: do autor.

A codificação utilizada para representar as cartas foi desenvolvida de acordo com categorias identificadas em (Winne, 2017), além de explorar o conhecimento dos autores e integrantes do grupo de pesquisa sobre o jogo de Truco. Essa representação captura uma escala de importância para cada uma das cartas. Por exemplo, Tabela 4 apresenta a codificação utilizada para representar as cartas, bem como a categoria que a carta pertence e a pontuação que cada carta proporciona para a modalidade do ENVIDO.

Tabela 4 – Codificação utilizada para representar cartas usadas no jogo de Truco

Categorias	**Carta**	**Quantidade de cartas com este valor**	**Codificação CBR**	**Pontos Envido**
Maiores Altas	1 de Espada	1	52	1
	1 de Bastos	1	50	1
Maiores Baixa	7 de Espada	1	42	7
	7 de Ouro	1	40	7

s				
Brancas Altas	Todos os 3’s	4	24	3
	Todos os 2’s	4	16	2
Brancas Médias	1 de Copas e 1 de Ouro	2	12	1
Negras Baixas	Todos os 12’s	4	8	0
	Todas os 11’s	4	7	0
	Todos os 10’s	4	6	0
Brancas Baixas	7 de Bastos e 7 de Copas	2	4	7
	Todos os 6’s	4	3	6
	Todos os 5’s	4	2	5
	Todos os 4’s	4	1	4

Fonte: do autor.

Na disputa de uma mão de Truco, diferentes consultas CBR são executadas pelos agentes desenvolvidos. Neste processo, diferentes consultas são formadas por diferentes atributos (utilizam diferentes atributos, os quais capturam os estados do jogo). Um exemplo de computação de similaridade entre um caso usado como consulta e um caso armazenado na base de casos é apresentado na Tabela 5.

Tabela 5 - Exemplo de computação de similaridade de uma consulta executada no sistema CBR

Atributo analisado	**Valor do atributo usado na consulta**	**Valor do atributo do caso armazenado na base de casos**	**Cálculo de similaridade usado**	**Similaridade local computada**

Carta alta	24	24	Si = 1 – ((|24 – 24|)/52)	1
Carta média	12	8	Si = 1 – ((|12 – 8|)/52)	0,9231
Carta baixa	6	7	Si = 1 – ((|6 – 7|)/52)	0,980769231
Jogador mão	1	1	1 == 1	1
Similaridade Global	**Computação de similaridade Global**			**Similaridade entre os casos**
	Sim = 1+0,9231+0,980769231+1=3,9039 Sim = 3,9039 / 4			Sim = 0,975975

Fonte: do autor.

A consulta apresentada como exemplo na Tabela 5 descreve a computação de similaridade, conforme apresentado na equação 2, feita para determinar a ação de jogo onde o jogador deve jogar a primeira carta no cenário em que este jogador é o primeiro a jogar. As informações do jogo disponíveis para tomar a decisão de que carta jogar são: cartas recebidas (carta alta, média e baixa recebidas) e quem inicia jogando (jogador mão).

3.1.1. MODELO DE RECUPERAÇÃO DE CASOS UTILIZADO

Em sistemas CBR, as decisões são tomadas com base em experiências de solução de problemas recuperadas de uma base de casos. Dessa forma, computar a similaridade com casos irrelevantes para o problema atual faz com que o tempo de processamento das consultas seja acrescido desnecessariamente. Conforme (Bellamy-Mcintyre, 2008; Chen *et al.*, 2018; Lucca *et al.*, 2018), algoritmos de *clustering* podem ser utilizados para criar bases de casos indexadas de múltiplas formas, permitindo computar a similaridade com casos mais relevantes para a solução do problema atual descrito em uma consulta. Em geral, o objetivo é reduzir o tempo de resposta das consultas realizadas no sistema CBR. Para isso, a base de casos utilizada nesta dissertação foi indexada/organizada da seguinte forma:

a) a base de casos foi dividida de acordo com as cartas recebidas pelo jogador para otimizar a etapa de recuperação de ações de jogo do tipo “jogar cartas” e “realizar apostas”, as quais são empregadas na modalidade TRUCO. Para isso, foram utilizados atributos relevantes para identificar as cartas recebidas;

b) a base de casos foi dividida de acordo com a pontuação da mão para otimizar a etapa de recuperação de ações de jogo envolvendo apostas realizadas nas modalidades

ENVIDO e FLOR. Para isso, foi utilizado apenas o atributo que representa a soma da pontuação de cartas que são do mesmo naipe.

Assim como desenvolvido em (Chen *et al.*, 2018), a definição do número de grupos existente na execução do algoritmo K – MEANS foi realizada com base no método *Elbow* através da execução do algoritmo WCSS. Com a base de casos particionada, a similaridade entre o caso consulta e os casos armazenados na base de casos é computada com o uso do centroide de cada um dos grupos formados. Em seguida, após determinar qual dos grupos é o mais similar com a consulta dada, a similaridade da consulta é computada com cada um dos casos que pertencem a esse grupo mais similar escolhido.

Para garantir que as experiências de jogo mais similares possíveis a uma dada consulta, sejam recuperadas e utilizadas, um valor de *threshold* adaptativo foi utilizado no mecanismo de recuperação de casos implementado. Isto é, inicialmente são recuperados apenas casos que possuem similaridade maior ou igual a 98% com a consulta, a qual é executada de acordo com os diferentes cenários de decisão do jogo de Truco (para cada decisão de jogo, diferentes conjuntos de atributos podem ser usados para capturar o estado do jogo e a ação de jogo que deve ser tomada pelo agente). Para garantir que os casos similares recuperados pertencem ao mesmo cenário de jogo da consulta, são removidos da lista de casos recuperados todos aqueles casos que não pertencem ao cenário da consulta. Caso não sejam recuperados no mínimo 5 casos com esse valor de similaridade, novas tentativas de consulta são executadas através da diminuição de 2% no *threshold* mínimo de similaridade exigido. Isso ocorre até que um número mínimo de casos similares seja recuperado com a similaridade mais alta possível. A partir disso, uma lista de casos é recuperada, e esses casos são usados como entrada na computação dos critérios de reuso propostos nessa dissertação.

3.2. O USO DE ALGORITMOS DE *CLUSTERING* PARA IDENTIFICAR PARES ESTADO–AÇÃO NO JOGO DE TRUCO

Nesta dissertação, algoritmos de *clustering* foram utilizados para encontrar diferentes padrões de jogo nos logs de jogos de Truco gravados como casos na base de casos. Por se tratar de um jogo baseado em turnos, onde cada turno permite diferentes informações para a tomada de decisão. A ideia foi identificar e representar diferentes combinações entre estado e ações de jogo para cada um dos diferentes cenários de decisão existentes nas configurações de turnos possíveis neste jogo. Neste caso, o jogo de Truco é organizado em diferentes estados de decisões, onde ações de jogo precisam ser tomadas a partir da análise de diferentes informações disponíveis. Os diferentes estados de decisão existentes, são identificados como cenários de decisão. Dessa forma, cada cenário proporciona diferentes informações para a realização de ações de jogo. Ao investigar o jogo de Truco, percebemos que existem 14 diferentes cenários de decisão neste jogo. Por isso, para identificar as diferentes táticas de jogo existentes em cada um dos cenários identificados, 14 diferentes agrupamentos de casos foram necessários para a criação de grupos que foram usados pelos algoritmos de decisão implementados nos agentes desenvolvidos. Os 14 cenários de decisão analisados são detalhados na Tabela 6:

No total, as somas dos 14 agrupamentos feitos para os diferentes cenários proporcionaram 67 grupos diferentes na base final. Os atributos utilizados em cada um dos diferentes agrupamentos para cada um dos cenários existentes no jogo de Truco podem ser encontrados na Tabela 1 do Apêndice B.

Tabela 6 – Cenários de agrupamentos

Tipo de Jogada	Cenários de decisão no jogo de Truco	Grupos de casos formados
Cenários para ações do tipo jogar cartas	primeira carta quando o agente é o primeiro a jogar	4
	primeira carta quando o agente é o segundo a jogar	4
	segunda carta quando o agente ganhou a primeira rodada	4
	segunda carta quando o agente perdeu a primeira rodada	6
	terceira carta quando o agente ganhou a segunda rodada	2
	terceira carta quando o agente perdeu a segunda rodada	4
Cenários para ações de apostas do tipo ENVIDO	combinações de apostas do tipo ENVIDO quando o agente é o primeiro a apostar	5
	combinações de apostas ENVIDO quando o agente é o segundo a apostar	5
Cenários para ações de apostas do tipo TRUCO	combinações de apostas do tipo TRUCO na primeira rodada quando o agente é o primeiro a ter o direito de apostar	8
	combinações de apostas do tipo TRUCO na primeira rodada quando o agente é o segundo a ter o direito de apostar	4
	combinações de apostas do tipo TRUCO na segunda rodada quando o agente é o primeiro a ter o direito de apostar	6
	combinações de apostas do tipo TRUCO na segunda rodada quando o agente é o segundo a ter o direito de apostar	4
	combinações de apostas do tipo TRUCO na terceira rodada quando o agente é o primeiro a ter o direito de apostar	7
	combinações de apostas do tipo TRUCO na terceira rodada	4

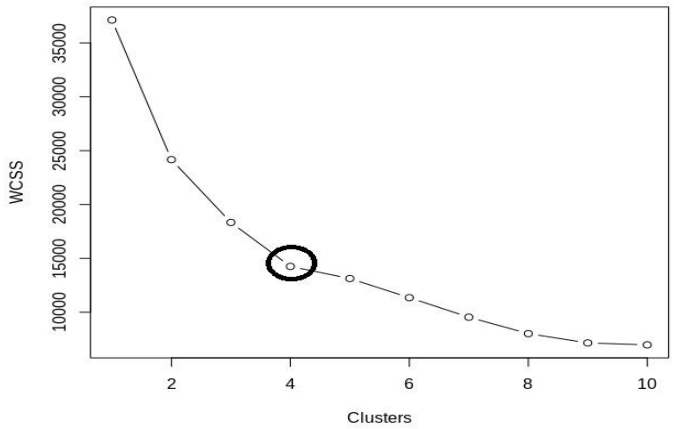
	quando o agente é o segundo a ter o direito de apostar	

Fonte: do autor.

Usando como entrada os casos armazenados na base de casos, em cada um dos 14 cenários de decisão organizados no sistema CBR desenvolvido, a definição do número de grupos formados nas execuções do algoritmo K – MEANS foi realizada com base no método *Elbow*, o qual considera o algoritmo WCSS. Por exemplo, Figura 5 apresenta o *plot Elbow* utilizado para identificar o número de grupos de casos existentes para a ação de jogo que envolve "jogar a primeira carta" em uma mão no cenário de jogo onde "o jogador é o segundo a jogar". Na Figura 5, por exemplo, o número de grupos sugerido é 4, esse número pode ser alterado de acordo com os casos contidos na base.

Na prática, esse processo de análise de grupos permite identificar táticas de jogo usadas em cada um dos cenários de decisão relacionados com ações de jogar cartas e realizar apostas. Isso permite analisar a relação entre estados de jogo (cartas recebidas, carta jogada pelo oponente, quem chamou truco e demais atributos que são utilizados para representar o estado de jogo) e ações de jogo que são executadas (jogou a carta alta recebida, jogou a carta media recebida, jogou a carta baixa recebida, foi ao baralho e demais ações de jogo possíveis).

Figura 5 – Exemplo de seleção do número de grupos no cenário em que o agente é o segundo a jogar a primeira carta de uma mão em disputa no jogo de Truco



Fonte: do autor.

No jogo de Truco, existem diferentes categorias de cartas que podem ser identificadas para apoiar o desenvolvimento de sistemas CBR e apoiar o desenvolvimento de tarefas de descoberta de conhecimento via algoritmos de *clustering*. Em geral, a identificação dessas categorias de cartas e criação de codificações numéricas para representar elas foram feitas de acordo com (Winne, 2017). Na prática, cartas que pertencem a categorias idênticas, as quais possuam a mesma relevância no jogo, por exemplo, tendem a não alterar o comportamento de jogo dos jogadores. Tabela 4

apresentada na seção 3.1 detalha as cartas do jogo de Truco e as respectivas categorias que cada uma delas podem ser representadas. Tabela 4 também apresenta a codificação utilizada pelos algoritmos de CBR, a qual foi definida para cada uma das cartas do baralho usado neste jogo. Além de ser utilizada pelos algoritmos de CBR, é relevante ressaltar que a codificação numérica utilizada para esses atributos é um passo importante para viabilizar a melhor execução de tarefas de descoberta de conhecimento em bases de casos. Neste caso, tal codificação geralmente deve ser construída de acordo com os objetivos de análise investigados no problema de aplicação alvo. Na prática, a codificação utilizada por um sistema CBR é geralmente bastante específica para os problemas de aplicação sendo atacados, podendo ser ajustada de diferentes formas para obter melhores resultados nas computações de similaridade desenvolvidas para resolver problemas nestas aplicações. Em sistemas CBR, por exemplo, essa codificação utilizada para representar valores de atributos pode ser capaz de diferenciar uma carta de outra mesmo que elas possuam a mesma relevância no jogo. Por outro lado, a codificação de dados requerida por algoritmos de agrupamento de dados precisa ser mais geral e capaz de facilitar a melhor identificação de padrões mais genéricos de jogo.

Nesta dissertação, para permitir a análise dos casos armazenados na base de casos via algoritmos de *clustering*, a codificação das cartas (codificação das categorias de cartas) usada na investigação destes agrupamentos foi construída/ajustada de acordo com a média dos valores encontrados em cada uma das categorias de cartas usadas na construção dos algoritmos de CBR. Por exemplo, a primeira categoria existente na Tabela 4 (ver na seção 3.1) é a categoria das "Maiores Altas". Esta categoria de cartas possuí duas diferentes cartas, uma representada pela codificação 52 e a outra representada pela codificação 50. Neste caso, a codificação utilizada para identificar esta categoria é representada pela soma destes valores dividido pelo número de ocorrências de categorias, por exemplo, considerando a soma de 50 com 52 dividido por 2, visto que existem duas cartas nessa categoria. Isso resulta na utilização da codificação 51, o qual é usado na análise dos agrupamentos de casos contidos na base de casos. Tabela 7 apresenta as codificações de cartas utilizadas para identificar as principais categorias de cartas de acordo com o que é apresentado na Tabela 4 da seção 3.1.

A codificação apresentada na Tabela 7 se fez necessária por dois motivos principais:

1. Cartas que pertencem a uma mesma categoria tentem a não alterar o comportamento dos jogadores (Winne, 2017).
2. Utilizar a codificação das cartas nos agrupamentos dificultaria a identificação das táticas de jogo existentes.

Tabela 7 – Codificação das cartas utilizada na execução dos algoritmos de *clustering*

Categoria de cartas no jogo de Truco	**Codificação numérica das cartas usada pelos algoritmos de *clustering***
Maiores Altas	51
Maiores Baixas	41
Brancas Altas	20

Brancas Médias	12
Negras Baixas	7
Brancas Baixas	3

Fonte: do autor.

Como cada jogador recebe três cartas em uma mão, as cartas jogadas são divididas entre: carta alta da mão, carta média da mão e carta baixa da mão. Como os grupos de casos desenvolvidos visam identificar combinações entre estados e ações, em ações do tipo jogar carta, é necessário identificar qual das 3 cartas foi jogada pelos jogadores em uma mão de jogo (carta alta da mão, carta média da mão ou carta baixa da mão). A codificação utilizada para identificar qual dessas cartas foi jogada também foi desenvolvida com o auxílio das categorias descritas na Tabela 4 (apresentada na seção 3.1). Isto é, dividimos as 6 categorias identificadas na Tabela 4 em 3, visto que temos cartas de valores altos, cartas de valores médios e cartas de valores baixos. A organização dessas categorias foi desenvolvida da seguinte forma:

- Cartas de valores altos: construída pela união das categorias “Maiores Altas” e “Maiores Baixas” identificadas na Tabela 4. A codificação final para identificar quando uma carta de valor alto foi jogada é construída através do resultado da a) soma de 50 com 52, as quais são as duas cartas que pertencem as “Maiores Altas” (identificadas na Tabela 4), com a b) soma das codificações das duas cartas que pertencem a categoria das ”Maiores Baixas”. Por exemplo, a soma de 42 com 40 (também identificadas na Tabela 4) dividida pelo número de cartas consideradas na soma do número de cartas pertencentes as categorias utilizadas, onde existem 4 cartas consideradas nesta situação.
- Cartas de valores médios: construída pela união das categorias “Brancas Altas” e “Brancas Médias” apresentadas na Tabela 4. A codificação final para identificar quando uma carta de valor médio foi jogada é construída pelo resultado da soma de a) 24 multiplicado por 4 (existem 4 cartas com essa codificação identificadas na categoria “Brancas Altas” na Tabela 4) com b) 16 multiplicado por 4 (existem 4 cartas com essa codificação também identificadas na categoria de “Brancas Altas” na Tabela 4) com c) 12 multiplicado por 2 (existem 2 cartas com essa codificação identificadas como Brancas médias na Tabela 4), onde esta soma é dividida por 10 que é o número de cartas pertencentes as categorias consideradas nesta situação.
- Cartas de valores baixos: construída pela união das categorias “Negras Baixas” e “Brancas Baixas” na Tabela 4. A codificação final construída para identificar quando uma carta de valor baixo foi jogada é construída pelo resultado da soma de a) 4 multiplicado por 8 (existem 4 cartas com esta codificação identificadas na categoria das “Negras Baixas” na Tabela 4) com b) 4 multiplicado por 7 (existem 4 cartas com esta codificação também identificadas na categoria das “Negras Baixas” na Tabela 4) com c) 4 multiplicado por 6 (existem 4 cartas com esta codificação também identificadas na categoria das “Negras Baixas” na Tabela 4) com d) 2 multiplicado por 4 (existem duas cartas com esta codificação identificadas na categoria das “Brancas Baixas” na Tabela 4) com e) 3 multiplicado por 4 (existem 4 cartas com esta codificação também identificadas na categoria das “Brancas Baixas” na Tabela 4) com f) 2 multiplicado por 4 (existem 4 cartas com esta codificação também identificadas como “Brancas

Baixas" na Tabela 4) com e) 1 multiplicado por 4 (existem 4 cartas com esta codificação também identificadas na categoria das "Brancas baixas" na Tabela 4), onde esta soma é dividida por 26, que é o número de cartas pertencentes as categorias utilizadas nesta situação.

A codificação final utilizada para identificar qual das três cartas foi jogada (necessária para a realização do agrupamento de ações de jogo do tipo jogar cartas) é apresentada na Tabela 8. Por ser de suma importância para a etapa de descoberta de conhecimento e por necessitar ser definida de acordo com a aplicação objetivo. Nesta situação específica, também acreditamos ser prudente utilizar as categorias das cartas, para identificar qual das cartas dentre as recebidas foi jogada.

Tabela 8 – Codificação para representar qual das cartas recebidas que foi jogada

Carta jogada	**Codificação para a carta jogada utilizada pelos algoritmos de clustering**
Jogou a carta Alta da mão	46
Jogou a carta Média da mão	16
Jogou a carta baixa da mão	4

Fonte: do autor.

Outra situação relevante tratada no desenvolvimento desta dissertação diz respeito a representação de rodadas de uma mão de jogo que não são concluídas pelos jogadores por vários motivos. Por exemplo, um jogador não tem uma boa mão, e simplesmente vai ao baralho encerrando a mão sem revelar as suas cartas para o oponente. Nesta situação, as cartas recebidas pelos jogadores podem não ser jogadas/reveladas na mesa de jogo. É preciso que os algoritmos de *clustering* sejam capazes de identificar motivos, os quais são claramente visíveis no final de uma rodada de jogo, que levaram as cartas a não terem sido jogadas na mesa, resultando em atributos de casos que não possuem valores associados nos casos armazenados na base de casos. Dessa forma, Tabela 9 representa as codificações utilizadas para representar os motivos das cartas não terem sido reveladas pelos jogadores durante a disputa de mãos de Truco. A utilização desta forma de codificação foi necessária por duas razões principais: a) algoritmos de *clustering* não trabalham com valores nulos nas computações de similaridade realizadas e b) é geralmente importante saber o porquê uma carta não foi jogada em uma disputa de uma mão.

Tabela 9 – Codificação relacionada aos motivos de cartas não terem sido jogadas

Carta jogada	**Codificação**

	***Clustering* para cartas não jogadas**
Oponente foi ao baralho	-46
Carta virada	-1
Agente foi ao baralho	-10
Agente perdeu a aposta	-15

Fonte: do autor.

Nos algoritmos de CBR desenvolvidos nesta dissertação, a computação de similaridade entre os atributos que representam os jogadores foi realizada de acordo com a função *equal* (conforme explicado na Tabela 3). Por isso, as codificações utilizadas neste atributo foram 1 para identificar o jogador 1, 2 para identificar o jogador 2 e 0 para quando nenhum dos jogadores realizou uma ação de jogo. Por utilizar a distância Euclidiana nas computações de similaridade realizadas pelo algoritmo K-MEANS, a codificação utilizada para representar cada um desses jogadores nos logs de mãos de jogos não pôde ser a mesma que a codificação utilizada pelos algoritmos de CBR desenvolvidos. Se a codificação 0 fosse empregada para representar que nenhum jogador realizou uma jogada, a computação da distância Euclidiana consideraria que o jogador 1 seria mais similar que o jogador 2, e isto não estaria correto. Por isso, a codificação utilizada para representar jogadores nos agrupamentos foi representada de acordo com a Tabela 10.

Tabela 10 – Codificação de jogadores usada nas computações de similaridade realizadas pelos algoritmos de *clustering*

Quem	**Codificação**
Jogador 1	1
Nenhum dos dois jogadores	2
Jogador 2	3

Fonte: do autor.

Antes de realizar os agrupamentos de casos, todos os valores de atributos usados como entrada no algoritmo K-MEANS foram escalonados de acordo com a Equação (5), onde o valor de *zi* é dado pelo resultado da divisão do desvio padrão S pela média de um dado elemento do conjunto x.

$$X_i = z_i \frac{x_{i-\bar{x}}}{S}$$

A Tabela 11 exemplifica os atributos utilizados para alguns dos agrupamentos existentes, a Tabela com informações completas dos agrupamentos pode ser encontrada no apêndice b.

Tabela 11 - Grupos de casos formados visando identificar as combinações entre estados e ações de jogo.

Agrupamentos de casos	**Atributos utilizados como entrada na execução do algoritmo de clustering**
Primeira carta a ser jogada quando o agente é o primeiro a jogar	Carta alta recebida, carta média recebida, carta baixa recebida, carta jogada.
Quem pediu ENVIDO, quando o agente é o primeiro a apostar	Quem pediu envido, Quem pediu real envido, Quem pediu falta envido.
Quem pediu TRUCO antes de jogar a primeira carta quando o agente é o primeiro a jogar	Quem pediu truco, Quem pediu retruco, Quem pediu vale quatro.
...	...

Fonte: do autor.

Como exemplo dos diferentes agrupamentos de casos construídos nesta dissertação, os quatro grupos de casos formados para a modalidade de aposta TRUCO no cenário em que o jogador é o segundo a jogar a primeira carta são os seguintes:

- Aposta nível 2 (RETRUCO) com mão balanceada: neste grupo, as apostas de TRUCO são aumentadas até RETRUCO. Porém, o jogador não possui cartas tão fortes. Jogador 2 sempre inicia a aposta e o Jogador 1 realiza esse aumento da aposta.
- Mão fraca (de acordo com as categorias identificadas na Tabela 4) com pouca interação entre jogadores: raramente algum dos jogadores aposta. Quando existe uma aposta, ela não é aumentada.

- Mão boa com interações variadas entre jogadores: neste grupo, as cartas recebidas são altas. Existem interações entre os jogadores em um ciclo de apostas completo. Outras apostas neste grupo apenas permanecem no nível 1 (TRUCO).
- Mão boa com interação mínima entre jogadores: cartas bem balanceadas. O oponente nunca aumenta a aposta mínima realizada por um jogador.

Tabela 12 apresenta os resultados obtidos pelos agrupamentos resultantes da análise de grupos de casos para a ação jogar a primeira carta no cenário em que o jogador é o segundo a jogar.

Tabela 12 - Grupos de casos resultantes considerando a ação jogar a primeira carta quando o jogador é o segundo a jogar

Grupo de casos	**Porcentagem de casos no grupo**	**Características do grupo**
Tática utilizada para cartas com valores balanceados: uma das cartas pertence a uma das 3 categorias com valor mais alto dentre as existentes na Tabela 7, outra carta pertence a uma das 3 categorias centrais (“Brancas Altas”, “Brancas Médias, Negras Baixas ”) , e outra carta faz parte de um das 2 categorias mais baixas dentre as apresentadas na Tabela 7.	23,75 %	Jogador 1 joga a carta mais baixa, a qual é capaz de superar a carta jogada pelo jogador 2.
Tática de jogo	27,78%	A carta mais alta recebida faz parte

covarde: jogador que normalmente não aceita apostas quando não tem uma mão forte no jogo		da categoria de cartas brancas altas na maioria dos casos deste grupo. A carta média recebida tende a pertencer as categorias de cartas brancas médias ou negras baixas. A carta baixa recebida tende a fazer parte da categoria de cartas brancas baixas na maioria das ocorrências. A primeira carta nunca é jogada porque o jogador 1 sempre vai ao baralho devido a uma aposta do oponente.
Táticas relacionadas ao blefe	0,05%	Nos casos deste grupo, a tendência é encontrar mãos balanceadas. Mão é encerrada na primeira rodada, visto que um dos dois jogadores aposta e o adversário sempre recusa a aposta. Quando existe uma aposta aceita, o aumento da aposta é realizado e o oponente vai ao baralho.
Tática de jogo para mão fraca: mãos compostas por cartas com valor baixo no jogo (geralmente no mínimo duas cartas pertencem a uma das duas categorias mais baixas apresentadas na Tabela 4)	41,96 %	Jogador 1 joga a carta que tem o valor mais baixo possível para vencer a carta jogada pelo adversário. Quando não é possível vencer a carta jogada pelo adversário, o jogador joga a carta mais baixa recebida.

Fonte: do autor.

Uma aposta do tipo ENVIDO pode ser feita em dois possíveis cenários: quando o jogador inicia tendo o direito de apostar, e quando o jogador é o segundo a ter direito de apostar. Os grupos de casos formados pela execução do algoritmo K - *MEANS* para

o cenário em que o jogador é o segundo a ter o direito de apostar nessa modalidade de apostas na base de casos inicial (coletadas por humanos) são os seguintes:

- Aposta de nível dois com pontuação maior do que 26: em todos os casos neste grupo. Jogador 1 chama REAL ENVIDO logo o início do jogo, antes de fazer qualquer outra jogada. Em poucos casos onde o oponente faz a aposta de nível 1 (ENVIDO), o jogador 1 aumenta para a de nível 2;

- Interação inexistente entre jogadores: jogador 1 não tem pontos e nenhum dos dois oponentes realiza uma aposta;

- Com uma pontuação de ENVIDO muito alta na mão, jogador realiza uma aposta máxima sem seguir uma sequência de interações para apostas menores.

- Jogador realiza uma aposta mínima: jogador 1 tende a fazer a aposta mínima e o oponente não aumentar essa aposta. Neste grupo, o jogador sempre tem pontos na mão, os quais variam de pontuações baixas (por exemplo, 20 pontos) até pontuações razoavelmente altas (por exemplo, 33 pontos).

Para apoiar a computação de políticas de reuso que exploram esses grupos de casos formados, os quais alguns exemplos foram apresentados anteriormente, um atributo para representar o grupo que cada caso pertence foi incluído no modelo de casos usado. Esse atributo multivalorado representa o grupo que o caso pertence em cada uma das diferentes ações de jogo consideradas nas análises de agrupamentos de casos executadas nesta dissertação. Dessa forma, esse atributo atua como uma solução no modelo de representação de casos utilizado, capaz de representar o cenário de decisão que o caso está inserido em cada uma das diferentes ações de jogo executadas neste jogo.

4. ABORDAGENS UTILIZADAS NA CRIAÇÃO DE POLÍTICAS DE REUSO, REVISÃO DE DECISÕES E APRENDIZAGEM AUTOMÁTICA

Novas políticas de reuso de soluções, as quais são gravadas em casos recuperados para consultas de sistemas CBR aplicados a jogos de cartas, são propostas e testadas neste trabalho. Tais políticas de reuso construídas são resultantes da aplicação de critérios de reuso de acordo com diferentes modelos de reuso:

- Modelo de reuso em dois passos: em um primeiro passo, um critério de reuso é aplicado de acordo com os casos recuperados. O objetivo é usar esses casos recuperados para escolher qual grupo de casos deve ser explorado no apoio a construção de uma resposta de uma consulta. Após um grupo ser selecionado, em um segundo passo, um critério de reuso que somente utiliza os casos recuperados pertencentes ao grupo escolhido é executado novamente para selecionar uma ação de jogo a ser tomada pelo agente. Parte desse modelo foi apresentado em (Paulus *et al.*, 2019), e agora é expandido nesta dissertação de forma que diferentes critérios de reuso possam ser executados nos 2 passos de reuso citados.

- Modelo de reuso convencional: um critério de reuso é diretamente computado usando as soluções gravadas nos casos recuperados. O objetivo é selecionar a ação de jogo que deve ser reusada e consequentemente tomada pelo agente. Tal modelo de reuso é descrito em (Rubin e Watson, 2010).

As possíveis combinações de diferentes critérios de reuso na escolha de grupos de casos e ações de jogo são variadas. Este modelo de reuso em dois passos é possível uma vez que algoritmos de agrupamento de dados permitem analisar os casos da base de casos e identificar como os jogadores se comportam em diferentes situações/cenários de jogo. Entre outros objetivos, este modelo de reuso em dois passos visa identificar que ações de jogo essencialmente tomadas em diferentes cenários de jogo não sejam contabilizadas como jogadas similares. Por exemplo, um jogador pode possuir cartas que pertencem a categorias de cartas com valores altos ou cartas que pertencem a categorias de cartas com valores baixos, e em ambas as situações, o jogador jogar a carta mais alta dentre as que possui na mão corrente. Em muitos sentidos, o ato de jogar a carta mais alta pode ser considerado como uma jogada similar nestes diferentes cenários de jogo, embora as configurações das mãos sejam relativamente diferentes. Se a consulta executada não for capaz de realizar essa distinção entre as diferentes mãos do jogador, a política de reuso usada consideraria que jogar a carta mais alta seria a melhor ação de jogo a ser usada. Contudo, se os casos representando essas mãos de jogo fossem organizados em grupos de casos diferentes, a política de reuso usada poderia identificar que uma ação de jogo diferente poderia ser a melhor opção a ser executada pelo agente. De maneira similar, uma mesma ação de aposta tem diferentes consequências de acordo com as diferentes interações de apostas realizadas pelos jogadores. Tal como identificado em um processo de análise de grupos envolvendo os casos da base de casos,

grupos diferentes podem capturar casos com interações de jogo similares entre si, e dissimilares de outros casos organizados em outros grupos. Se a configuração das consultas executadas no sistema CBR não puder capturar explicitamente essas diferentes interações de jogo, os casos recuperados podem conter interações de jogo diferentes e, consequentemente, o modelo de reuso convencional também não consideraria esse fato na computação de soluções reusadas para permitir a solução do problema de jogo corrente.

Algoritmo 1 apresenta o modelo de reuso convencional. Esse algoritmo recebe como entrada os casos armazenados na base de casos, o caso usado como consulta que representa o estado atual do jogo (problema atual a ser resolvido), a similaridade mínima (*threshold* de similaridade) para um caso ser considerado similar ao problema atual, e o critério de reuso que deve ser utilizado para escolher a ação de jogo a ser executada conforme análise das soluções gravadas nos casos recuperados. Após filtrar os casos que são mais similares que o *threshod* estabelecido, não considerando os casos que possuem valores de similaridade menores que o valor do *threshold* determinado, um critério de reuso é aplicado para escolher qual ação de jogo deve ser reusada. Desta forma, a escolha da solução a ser retornada é realizada por meio da análise de todos os casos mais similares recuperados para a consulta dada.

Algoritmo 1 – Modelo de reuso convencional

Algoritmo 1 : Modelo de reuso convencional

```
1. data: Given: CB, case base; query, query considering the current game problem
situation; st, similarity threshold; aReuseCriteria, reuse criteria ∈ {MJ, PL,
NP, PV}
2. result: selectedSolution, game action selected.
3.      begin
4.          R = {case1,case2,…,case_N  ∈ CB | ( sim(query, case_n) > st )}
5.          selectedSolution = chooseSolution(R, aReuseCriteria)
6.          return selectedSolution
7.      end
```

Fonte: pseudocódigo adaptado a partir de (Rubin e Watson, 2010)

Diferente do processo de reuso convencional, Algoritmo 2 detalha um modelo de reuso em dois passos. Esse algoritmo recebe como entrada os casos contidos na base de casos, uma lista com a indicação dos clusters que estão representados nesses casos recuperados (não são considerados nas computações os clusters que nenhum caso recuperado esteja presente), a consulta que captura o estado atual do jogo, a similaridade mínima que cada caso deve possuir em relação a consulta para que a solução gravada nestes casos seja considerada no processo de reuso, o critério de reuso que deve ser utilizado para escolher um dos clusters representado nos casos recuperados (identificado como critério de reuso extra cluster) e o critério de reuso que deve ser utilizado para escolha de solução para o problema atual (identificado como critério de reuso intra cluster). É importante observar que um critério de reuso é executado para escolher um cluster dentre os clusters presentes nos casos recuperados, os quais devem possuir uma similaridade com a consulta maior ou igual ao *threshold* estabelecido (linha 1). Em uma tarefa de filtragem executada a seguir, são mantidos na lista de casos recuperados e considerados no processo de reuso de soluções apenas os casos que pertencem ao cluster escolhido (linha 6). Usando essa lista, um critério de reuso é usado para escolher a solução que deve ser utilizada na resposta do problema atual.

Algoritmo 2 – Modelo de reuso em dois passos explorado por políticas de reuso propostas nesta dissertação

Algoritmo 2 : Modelo de reuso em dois passos

```
1. data: Given: CB, case base; C: clusters in the cases;  query, query
   considering the current game problem situation; st, similarity threshold;
   aReuseCriteria, reuse criteria ∈ {MJ, PL, NP, PV}; cReuseCriteria, reuse
   criteria ∈ {MJ, PL, NP, PV}
2. result: selectedSolution, game action selected.
3.     begin
4.         R = {case1,case2,…,case_N  ∈ CB | ( sim(query, case_n) > st )}
5.         selectedCluster = chooseCluster(R, C, cReusePolicy)
6.         R' = {case_n ∈ R | case_n.clusterLabel == selectedCluster}
7.         selectedSolution = chooseSolution(R', aReusePolicy)
8.         return selectedSolution
9.     end
```

Fonte: do autor.

Nesta dissertação, diferentes critérios de reuso podem ser utilizados tanto para escolha do grupo a ser utilizado para filtrar os casos que serão considerados na escolha da solução quanto para escolha das ações de jogo a serem reusadas a partir de uma lista filtrada/reduzida de casos recuperados. Dessa forma, a palavra *ITEMS* nos Algoritmos 3, 4, 5 e 6 é utilizada para identificar clusters ou soluções, já que os critérios de reuso propostos podem ser utilizados para a escolha de ambos. Nesta dissertação, existem 2 diferentes modelos de reuso (modelo de reuso convencional e modelo de reuso em dois passos), 4 diferentes critérios de reuso, onde a aplicação de critérios de reuso em algum dos modelos de reuso existentes resulta em uma política de reuso.

4.1.1. CRITÉRIOS DE REUSO EXPLORADOS

Nesta dissertação, 4 diferentes critérios de reuso foram explorados, a utilização de diferentes critérios permite a seleção de ITEMS (clusters ou soluções) com diferentes características. Por possuir diferentes objetivos na seleção dos casos, a combinação entre os diferentes critérios de reuso proporciona diferentes resultados. Para esta pesquisa, foram revisados na literatura, critérios de reuso geralmente aplicados em jogos de cartas. Dessa forma, os critérios encontrados foram adaptados para o jogo de Truco e combinados para construir diferentes propostas de políticas de reuso. As diferentes abordagens foram:

- O critério de reuso que computa a regra da maioria - *majority-rules – MJ*
- Loteria baseada em probabilidades - *probability lottery* – PL
- Probabilidade de vitória - *probability victory* – PV
- Número de pontos - *number of points* - NP

O critério de reuso que computa a regra da maioria - *majority-rules – MJ* obteve o melhor desempenho para escolha de ações de jogo em (Rubin e Watson, 2010). O

critério MJ seleciona o cluster / solução contida (o) na maioria dos casos similares recuperados. Quando aplicado diretamente na escolha de uma solução para resolver um problema atual, o critério MJ é identificado como solução da maioria - *majority solution* - *MJS*. Na escolha de grupos de casos, a utilização deste critério para escolher um grupo foi identificada como solução da maioria com clusters – *majority rules with cluster* - MJC. Algoritmo 3 apresenta o critério de reuso MJ, onde uma lista de clusters ou de soluções é usada como entrada. Isso permite computar o número de ocorrências de cada um dos clusters ou soluções existentes na lista de entrada. De acordo com essa computação, a ação de jogo ou cluster de casos com maior número de ocorrências de acordo com os casos analisados é retornado.

Algoritmo 3 – Critério de reuso que computa a regra da maioria - *majority-rules* - MJ

Algoritmo 3 : MJ (Majority rules)

```
1. data:  ITEMS, list of existing clusters or actions; POSSIBLEITEMS, list of
   possible clusters or actions.
2. result: selectedItem, solution or cluster selected.
3. begin
4.    initialize
5.     incrementItems[] = {0}; //increment existing clusters or actions
6.     highestAmount = 0;
7.    foreach pi in POSSIBLEITEMS do
8.      foreach i in ITEMS do
9.         if  pi ==   i  then
10.           incrementItems[pi] ++
11.        end if
12.     end foreach
13.   end foreach
14.      foreach pi in POSSIBLEITEMS do
15.         if  length(incrementItems[pi])  > highestAmount     then
16.           highestAmount = length(incrementItems[pi])
17.           selectedItem =  pi
18.       end if
19.      end foreach
20.     return selectedItem
21. end
```

Fonte: do autor.

Em (Rubin e Watson, 2007; Rubin e Watson, 2010), também foi apresentado um critério de reuso baseado na computação de probabilidades. O objetivo deste critério é computar a probabilidade de cada cluster / solução ser escolhida de acordo com os casos recuperados. A partir disso, um método de sorteio usa esses valores de probabilidade para computar as chances de cada cluster ou solução visto que a probabilidade de cada um destes itens ser sorteado é considerada. Este critério é identificado como loteria baseada em probabilidades - *probability lottery* - PL. A aplicação do critério PL na escolha da solução é identificada como loteria baseada em probabilidades de soluções - *probability lottery solution* - PLS. A aplicação deste critério para a escolha do grupo de casos no modelo de reuso em dois passos é identificada como loteria baseada em probabilidades com clusters - *probability lottery with cluster* - PLC. Algoritmo 4 apresenta o critério de reuso PL, onde uma lista de clusters ou de soluções é usada como entrada. Após, o número de ocorrências de cada cluster ou solução encontrada na lista de entrada é computada. Os valores quantitativos resultantes dessas computações são

aplicados em uma função de sorteio onde as chances de cada cluster ou solução serem escolhidos são consideradas com base nos quantitativos obtidos.

Algoritmo 4 – Critério identificado como loteria baseada em probabilidades - *probability lottery* - PL

Algoritmo 4 : PL (probability lottery)

```
1. data:  ITEMS, list of existing clusters or actions; POSSIBLEITEMS, list of
   possible clusters or actions.
2. result: selectedItem, solution or cluster selected.
3. begin
4.    initialize
5.     incrementItems[] = {0}; //increment existing clusters or actions
6.     probability[] = {0}; //probability clusters or actions to be selected
7.      foreach pi in POSSIBLEITEMS do
8.       foreach i in ITEMS do
9.         if  pi ==   i  then
10.           incrementItems[pi] ++
11.       end if
12.     end foreach
13.   end foreach
14.      foreach pi in POSSIBLEITEMS do
15.         probability[pi] = length (ITEMS) / incrementItems[pi]
16.      end foreach
17. selectedItem =  random(ITEMS, probability[])
18.     return selectedItem
19. end
```

Fonte: do autor.

Para criar políticas de reuso capazes de selecionar as ações de jogo que proporcionem maiores chances de sucesso no jogo, uma mesma política de reuso integra a) a técnica que propõe utilizar consequências no jogo resultantes da escolha de uma determinada ação de jogo conforme apresentado em (Sandven e Tessem, 2006) com b) a técnica que propõe utilizar probabilidade na escolha de soluções conforme apresentada em (Rubin e Watson, 2007; Rubin e Watson, 2010). A união dessas técnicas resultou na criação de um critério de reuso denominado probabilidade de vitória - *probability victory* - PV. Neste critério, a quantidade de vitórias que cada solução ou cluster alcançou no jogo é considerada. Esse critério retorna a ação de jogo ou cluster que possui maior chance de sucesso de acordo com os casos recuperados. Quando aplicado para escolher qual solução deve ser utilizada, essa política de reuso é denominada probabilidade de vitória de uma solução - *probability victory solution* - PVS. Ela é denominada como probabilidade de vitória com clusters - *probability victory with cluster* - PVC quando ela é utilizada para escolher o grupo de casos que deve ser analisado na resposta de uma consulta. Algoritmo 5 apresenta o critério de reuso PV, onde após computar o número de ocorrências de cada item (cluster ou ação de jogo) existente, é também verificado o número de casos de sucesso (neste caso, vitórias) para cada uma das diferentes ações de jogo existentes nos casos recuperados. Dessa forma, é sugerida a solução / cluster que possui o maior número de vitórias.

Algoritmo 5 – Critério de reuso denominado probabilidade de vitória - *probability victory* - PV

Algoritmo 5 : PV (probability victory)

```
1. data:  ITEMS, list of existing clusters or actions; POSSIBLEITEMS, list of
   possible clusters or actions.
2. result: selectedItem, solution or cluster selected.
3. begin
4.    initialize
5.     incrementItems [] = {0}; //increment existing clusters or actions
6.     incrementVictory [] = {0}; //increment existing clusters or actions
7.     probabilityVictory[] = {0}; // probability cluster or action
8.     highestProbabilityVictory = 0;
9.      foreach pi in POSSIBLEITEMS do
10.      foreach i in ITEMS do
11.       if  pi ==   i  then
12.              incrementItems[pi] ++
13.           if   i.hasWon == TRUE
14.                incrementWin[pi]++
15.           end if
16.       end if
17.     end foreach
18.   end foreach
19.      foreach pi in POSSIBLEITEMS do
20.         probabilityVictory[pi] = length (incrementItems[pi]) / length
   (incrementVictory[pi])
21.      end foreach
22.      foreach pi in POSSIBLEITEMS do
23.         if probabilityVictory[pi] > highestProbabilityVictory then
24.             highestProbabilityVictory = probabilityVictory[pi]
25.             selectedItem = pi
26.         end if
27.      end foreach
28.     return selectedItem
29.      end
```

Fonte: do autor.

Ações de jogo que envolvem apostas têm o objetivo de conquistar pontos no jogo de Truco. Além de possuir consequências de vitória ou fracasso, essas ações também têm como consequência uma quantidade de pontos conquistados ou perdidos. Tal como apresentado em (Sandven e Tessem, 2006; Rubin e Watson, 2010), o valor perdido ou conquistado em relação a cada uma das diferentes ações de jogo consideradas é armazenado como um atributo na representação de um caso. Baseado nesta ideia, foi criado um critério de reuso que considera o saldo de pontos que cada ação de jogo / cluster representada nos casos recuperados para uma consulta. É importante salientar que esse critério foi construído exclusivamente para ações de jogo envolvendo apostas, visto que são as ações de apostas que interferem em consequências deste tipo (quantidade de pontos ganhos ou perdidos). Dessa forma, o critério explora a ideia proposta pelo critério de probabilidade vitória - *probability victory* - PV para ações de jogo que possuem apenas consequências binárias. Esse critério de reuso é identificado como número de pontos - *number of points* - NP. Quando esse critério é utilizado para escolher uma ação de jogo a ser reusada, ele é identificado como número de pontos para solução - *number of points solution* - NPS. Quando empregado para escolher o grupo onde as soluções gravadas nos casos recuperados devem ser analisadas, esse critério é identificado como número de pontos com clusters - *number of points* with *cluster* - NPC. Como descrito no Algoritmo 6, é computada a soma dos ganhos/perdas proporcionados

por cada ação ou cluster. Dessa forma, o cluster ou ação de jogo que possibilitou maiores ganhos é retornado.

Algoritmo 6 - Critério de reuso identificado como número de pontos - *number of points* - NP

Algoritmo 6 : NP (number of points)

```
1. data:  ITEMS, list of existing clusters or actions; POSSIBLEITEMS, list of
   possible clusters or actions.
2. result: selectedItem, action done in selected cluster.
3. begin
4.    initialize
5.     incrementItems [] = {0}; //increment existing clusters or actions in
   cases
6.     numberOfPoints[] = {0}; // numberOfPoints
7.     highestNumberOfPoints = 0;
8.      foreach pi in POSSIBLEITEMS do
9.       foreach i in ITEMS do
10.       if pi ==   i then
11.            incrementItems[pi] ++
12.            numberOfPoints[pi] = i.numberOfPointsWon
13.       end if
14.     end foreach
15.   end foreach
16.      foreach pi in POSSIBLEITEMS do
17.         if numberOfPoints[pi] > highestNumberOfPoints then
18.             highestNumberOfPoints = numberOfPoints[pi]
19.             selectedItem = pi
20.         end if
21.      end foreach
22.     return selectedItem
23. end
```

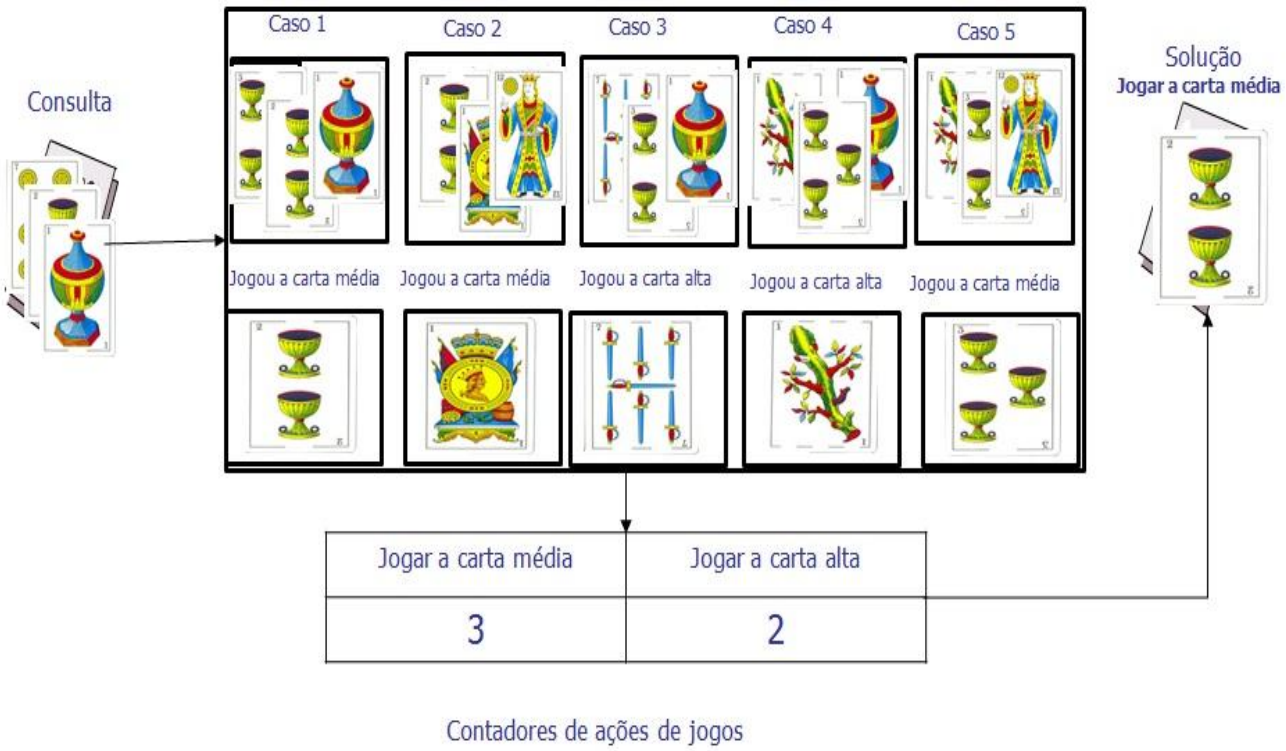
Fonte: do autor.

Para demonstrar como os critérios de reuso são aplicados nos dois modelos de reuso explorados nesta dissertação, exemplos de mãos de Truco podem ser usados para ilustrar a escolha de ações de jogos. Figuras 6 e 7 representam casos recuperados para uma dada consulta. O exemplo proposto ilustra uma consulta onde o agente precisa determinar qual carta jogar na primeira rodada no cenário de jogo em que o agente é o primeiro a jogar.

Na Figura 6 um exemplo do critério MJ (*majority rules*) é apresentado. Neste exemplo, MJ é diretamente aplicado no reuso de uma solução gravada em um conjunto de casos recuperados. É possível observar que a ação de jogo que mais frequentemente ocorre nos casos recuperados é jogar a carta média. Logo, a solução escolhida pela maioria dos casos recuperados (MJ) foi jogar a carta média. Porém, os estados de jogo apresentados nos três casos recuperados que jogaram a carta média não é o mesmo. Por exemplo, a carta baixa do caso 1 tem o mesmo valor que a carta média do caso 2, embora ambas tenham sido codificadas como cartas médias. Por sua vez, a carta média do caso 5 é maior do que a carta alta do caso 2. Além disso, essa carta média do caso 5 é a mesma que a carta alta do caso 1. Na prática, o critério de reuso MJ aplicado diretamente nas ações de jogo existentes nos casos recuperados considera tais jogadas como idênticas mesmo que elas tenham sido executadas em estados de jogo diferentes. Quando isso

acontece, uma ação de jogo possivelmente incorreta poderia ser reusada e executada pelo agente.

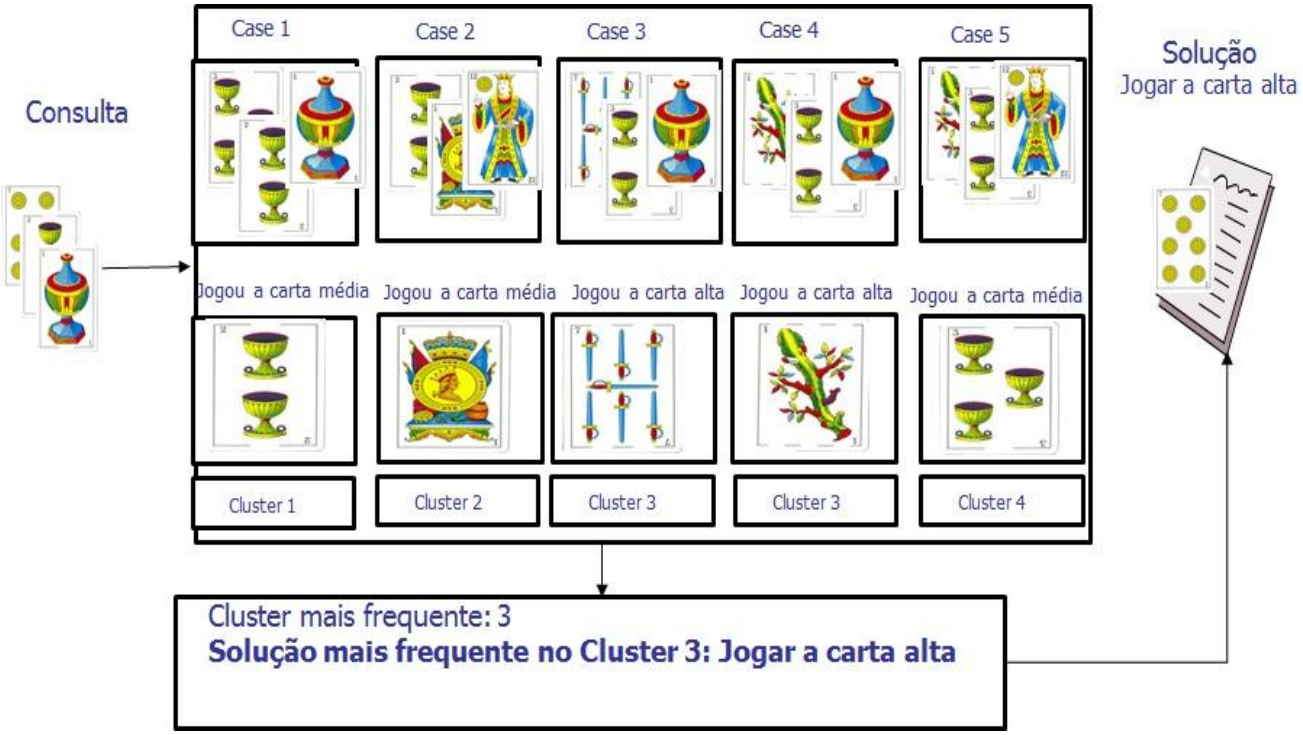
Figura 6 – Exemplo do critério de reuso MJ aplicado diretamente na escolha da ação de jogo.



Fonte: do autor.

Figura 7 apresenta um exemplo do critério de reuso MJ aplicado no modelo de reuso em dois passos, tanto para escolha do cluster quanto para escolha da solução a ser reusada na resolução de uma consulta. Neste exemplo, é possível verificar que os três casos que contribuíram para a escolha da solução no exemplo anterior são agora identificados como pertencendo a diferentes clusters de casos. No exemplo, os casos 3 e 4 fazem parte do cluster 3. Como o critério de reuso utilizado é a votação da maioria MJ, apenas os casos que pertencem ao cluster 3 são mantidos no processo de reuso. Como nesses dois casos (casos 3 e 4) a carta jogada é a carta alta, o critério de maioria resulta na escolha da carta alta como sugestão de ação de jogo a ser executada. Apesar destes exemplos utilizarem o mesmo critério de reuso nos dois passos (escolha do cluster e escolha da solução), diferentes combinações entre critérios são propostas e experimentadas nesta dissertação.

Figura 7 – Exemplo do critério MJ aplicado em nosso modelo de reuso de dois passos, tanto para escolha do cluster quanto para escolha da ação de jogo.



Fonte: do autor.

Cada uma das diferentes políticas de reuso, descritas nesta dissertação, prioriza a escolha de diferentes comportamentos de jogo. Porém, ainda não é possível detalhar quais políticas seriam capazes de alcançar os melhores resultados em partidas disputadas. Dessa forma, diferentes combinações possíveis entre critérios de reuso para escolha dos clusters e seleção de soluções foram implementadas neste trabalho. Tais combinações permitiram a identificação de 20 diferentes políticas de reuso. Essas políticas podem ser divididas em três diferentes categorias de acordo com a forma que os critérios de reuso são empregados:

- Critérios de reuso aplicados diretamente na escolha da solução (modelo de reuso convencional). Esses agentes representam políticas de reuso propostas na literatura (Sandven e Tessem, 2006; Rubin e Watson, 2010). Ao todo, o uso destas políticas de reuso permitiu a implementação de 4 diferentes agentes:

Tabela 13 – Agentes criados com o modelo de reuso convencional.

Nome	Escolha da solução

MJS	MJ (*majority rules*)
PLS	PL (*probability lottery)*
PVS	PV (probability victory)
NPS	NP (*number of points*)

Fonte: do autor.

- Políticas de reuso onde o mesmo critério de reuso é aplicado duas vezes de acordo com o modelo de reuso de casos em dois passos, tal como apresentado em (Paulus *et al.*, 2019) (modelo de reuso em dois passos). Ao todo, o uso destas políticas de reuso permitiu a implementação de 4 diferentes agentes:

Tabela 14 – Agentes com o mesmo critério de reuso nos dois passos de recuperação utilizados pelo modelo de reuso proposto

Nome	Escolha do cluster	Escolha da solução
MJCS	MJ (*majority rules*)	MJ (*majority rules*)
PLCS	PL (*probability lottery)*	PL (*probability lottery)*
PVCS	PV (probability victory)	PV (probability victory)
NPCS	NP (*number of points*)	NP (*number of points*)

Fonte: do autor.

- Políticas de reuso onde diferentes critérios de reuso são combinados na escolha de clusters e soluções. Essa abordagem permitiu a implementação de 12 diferentes agentes.

Tabela 15 – Agentes com diferentes combinações de critérios de reuso no modelo de reuso em dois passos proposto.

Nome	Escolha do cluster	Escolha da solução

NPCMJS	NP (*number of points*)	MJ (*majority rules*)
NPCPVS	NP (*number of points*)	PV(*probability victory*)
NPCPLS	NP (*number of points*)	PL (*probability lottery*)
MJCNPS	MJ (*majority rules*)	NP (*number of points*)
MJCPVS	MJ (*majority rules*)	PV(*probability victory*)
MJCPLS	MJ (*majority rules*)	PL (*probability lottery*)
PLCNPS	PL(*probability lottery*)	NP (*number of points*)
PLCMJS	PL (*probability lottery*)	MJ (*majority rules*)
PLCPVS	PL (*probability lottery*)	PV(*probability victory*)
PVCNPS	PV(*probability victory*)	NP (*number of points*)
PVCMJS	PV(*probability victory*)	MJ (*majority rules*)
PVCPLS	PV (*probability victory*)	PL (*probability lottery*)

Fonte: do autor.

A aplicação dos diferentes critérios de reuso nos dois modelos possíveis (modelo de reuso convencional e modelo de reuso em dois passos) permitiu a criação de políticas de reuso capazes de priorizar a análise de diferentes características de jogo existentes nos casos. Apesar de investigar a capacidade de políticas de reuso resultarem em boas decisões em disputas de jogo de Truco, o objetivo é também investigar a criação de agentes virtuais através da utilização de logs de jogos provenientes de jogadores com diferentes níveis de habilidade. Para isso, a implementação de uma etapa de revisão de soluções também se fez necessária no desenvolvimento desta dissertação.

4.1.2. REVISÃO DE DECISÕES

De acordo com (Winne, 2017), o jogo de Truco é repleto de jogos mentais e habilidades provindas da aquisição de experiências pelos jogadores. Portanto, existem alguns tipos de jogadas que não podem ser reproduzidas por dois diferentes motivos:

- Em bases de casos onde existem casos gerados por jogadores com pouca experiência de jogo, ações de jogo consideradas incorretas tomadas por esses jogadores novatos podem resultar em uma grande perda de pontos no jogo de Truco.
- Jogadores experientes são capazes de prever o comportamento de jogo do oponente de acordo com observações feitas no decorrer de uma partida sendo disputada. Nestas situações, esses jogadores são capazes de identificar o tipo de jogada que deve ser tomada de acordo com o comportamento de jogo do oponente. Por este motivo, jogadores com esta capacidade podem criar experiências de solução de problemas (casos) que podem ser úteis apenas em contextos de jogo muito específicos. Por exemplo, o jogador está com uma mão em que a chance de vencer é muito pequena e as consequências negativas de ações de jogo envolvendo apostas podem ser muito altas. Mesmo diante deste cenário, este jogador realiza uma aposta arriscada porque detectou no decorrer da partida que o oponente tende a negar esse tipo de aposta.

Com o objetivo de evitar que comportamentos extremos sejam contabilizados pelas políticas de reuso propostas e ao mesmo tempo permitir que as ações de jogo selecionadas pelo reuso sejam mantidas na grande maioria das situações de jogo, jogadas muito extremas onde as chances de fracasso são muito grandes foram revisadas. Para isso, uma lista de comportamentos de jogo identificados como “suicidas” foi criada e explorada pelos agentes implementados. Dessa forma, sempre que uma jogada é sugerida de acordo com a computação de uma política de reuso, é verificado se essa jogada não faz parte da lista definida. As jogadas consideradas na lista podem ser exemplificadas por:

a) Apostas lógicas que deixaram de ser feitas: quando jogadores humanos jogam, por diferentes motivos (conhecer muito bem o oponente ou não ter muita experiência) algumas jogadas consideradas lógicas são deixadas de ser feitas, por exemplo:

- O jogador é o segundo a jogar a terceira carta, a carta na mão é maior do que a jogada pelo oponente e a aposta não é aumentada.

b) Blefes compulsivos: algumas ações de blefes precisam ser feitas com critérios, e geralmente estão relacionadas a identificação do comportamento do oponente durante o jogo, por isso, jogadas de blefe que possuem uma alta chance de fracasso foram filtradas, por exemplo:

- O jogador é o primeiro a apostar, não tem nada de pontos e faz alguma aposta relacionada ao ENVIDO.

c) Cartas erradas jogadas, por exemplo:

- O oponente joga uma carta (em qualquer rodada) e o jogador larga a carta mais alta da mão para superar uma carta que poderia ter sido superada com a carta de valor médio (caso ainda esteja disponível), ou larga a carta de valor médio da mão para fazer uma carta que poderia ter sido superada com a carta de valor baixo da mão (caso ainda não tenha sido jogada).

4.1.3. APRENDIZAGEM AUTOMÁTICA DA BASE DE CASOS

A competência de sistemas CBR em resolver problemas está diretamente relacionada com a competência observada nos casos armazenados na base de casos usada por esses sistemas. Do ponto de vista da base de casos, tal noção de competência em sistemas CBR indica que uma base com maior cobertura de casos tende a proporcionar a tomada de melhores decisões (Floyd *et al.*, 2008). Em resumo, a base de casos é um dos principais repositórios de conhecimento destes sistemas (Richter e Weber, 2013).

Esta etapa foi desenvolvida neste trabalho com dois objetivos principais:

- Sobrepor a dificuldade de coletar experiências: em alguns tipos de aplicações é difícil conseguir uma grande quantidade de amostras criadas por seres humanos.
- Validar políticas de reuso em dois passos em situações não tão favoráveis: com uma maior quantidade de casos, é possível considerar apenas casos com uma similaridade mínima alta ao problema atual. Ao utilizar apenas experiências bem

similares ao problema em questão, diminuí o número de experiências que pertencem a diferentes estados de decisão. Com isso, a utilização de um *threshold* mais alto tende a beneficiar as políticas de reuso convencionais.

Para possibilitar que agentes armazenem e explorem novas experiências de jogo (nesta dissertação capturadas como mãos de jogo de Truco disputadas entre pares de oponentes) e que aprendam com erros e acertos observados em partidas de Truco, um modelo de aprendizagem da base de casos baseado em *self play* foi explorado nesta dissertação. Em geral, esse processo de *self play* permitiu usar uma base de casos de 3195 casos, inicialmente coletada a partir de partidas de Truco disputadas por humanos, para expandir significativamente essa base para um total de 27515 casos. Para isso ser possível, foi necessário disputar 1833 partidas de Truco na modalidade de *self play*. Todo esse processo de treinamento foi desenvolvido em um único computador, por isso demorou várias semanas sendo executado de forma ininterrupta (embora a implementação de pequenas otimizações nesses algoritmos de aprendizado possa resultar em menores tempos de aprendizado). Além disso, a base de casos resultante também foi analisada manualmente de forma a permitir identificar casos que pudessem ser removidos da base por diferentes motivos (em geral, foram efetuadas atividades de limpeza de dados, tal como descritas em processos de pré-processamento em mineração de dados). Entre os critérios usados nesse processo de limpeza da base de casos, os critérios de revisão descritos na seção 4.1.2 desta dissertação foram explorados para determinar quais casos não deveriam ser mantidos na base de casos.

Para obter uma resposta satisfatória para uma consulta emitida para apoiar o processo de tomada de decisão de um agente, foi definido que o processo de recuperação de casos usado para resolver essa consulta durante as partidas de *self play* deveria retornar no mínimo 100 casos da base de casos. Além disso, todos esses 100 casos retornados deveriam ter no mínimo 98% de similaridade (*threshold* de recuperação usado) com a consulta. No final, este mesmo critério está sendo empregado nos algoritmos de CBR utilizados nesta dissertação. Enquanto esse critério de aprendizagem definido não foi satisfeito a partir da avaliação dos resultados das consultas executadas, mãos de Truco disputadas na modalidade *self play* foram gravadas como novos casos na base de casos. Logo, novas consultas executadas pelos agentes já poderiam receber como resultado casos adicionados na base de casos como resultado da análise de consultas anteriormente executadas neste processo de treinamento. Na prática, novos

casos foram gerados e adicionados frequentemente na base de casos em partidas disputadas em momentos iniciais desse processo de *self play*, e eram adicionados com muito menor frequência do que em partidas disputadas em fases mais adiantadas desse processo de treinamento.

No processo de treinamento visando a aquisição de novos casos, diferentes políticas de reuso foram utilizadas pelos agentes empregados nas partidas de *self play* disputadas. Em cada partida, tais políticas de reuso foram selecionadas de forma aleatória e usadas por cada um dos agentes utilizados. Isso permitiu coletar partidas de Truco disputadas entre oponentes com diferentes características, visto que cada agente oponente poderia estar empregando um diferente critério de reuso proposto nesta dissertação. Em geral, a ideia foi construir uma base de casos de tamanho mais significativo armazenando experiências de solução de problemas diversificadas, assim podendo representar capacidades de solução de problemas de agentes com características distintas. Na criação das novas experiências, um dos dois agentes (sorteados aleatoriamente para criar as experiências) era escolhido para persistir os casos na base. Desta maneira, o agente que persistiu a experiência na base foi considerado como jogador 1 (jogador robô) e o agente oponente como jogador 2 (jogador humano).

Quando políticas de reuso que utilizam clusters são computadas para apoiar a captura de novos casos a serem adicionados na base de casos, é necessário que cada novo caso seja atribuído aos grupos de casos definidos para cada um dos 14 cenários de decisão existentes no jogo de Truco, como também nos dois diferentes grupos de casos usados na indexação da base de casos (usados na otimização do tempo de resposta de consultas). Na prática, tais organizações de casos em grupos são diretamente usadas na computação de soluções para consultas de acordo com as políticas de reuso propostas. Para isso, o atributo do modelo de casos usado que representa o grupo que o caso pertence para cada um dos cenários de jogo (decisão a ser tomada no jogo) no jogo de Truco foi preenchido com a identificação do grupo cujo centroide é o mais similar com o novo caso a ser armazenado na base de casos. Porém, a medida que novos casos foram armazenados na base de casos, o número de grupos de casos formados (e a composição destes grupos) alterou de várias formas. Isso indica que tais agrupamentos de casos deveriam ser redefinidos/reorganizados a medida que novos casos eram adicionados na base de casos.

Para resolver o problema que envolve a reorganização automática de casos em diferentes grupos durante o processo de treinamento realizado, derivadas de primeira ordem foram computadas para identificar a taxa de variação de cada ponto K em um conjunto de pontos usado como entrada na construção de um plot Elbow. Usando esses valores de derivada, foi possível determinar automaticamente o ponto *Elbow* sem a necessidade de plotar o gráfico. No processo de treinamento baseado em *self play*, essa análise de número e composição de grupos foi realizada automaticamente ao final de cada partida disputada. Caso o ponto *Elbow* identificado fosse diferente do número de grupos utilizado no agrupamento atualmente utilizado nas partidas de Truco, o algoritmo *K – MEANS* foi reexecutado usando como entrada o novo número de grupos identificado. Isso permitiu construir novos agrupamentos de casos sempre que necessário, os quais foram utilizados em partidas subsequentes disputadas no processo de aprendizado.

Na prática, a identificação do número de clusters *K* em tempo de execução das partidas de self play é realizada a partir de uma sequência de passos. Uma vez que valores de WCSS são calculados usando os casos e grupos formados nas novas execuções do algoritmo K-MEANS, para cada um dos número de grupos K(s) obtidos, os seguintes passos foram ser realizados:

1. Primeiro, normalizar os resultados de *WCSS* através da Equação 6:

$$WSS_k = \frac{WCSS_k}{WCSS_{max}}$$

2. Posteriormente, calcular a derivada de cada um dos resultados de *WCSS* usando a Equação 7:

$$D_k = \frac{WCSS_K[K+1] - WCSS_K[K]}{K[K+1] - K[K]}$$

3. Todos os pontos onde a derivada atual do ponto K e a derivada do ponto anterior k-1 possuem o mesmo sinal são submetidos à Equação 8:

$$DD_k = \left|\frac{D_k}{D_{k-1}}\right|$$

4. Por último, o ponto K que possuir o menor resultado na Equação 9 é o ponto que descreve o número de clusters K a ser escolhido e usado no algoritmo K-MEANS.

$$\min DD_K = \text{MIN}(DD_K)$$

Em resumo, para superarmos a dificuldade de coleta de casos, desenvolvemos um modelo de aprendizado automático, no qual a imitação de experiências passadas foi utilizada para adicionar novas experiências na base de casos. Porém, para avaliar as alterações do número de grupos de comportamentos existentes na base de casos em tempo de execução, derivadas do resultado do valor WCSS de cada um dos índices k testados, foram utilizados para avaliar alterações no número de grupos de comportamentos existentes. A aplicação desta proposta possibilitou aumentarmos a base de casos de 3195 para 27515 casos. Dessa forma, fomos capazes de testar nossas abordagens em uma situação aonde vários casos são disponíveis.

5. EXPERIMENTOS E RESULTADOS

A metodologia de teste utilizada para analisar a efetividade das técnicas implementadas nesta dissertação é baseada na proposta de competição experimental descrita pela ACPC (*Annual Computer Poker Competition*) (Rubin e Watson, 2012). De acordo com essa proposta, os agentes implementados empregam partidas duplicadas na disputa de jogos de Truco. Para ser justo na análise dos resultados dessa competição entre agentes implementados de acordo com diferentes técnicas, um dado conjunto de cartas A é dado a um jogador J e um conjunto de cartas B é dado a um jogador K. Após o término dessa partida, os conjuntos de cartas recebidos por jogadores J e K são armazenados e a partida é repetida agora utilizando os conjuntos invertidos de cartas. Neste caso, o jogador J recebe o conjunto de cartas B e o jogador K recebe o conjunto de cartas A. Ao final das partidas, as cartas recebidas por cada jogador são consideradas equivalentes, o que permite usar um mesmo conjunto de cartas para avaliar a efetividade desses agentes. Além desses testes baseados em partidas duplicadas, a efetividade dos agentes implementados foi analisada usando um conjunto de partidas de Truco disputadas contra jogadores humanos. Para realizar estes testes, um sistema web desenvolvido para permitir coletar partidas de Truco foi adaptado de forma que jogadores humanos pudessem jogar contra agentes.

5.1. PLANO DE TESTES

Por possuir um total de 20 agentes, os testes foram organizados em 4 etapas:

Etapa 1 - Todos contra todos: nessa etapa, os 20 agentes desenvolvidos foram usados na disputa das partidas duplicadas, onde todos jogaram contra todos. Além disso, o desempenho dos agentes construídos a partir do modelo de reuso em dois passos proposto nesta dissertação foi comparado ao desempenho de agentes implementados utilizando técnicas propostas na literatura (Rubin e Watson, 2010; Paulus *et al.*, 2019). A partir desses testes, os agentes que venceram mais que 50% das partidas disputadas foram selecionados de forma que outros testes pudessem ser realizados nas técnicas utilizadas na implementação desses agentes. Utilizando a base de casos inicialmente coletada com jogadores humanos, a qual continha 3195 casos, essa etapa inicial de testes foi realizada para permitir a seleção de um conjunto de 10 agentes que demonstraram ser mais competentes no jogo. Em seguida, tais agentes foram utilizados no processo de aprendizagem da base de casos, assim como descrito na seção 4.1.3. Por ser utilizada para selecionar os agentes que foram testados nas próximas etapas de testes realizadas, esta etapa é descrita como eliminatória 1.

Etapa 2 - Análise do processo de aprendizagem automático de casos: nessa etapa de testes, por meio da utilização de diferentes bases de casos, agentes baseados em políticas de reuso

de casos idênticas disputaram um conjunto de partidas. Nestas disputas, enquanto um dos agentes utilizou a base de casos inicialmente coletada com jogadores humanos, a qual continha 3195 casos, o outro agente envolvido na disputa da partida de jogo utilizou a base de casos resultante do processo de aprendizagem desenvolvido, a qual continha 27515 casos. A ideia destes testes foi analisar os efeitos do processo de aprendizagem automático de casos executado nesta dissertação.

Etapa 3 - Eliminatória 2: esta etapa de testes aplica os agentes classificados na etapa anterior em uma avaliação onde todos os classificados são aplicados em partidas duplicadas. Esta etapa se faz necessária para avaliar o desempenho dos agentes quando submetidos em partidas onde apenas os melhores agentes selecionados pela etapa anterior são considerados. Dessa forma, esta etapa permite a realização de um novo filtro de melhores agentes. Essa etapa de testes é feita após a aprendizagem automática. Por isso, nesta etapa de teste as políticas de reuso convencionais estarão no melhor cenário possível, visto que, com uma maior quantidade de casos é possível utilizar um *treshold* maior e diminuir a quantidade de casos recuperados em diferentes estados de decisão.

Etapa 4 - Avaliação contra jogadores humanos: nessa etapa de testes, os 4 melhores agentes resultantes das etapas de testes anteriormente desenvolvidas foram submetidos a disputa de partidas de Truco contra jogadores humanos. Nesta etapa de validação, além do desempenho quantitativo dos agentes, características qualitativas também foram avaliadas a partir de um questionário preenchido por jogadores humanos participantes dos testes.

5.2. EXECUÇÃO DOS TESTES

Para ser possível a realização dos testes e treinamentos, duas *engines* de Truco foram desenvolvidas: *web* e desktop. O desenvolvimento da *engine* desktop permitiu o desenvolvimento de competições envolvendo agentes virtuais, além de permitir desenvolver o processo de aprendizagem automática da base de casos. Para ser possível avaliar o desempenho dos agentes em partidas disputadas contra jogadores humanos, foi necessário adaptar o sistema *Web* para permitir a seleção automática dos agentes a serem usados em cada partida. Para isso, uma API web desenvolvida permitiu a instanciação automática dos agentes implementados. Apesar de serem implementações diferentes (*web* e desktop), ambas utilizam o mesmo padrão de consultas CBR.

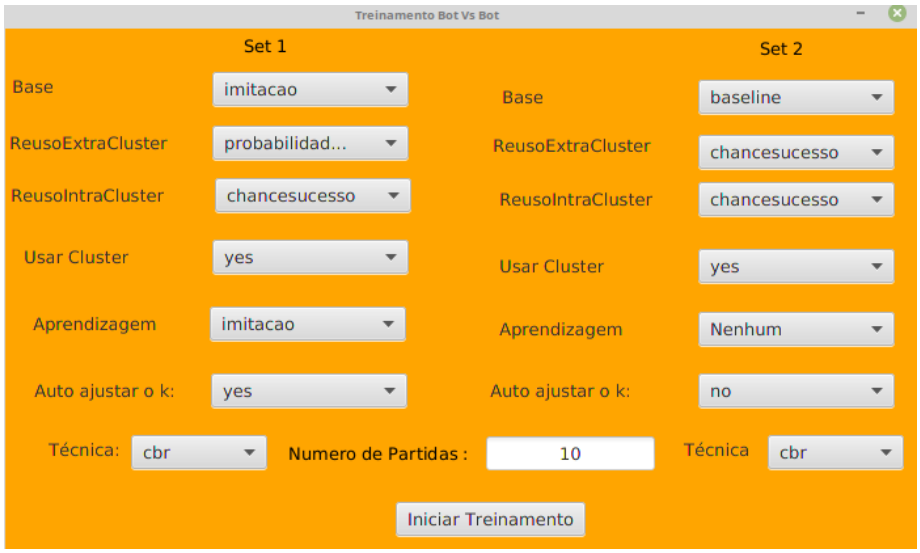
Figura 8 apresenta uma tela usada no processo de testes. Essa tela contém os parâmetros utilizados em partidas de Truco disputadas de forma automática, sejam elas com objetivo de validação dos agentes implementados ou retenção de novos casos na base de casos. Em geral, essa tela foi utilizada para configurar testes / treinamentos, onde os parâmetros que devem ser preenchidos são:

- Base de casos: permite escolher a base de casos a ser utilizada.
- Reuso extra cluster: permite escolher o critério de reuso a ser utilizado na escolha do cluster de casos, assim como implementado na computação do primeiro passo de resposta de uma consulta dada.
- Reuso intra cluster: permite escolher o critério de reuso a ser utilizado na escolha de uma solução para a consulta, assim como implementado na computação do segundo passo de resposta de uma consulta dada.
- Usar clusters: caso seja selecionado “no”, o reuso extra cluster é desconsiderado.

- Técnica de aprendizagem: critério de aprendizado a ser utilizado. Nesta dissertação, apenas a técnica de imitação foi implementada.
- Auto ajustar o número de clusters K: esse parâmetro é utilizado para permitir recalcular os grupos existentes nas diferentes execuções do algoritmo de *clustering*. Essa configuração é especialmente útil quando o processo de aprendizagem automática é executado.
- Técnica de IA: esse parâmetro permite que implementações de agentes com diferentes técnicas possam utilizar a mesma *engine*. Isto foi implementado para que outras pesquisas no futuro possam utilizar essa mesma implementação. Dessa forma, nossos agentes poderão ser validados em disputas de Truco realizadas contra agentes implementados de acordo com outras técnicas de IA.

Vale destacar alguns detalhes presentes na tela apresentada na Figura 8, por exemplo, os identificadores "set 1" e "set 2" são utilizados para permitir validações no modelo ACPC, isto é, é adicionado um número de partidas e configurado um agente no "set 1" (configuração de cartas 1) e outro agente no "set 2" (configuração de cartas 2). Depois das partidas terem sido concluídas, são invertidos os agentes (as configurações que estavam no set 1 vão para o set 2 e as que estavam no set 2 vão para o set 1), e o mesmo número de partidas é repetido. Na Figura 8, o número de partidas configurado é exemplificado pelo número 10. Por fim, o botão identificado como "iniciar treinamento" inicia as partidas com os parâmetros configurados.

Figura 8 - Interface usada na configuração de partidas *bot* vs *bot* de Truco



Fonte: do autor.

Figura 9 apresenta a tela *web* construída para permitir a seleção dos agentes oponentes nas partidas de Truco disputadas. Ao iniciar uma partida, a escolha de um oponente é necessária. Quando um jogador adversário é selecionado, uma instância CBR é iniciada a partir das configurações do agente selecionado. Para isso, foi desenvolvido uma API utilizada para comunicar o sistema web com o *engine* que

executa a aplicação CBR. Dessa forma, a mesma implementação CBR é utilizada em ambas as *engines* (web e desktop). Assim, a API de Truco desenvolvida realiza a comunicação entre o agente CBR instanciado e a tela de jogo em uma abordagem de implementação distribuída. Dessa maneira, o mesmo sistema *web* pode ser utilizado tanto para partidas onde: a) ambos os jogadores são seres humanos, b) um dos jogadores é um ser humano e o outro é um jogador virtual, c) ambos os jogadores são inteligências artificiais. Através da API que desenvolvemos, outras implementações de agentes virtuais podem ser facilmente integradas ao sistema *web* criado.

Figura 9 - Tela de escolha do oponente

Fonte: Grupo de pesquisa.

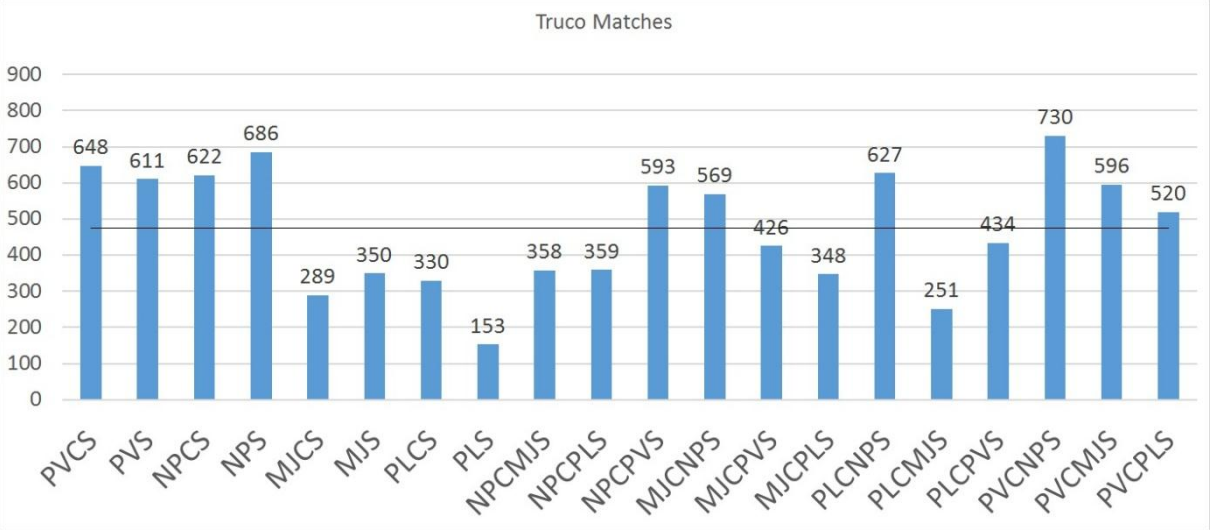
5.2.1. EXPERIMENTOS UTILIZANDO OS AGENTES IMPLEMENTADOS E A BASE DE CASOS INICIALMENTE COLETADA

Esse conjunto de testes foi desenvolvido com o uso da base de casos inicial de 3195 casos, os quais foram coletados a partir de partidas disputadas entre jogadores humanos. Todos os agentes implementados foram submetidos a 25 partidas duplicadas, as quais exploraram todas as configurações de oponentes possíveis. Cada um desses agentes jogou 950 partidas, o que totalizou 19000 partidas de Truco.

As partidas disputadas entre todos os agentes permitiram analisar o desempenho geral de cada agente implementado. Figura 10 apresenta o desempenho acumulado dos 20 agentes. Considerando que cada agente jogou 950 partidas, para ter um desempenho de 50%, o agente deve ganhar no mínimo 475 partidas.

Na etapa de testes seguinte a esta primeira etapa, a qual resultou na eliminação dos agentes com piores desempenhos, apenas os agentes que ganharam mais do que a metade das partidas disputadas foram considerados. Além de ter bons resultados em partidas disputadas, o objetivo é selecionar agentes com diferentes perfis de comportamento de jogo, os quais são usados na construção/expansão da base de casos de acordo com o processo de treinamento descrito na seção 4.1.3 desta dissertação.

Figura 10 - Resultados de partidas de Truco disputadas entre todos contra todos os agentes implementados.



Fonte: do autor.

Ao todo, a linha horizontal plotada no gráfico apresentado na Figura 10 indica que apenas 10 agentes ganharam mais da metade das partidas disputadas. Dentre eles, 8 agentes utilizaram o modelo de reuso em dois passos proposto nesta dissertação, enquanto que outros 2 agentes utilizaram o modelo de reuso convencional. O ranking de classificação destes competidores é o seguinte:

PVCNPS: O agente com melhor desempenho é resultante da união de dois diferentes critérios de reuso executados de acordo com o modelo de reuso em dois passos proposto nesta dissertação. O agente PVCNPS utilizou o critério PV (*probability victory*) na escolha do cluster (Ci) e o critério NP (*number of points*) na escolha das ações de jogo (S) presentes nos casos recuperados que pertencem ao cluster Ci selecionado. Dessa forma, é selecionada a ação de jogo que permitiu maior ganho nas mãos de jogos gravadas nos casos que pertencem ao grupo que possuir maior probabilidade de vitória.

NPS: O segundo agente com maior número de vitórias explorou o critério de reuso NP (*number of points*) utilizado de acordo com o modelo de reuso convencional descrito em (Rubin e Watson, 2010). A partir da observação deste agente em partidas disputadas, é possível afirmar que o agente NPS implementa um comportamento de jogo bastante agressivo.

PVCS: Este é o terceiro agente com melhor desempenho de acordo com as partidas disputadas. Este agente PVCS prioriza a escolha de clusters e ações de jogo que proporcionam maior probabilidade de vitória. Da maneira como foi implementado, foi possível observar que o comportamento deste agente tendeu a ser mais conservador que o comportamento de jogo executado pelos agentes PVCNPS e NPS.

PLCNPS: O quarto agente que mais ganhou partidas utilizou o critério PL (*probability lottery*) na escolha do cluster e o critério NP (*number of points*) na escolha das ações de jogo. É possível afirmar que esse agente combina imprevisibilidade com a maximização de resultados.

NPCS: A quinta posição foi conquistada pelo agente NPCS. Este agente utiliza o critério NP (*number of points*) tanto na escolha do cluster quanto na escolha da ação de jogo. Este agente tem um comportamento agressivo que busca maximizar os ganhos em ambas as etapas de seleção empregadas no modelo de reuso em dois passos executado.

PVS: Na sexta colocação está o agente PVS que aplicou o critério PV (*probability victory*) diretamente na escolha da solução. É possível afirmar que esse agente tende a ser conservador e considerar ações de jogo empregadas em estados de jogo possivelmente diferentes.

PVCMJS: O sétimo agente com melhor desempenho seleciona o cluster com maior probabilidade de vitória em uma mão de jogo. Com base nos casos recuperados que pertencem ao cluster selecionado, a ação de jogo que a maioria dos casos recuperados executaram é reusada. Conforme observado, esse agente tende a apresentar um comportamento conservador.

NPCPVS: O oitavo agente com melhor desempenho prioriza a escolha dos clusters que possibilitam maximizar os ganhos, e a escolha das ações de jogo com maior probabilidade de vitória (tanto nas ações de jogar cartas quanto nas ações de apostas). Em muitos sentidos, é possível afirmar que este agente não é muito competitivo visto

que os grupos de casos formados que representam os maiores ganhos tendem a conter ações de jogo arriscadas com altas chances de fracasso.

MJCNPS: O nono colocado prioriza pela escolha do cluster em que a maioria dos casos recuperados estão contidos. A escolha da solução seleciona as ações de jogo que proporcionaram maiores ganhos de acordo com os casos recuperados que pertencem ao cluster escolhido. A partir da observação do comportamento de jogo deste agente, o agente MJCNPS pode ser considerado previsível em certas situações e dependente da qualidade dos casos contidos na base de casos, isto é, o critério MJC tende a selecionar grupos que representam o comportamento de jogo mais frequentemente encontrado nos casos recuperados.

PVCPLS: Na décima posição está o agente que prioriza a escolha de casos que pertencem ao cluster que contém casos com as maiores chances de vitória. A escolha da solução é realizada por meio de um sorteio onde as probabilidades de cada ação de jogo ser utilizada são previamente definidas. Conforme observação, esse agente tende a apresentar comportamentos de jogo imprevisíveis e conservadores.

5.2.2. EXPERIMENTOS DESENVOLVIDOS CONSIDERANDO UM CONJUNTO DE AGENTES SELECIONADOS E A BASE DE CASOS PÓS-APRENDIZADO AUTOMÁTICO VS BASE DE CASOS INICIALMENTE COLETADA

O objetivo desta etapa de testes foi analisar o impacto do treinamento automático na performance das diferentes políticas de reuso implementadas. Para isso, os melhores agentes identificados na etapa de experimentação anterior foram submetidos a partidas duplicadas, onde tais agentes usaram as diferentes bases de casos existentes: base de casos inicialmente coletada (aqui denominada *baseline*) e base de casos resultante do processo de aprendizagem por imitação. Na totalidade, 500 partidas foram disputadas nesta etapa de testes, dentre as quais 287 foram vencidas pelos agentes que utilizaram a base de casos resultante do processo de treinamento desenvolvido.

Dentre as políticas de reuso avaliadas, apenas 3 delas não apresentaram uma melhoria de performance obtida a partir da expansão do número de casos da base de casos. Em particular, 3 políticas de reuso implementadas resultaram em uma redução no

número de vitórias alcançadas pelos seus respectivos agentes: MJCNPS, PLCNPS e NPCS. Como é possível verificar, essas políticas de reuso selecionam ações de jogo que proporcionam maiores ganhos de acordo com os casos recuperados e organizados pelo grupo de casos selecionado. Em geral, o comportamento de jogo resultante da aplicação do critério de maioria não garante que boas ações de jogo sejam selecionadas. Mais ainda, os critérios MJC e PLC utilizam a frequência que os grupos aparecem nos casos recuperados para permitir escolher o grupo que deve ser utilizado na computação das políticas de reuso. Sendo assim, a seleção da jogada que proporcionou maiores ganhos, considerando apenas os grupos que a maioria dos casos similares recuperados pertencem, tende a resultar em jogadas que não são garantidamente efetivas, principalmente quando a base de casos foi construída por jogadores com diferentes níveis de experiência. Já a aplicação do critério que busca selecionar a ação de jogo que possibilitou maiores ganhos dentro do grupo com maiores ganhos tende a resultar em agentes muito mais agressivos do que todos os demais agentes. Quando a base de casos treinada é utilizada, isto acontece porque é natural que novas jogadas sejam criadas, onde esta expansão também inclui casos contendo comportamentos de jogo agressivos.

Em geral, 7 políticas de reuso implementadas pelos agentes testados demonstraram um incremento de performance com o aumento da base de casos utilizada. Estas políticas de reuso são as seguintes:

NPS: por prezar pela seleção da ação de jogo que resultou nos maiores ganhos, essa política de reuso não considera a utilização de agrupamentos de casos. Em geral, a utilização de uma base de casos expandida favoreceu o reuso de ações de jogo computadas a partir da análise de um maior número de casos recuperados que demonstravam ocorrências de fracasso (derrotas no jogo). Em contraste com bases de casos menores, o fato da base de casos armazenar um maior número de casos de derrota fez com que os casos recuperados para uma consulta dada levassem a política NPS a sugerir ações de jogo típicas de um comportamento de jogo menos ofensivo para o agente. Além disso, a utilização de um *threshold* de similaridade mais alto (possível de ser implementado com um maior número de casos na base de casos) tende a reduzir o número de casos recuperados contendo ações de jogo realizadas em diferentes cenários de jogo, as quais são computadas pela política NPS como jogadas idênticas (isto é, incorretamente computadas como idênticas) no processo de reuso de soluções. Como consequência dessas características (comportamento de jogo menos ofensivo e ações de

jogo consideradas em contextos de jogo mais similares entre si), o agente NPS acabou aumentando o seu número de vitórias nas partidas disputadas.

PVCNPS: o uso dessa política de reuso primeiro seleciona o grupo de ações de jogo que proporcionam maiores chances de vitória. Dessa maneira, a escolha da ação que proporcionou maiores ganhos é realizada de acordo com um grupo de casos que fornecem certa segurança para as decisões. Por isso, o aumento do número de casos na base de casos possibilitou a evolução do desempenho deste agente PVCNPS.

PVCMJS: essa política de reuso também se beneficiou com o aumento do número de casos na base de casos, principalmente pelo fato de utilizar o critério PV para a escolha do grupo de casos a ser analisado no primeiro passo de reuso. Combinado ao critério MJ para a escolha da ação de jogo a ser executada no segundo passo de reuso, o agente PVCMJS permitiu a seleção da ação de jogo mais comumente efetuada de acordo com os casos que pertencem ao grupo com maior probabilidade de vitória.

PVCPLS: o aumento do número de casos na base de casos também resultou no aumento de performance desta política. Neste caso, a política PVCPLS indica que ações de jogo efetuadas dentre os casos que pertencem ao grupo com maior probabilidade de vitória devem ser utilizadas em uma função de sorteio, e esta função considera as chances de cada ação de jogo ser executada. Desta forma, é possível garantir que as ações consideradas no sorteio pertencem ao grupo de casos que possibilita maior probabilidade de vitória.

PVS: esta política de reuso alcançou uma melhora significativa de performance quando a base de casos expandida foi usada pelo agente PVS. Isso se deve ao fato de que o aumento do número de amostras permite alcançar uma maior acurácia em decisões baseadas em probabilidade.

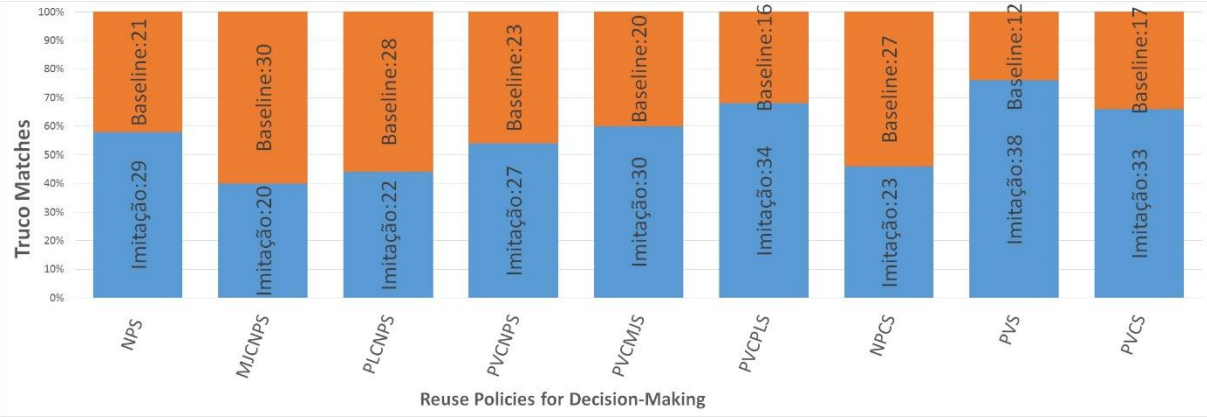
PVCS: o aumento do número de casos na base de casos também proporcionou benefícios quando o critério PV foi explorado por um agente de acordo com modelo de reuso de dois passos. Acreditamos que isso se deva ao aumento de acurácia que um maior número de amostras proporciona a critérios baseados em probabilidade.

NPCPVS: a escolha da ação de jogo com maior chance de vitória dentro do grupo que proporcionou maiores ganhos também se beneficiou com o aumento do número de casos na base de casos. Isso se deve ao fato de que, mesmo que existam ações de jogo consideradas arriscadas dentro do grupo que proporcionou maiores ganhos, o

critério PV apenas sugere ações em que a probabilidade de vitória é maior do que a de derrota.

Figura 11 apresenta os desempenhos obtidos por cada uma das políticas de reuso implementadas pelos 10 agentes de acordo com as diferentes configurações de base de casos utilizadas. Em resumo, nesta Figura é possível perceber que políticas que aplicaram o critério de reuso PV para a escolha do cluster a ser utilizado para a seleção da solução, evoluíram com a inserção de novos casos. Enquanto isso, quando o critério NP foi aplicado sem combinar com o critério PV, a adição dos novos casos não foi benéfica. Isso por que nos casos aprendidos, grupos/ações que proporcionaram uma grande quantidade de vitórias podem ser compostos por jogadas que tem um alto risco de fracasso, por isso obtiveram piores resultados quando utilizadas em conjunto com a base treinada.

Figura 11 – Performance dos agentes implementados a partir do uso da Base de casos resultante do processo de aprendizado (Imitação) vs Base de casos inicialmente coletada (Baseline)



Fonte: do autor.

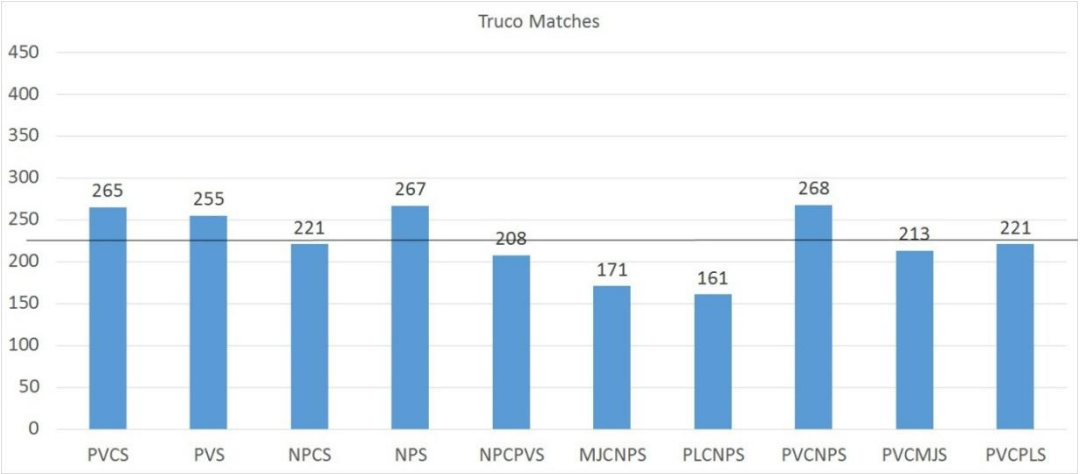
A melhora no desempenho obtida pela maioria das políticas de reuso usadas pelos agentes testados comprova a efetividade do uso de uma base de casos expandida, a qual foi obtida de acordo com o processo de treinamento/aquisição automática de casos desenvolvido nesta dissertação. Da mesma forma, a redução no desempenho de 3 agentes testados sugere que novas jogadas agressivas de sucesso foram criadas. A criação desses novos comportamentos é benéfica para a pesquisa, visto que, o objetivo principal do trabalho é identificar políticas de reuso capazes de proporcionar as melhores soluções em bases de casos construídas por jogadores com diferentes estilos de jogos.

5.2.3. TODOS CONTRA TODOS PÓS ELIMINATÓRIA 1

O objetivo desta etapa de testes é utilizar a estratégia de partidas duplicadas para aprofundar a análise do desempenho de um conjunto de agentes selecionados. Em particular, somente os 10 melhores agentes da etapa de testes anteriormente desenvolvida foram selecionados para novos experimentos nesta dissertação. Nesta etapa, todos os 10 agentes utilizados empregaram a base de casos expandida contendo 27515 casos na computação de decisões de jogo. Tais agentes selecionados foram utilizados em partidas de Truco onde todas as configurações de oponentes possíveis foram testadas (disputas todos contra todos). Em geral, cada agente disputou 450 partidas, totalizando 2250 partidas utilizadas nesta etapa de testes.

Figura 12 apresenta o desempenho acumulado dos 10 agentes utilizados neste processo de disputa. Este desempenho é descrito em termos de número de vitórias. Na próxima etapa de testes, apenas os agentes que ganharam mais da metade das partidas aqui disputadas são considerados (os demais são descartados). O objetivo desta seleção foi utilizar apenas os agentes com melhor desempenho em testes subsequentes, os quais foram realizados contra jogadores humanos (devido a dificuldade de encontrar jogadores humanos dispostos a jogar um número elevado de partidas, um menor número de agentes teve que ser utilizado neste processo final de validação).

Figura 12 - Resultados obtidos em partidas duplicadas disputadas entre os 10 melhores agentes implementados utilizando a base de casos resultante do processo de aprendizagem (base de casos denominada Imitação)



Fonte: do autor.

Ao todo, 4 agentes testados venceram mais da metade das partidas disputadas. Dentre eles, 2 utilizaram o modelo de reuso em dois passos proposto nesta dissertação e 2 utilizaram o modelo de reuso convencional proposto na literatura (Rubin e Watson,

2010). Em particular, Tabela 16 apresenta as diferenças médias de pontuação no jogo de Truco que cada um desses 4 agentes (abordagens de reuso) obtiveram nas partidas disputadas. Neste caso, uma diferença de placar positiva é usada para descrever casos de vitória e negativa para casos de derrota. O objetivo desta análise é determinar se as políticas de reuso implementadas por esses agentes permitiram que eles aumentassem os seus ganhos e diminuíssem as suas perdas nas partidas de Truco disputadas.

Tabela 16 – Diferença média de pontuação que cada um dos agentes obteve nas partidas disputadas.

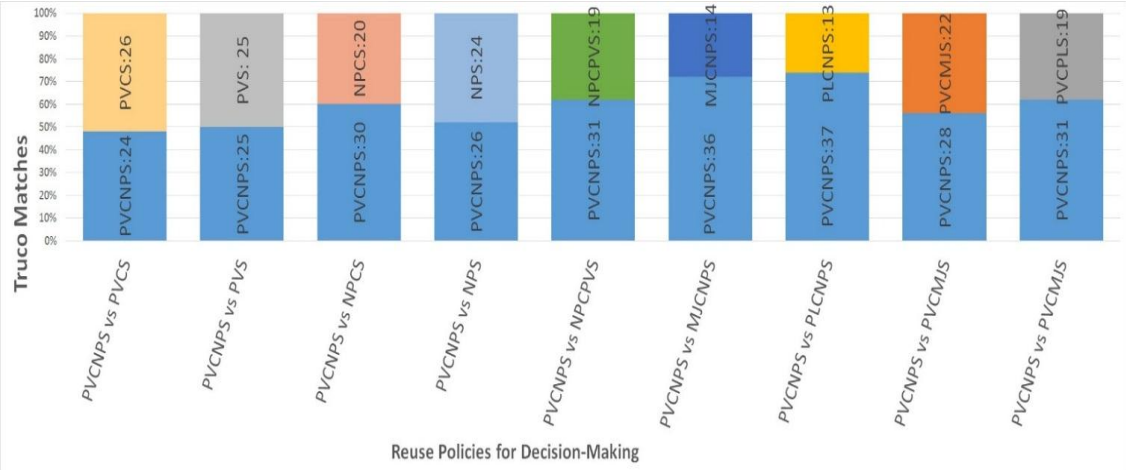
Agente	Diferença média de pontos em vitórias	Diferença média de pontos em derrotas
PVCNPS	7,01	-5,08
NPS	6,03	-5,80
PVCS	5,80	-6,90
PVS	5,70	-6,28

Fonte: do autor.

Em ordem decrescente de classificação, os resultados dos testes envolvendo os 4 agentes selecionados nesta etapa de testes são apresentados a seguir:

PVCNPS: Além de ter sido a política de reuso que proporcionou que o agente obtivesse a maior diferença de pontuação em vitórias e a menor diferença de pontuação em derrotas (ver Tabela 16), essa política proporcionou que este agente obtivesse o maior número de vitórias. Figura 13 apresenta os resultados obtidos pelo agente PVCNPS contra agentes implementados de acordo com as demais políticas de reuso utilizadas nestes testes. Em geral, a combinação entre um critério de tomada de decisão considerado conservador para escolha do grupo e um critério de tomada de decisão considerado agressivo para a escolha da ação de jogo permitiu que o agente PVCNPS vencesse a maioria dos oponentes. O efeito positivo da política de reuso PVCNPS foi ainda maior quando os agentes oponentes utilizaram um comportamento conservador de jogo, ou seja, oponentes que tendem a apostar apenas quando têm mãos de jogo muito boas ou quando os adversários fizeram uso de políticas agressivas com a utilização de grupos de casos onde existiam jogadas muito arriscadas.

Figura 13 - Desempenho do agente PVCNPS em partidas de Truco disputadas contra outros agentes selecionados.



Fonte: do autor.

NPS: o agente baseado na política NPS obteve o segundo melhor resultado neste terceiro conjunto de testes, onde figura 14 apresenta os resultados obtidos pelo agente NPS contra outros agentes implementados. Por aplicar o critério NP diretamente na escolha da solução, essa política permite que o agente apresente um comportamento mais agressivo que o comportamento obtido quando o critério PVCNPS é usado. Por isso, em algumas situações o agente implementado de acordo com a política NPS permite obter resultados melhores. Em outras situações de jogo, o agente NPS é penalizado pois a política NPS tende a sugerir a execução de ações de jogo arriscadas (as mesmas ações que proporcionam maiores ganhos são as que causam maiores perdas), as quais podem não ter sucesso.

PVCS: nestes testes, o agente implementado com a política PVCS obteve o terceiro maior número de vitórias. A abordagem PVCS, portanto, possibilitou a implementação de um agente capaz de priorizar ações de jogo com uma alta probabilidade de vitória. Apesar de implementar uma política que pode ser considerada conservadora, o agente PVCS obteve a terceira maior diferença de pontuações em vitórias.

PVS: dentre as 4 políticas que foram usadas na implementação dos agentes testados, os quais venceram mais da metade das partidas disputadas, PVS foi a política usada pelo

agente que obteve o menor número de vitórias. Porém, o agente PVS obteve a terceira menor diferença de pontuações em derrotas (ver Tabela 16).

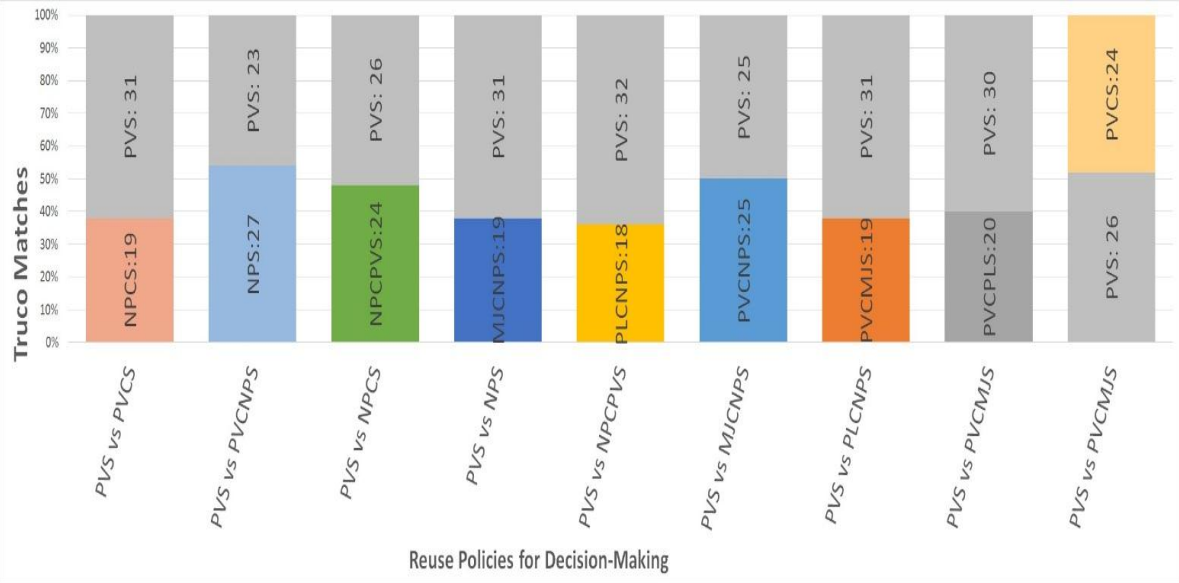
Figura 14 - Desempenho do agente NPS em partidas de Truco disputadas contra outros agentes selecionados.

Fonte: do autor.

Figura 15 - Desempenho do agente PVCS em partidas de Truco disputadas contra outros agentes selecionados.

Fonte: do autor.

Figura 16 - Desempenho do agente PVS em partidas de Truco disputadas contra outros agentes selecionados.



Fonte: do autor.

Em resumo, os resultados obtidos nesta etapa de testes permitiram a seleção dos melhores agentes desenvolvidos, apesar de utilizar somente os melhores agentes identificados na etapa de testes anterior. Como seres humanos utilizam capacidades extras de avaliação (avaliação qualitativa, por exemplo), os agentes que conseguiram ganhar no mínimo a metade das partidas disputadas nesta última etapa de testes foram selecionados para uma nova etapa de avaliação das técnicas propostas nesta dissertação.

5.2.4. TESTES CONTRA JOGADORES HUMANOS

De acordo com (Niklaus *et al.*, 2019), testar o desempenho de implementações de IA contra jogadores humanos é uma importante maneira de avaliar a efetividade de agentes capazes de jogar cartas. Para essa validação ser possível, os 4 melhores agentes identificados nos testes anteriormente desenvolvidos foram submetidos a 13 partidas de Truco disputadas contra jogadores humanos. Devido ao tempo necessário para jogar tais partidas, alguns participantes não conseguiram avaliar todos os 4 agentes. No total, a realização destes testes contou com a colaboração de 22 diferentes jogadores/participantes.

Dentre as políticas de reuso analisadas, a união entre os critérios PV e NP aplicados ao modelo de reuso em dois passos foi a abordagem que alcançou melhores resultados. Além de ganhar 7 das 13 partidas disputadas contra jogadores humanos, a

política PVCNPS apresentou a maior capacidade de maximizar os ganhos e minimizar as perdas. Tabela 17 apresenta o desempenho de cada um dos agentes avaliados.

Tabela 17 - Desempenho Agentes vs Humanos

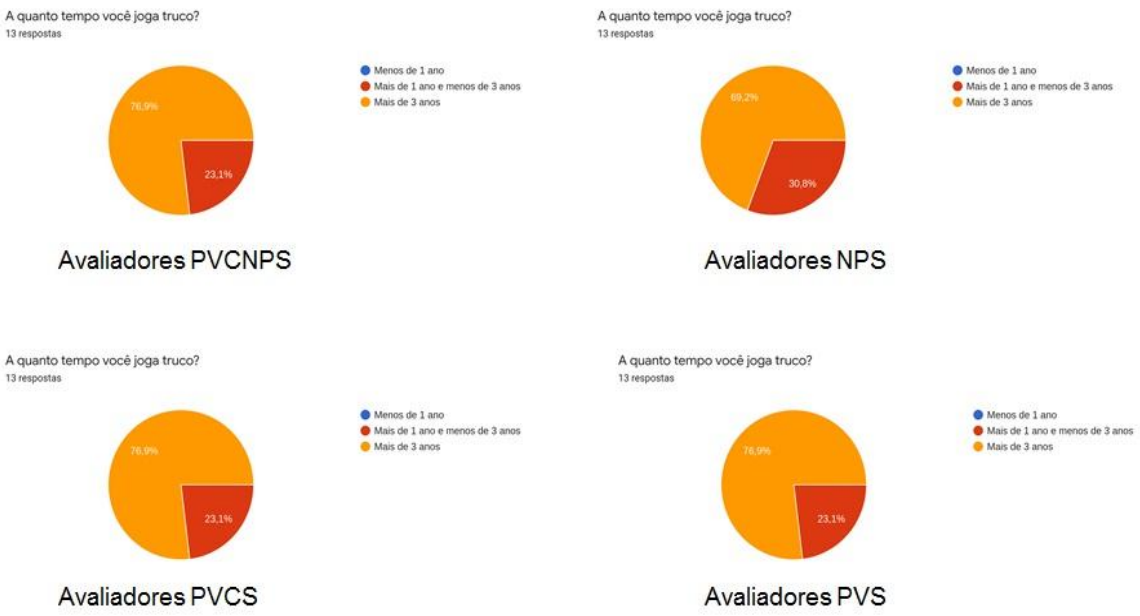
Agente	Quantidade de vitórias dos agentes (total de 13 partidas disputadas)	Diferença média de pontos em vitórias	Diferença média de pontos em derrotas
PVCNPS	7	6,75	-4,10
NPS	6	6,71	-8,11
PVCS	1	11	-8,26
PVS	5	5,28	-12

Fonte: do autor.

Para ser possível coletar informações qualitativas do desempenho de jogo de cada um desses agentes, os jogadores humanos foram direcionados a responder um formulário de avaliação ao final de cada partida disputada. As perguntas do questionário buscaram avaliar diferentes características dos 4 agentes selecionados: competência, coerência, previsibilidade e divertimento. Para facilitar a análise do questionário, as respostas dos jogadores humanos foram coletadas em escalas de 1 (avaliação muito negativa) e 5 (avaliação muito positiva).

Como diferentes participantes avaliaram as abordagens usadas na implementação dos agentes, a experiência aproximada de cada jogador no jogo de Truco também foi questionada (em termos de tempo que estes jogadores jogavam Truco). Figura 17 apresenta a experiência dos participantes deste processo de avaliação.

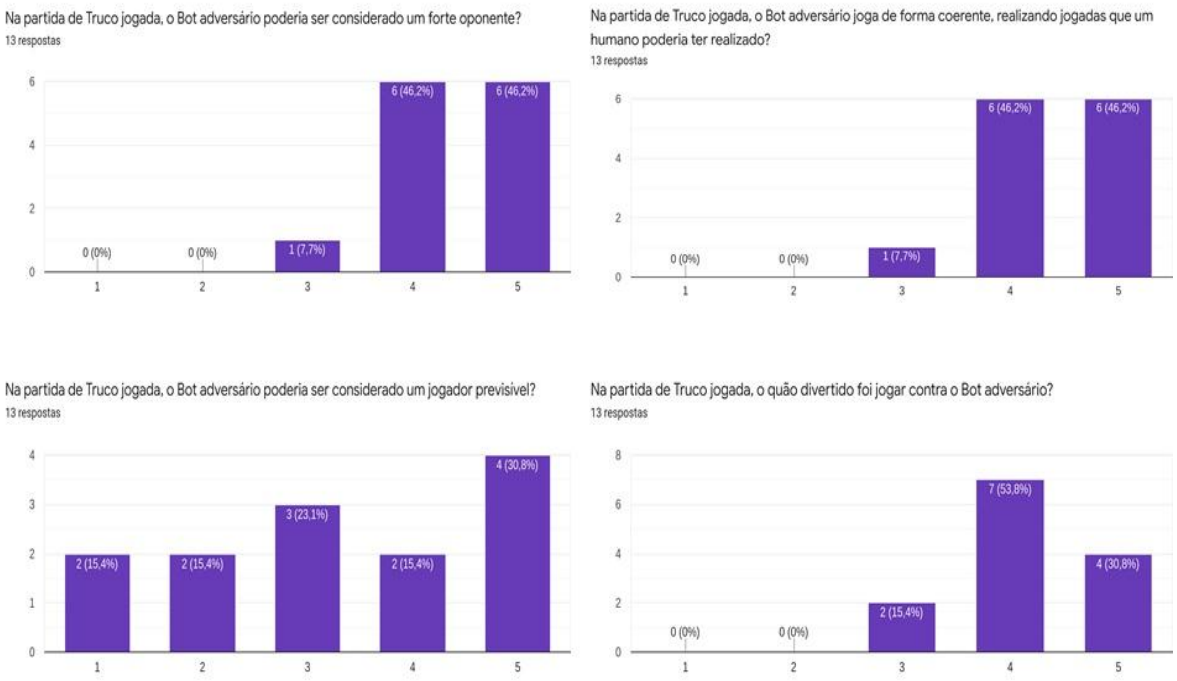
Figura 17 - Experiência dos avaliadores



Fonte: do autor.

Figura 18 apresenta as respostas dos participantes do experimento coletadas sobre o agente PVCNPS. Dentre as 13 respostas coletadas em relação a competência do agente PVCNPS, 12 participantes consideraram esse agente um oponente forte ou muito forte, enquanto apenas 1 participante considerou que esse agente é um oponente regular. Na análise da coerência deste agente, 12 participantes avaliaram o agente como muito coerente ou coerente, enquanto apenas 1 participante avaliou o agente como regular. A avaliação de previsibilidade do agente PVCNPS resultou em uma grande variabilidade de respostas. Acreditamos que essa variabilidade está relacionada com a escolha de grupos de casos contendo grande chance de vitória de acordo com o critério PV na escolha dos clusters e o comportamento agressivo do critério NP na escolha das ações de jogo. No quesito divertimento, 11 participantes consideraram este agente muito divertido ou divertido, enquanto apenas 2 pessoas avaliaram ele como regular. No geral, as avaliações indicaram que a abordagem de reuso PVCNPS foi capaz de apoiar a implementação de um agente competitivo e capaz de tomar decisões coerentes no jogo de Truco.

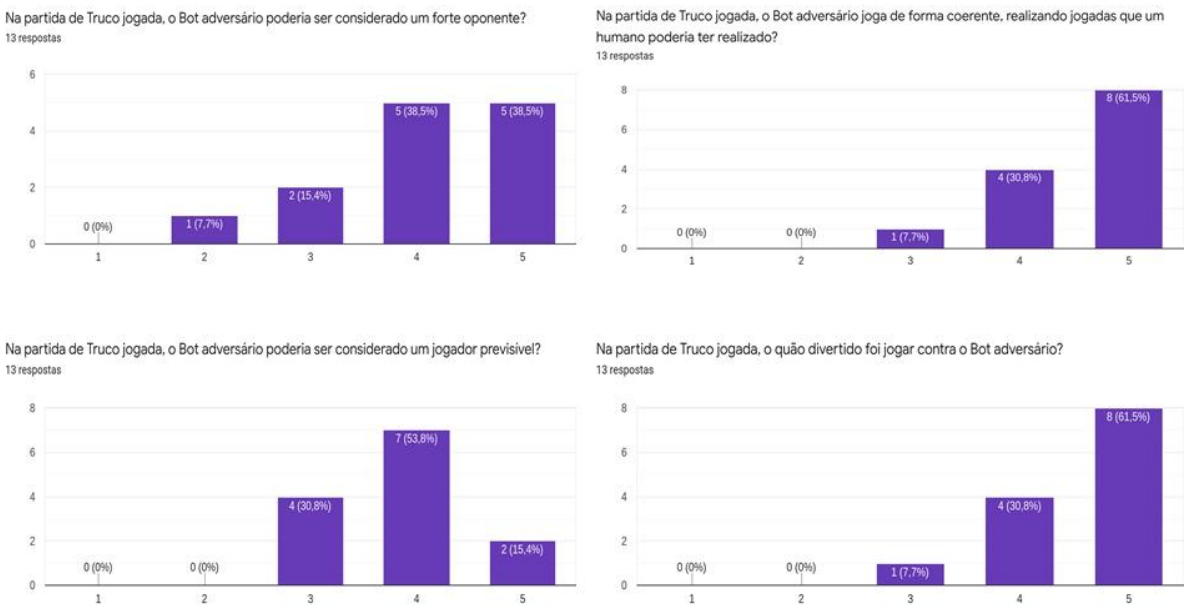
Figura 18 - Avaliações do Agente PVCNPS



Fonte: do autor.

Figura 19 apresenta as avaliações recebidas pelo agente NPS. Ao serem questionados sobre quão forte jogador de Truco foi esse agente, 9 participantes consideraram um jogador forte ou muito forte, 2 participantes consideraram o agente um jogador razoável e apenas 1 participante considerou ele como um jogador fraco. No quesito coerência, 12 participantes avaliaram esse agente NPS como muito coerente ou coerente e apenas 1 participante avaliou ele como regular. No quesito previsibilidade, 9 participantes avaliaram o agente como muito imprevisível ou imprevisível, enquanto 4 participantes avaliaram o agente como regular. Quando questionados sobre quão divertido foi jogar contra esse agente, 9 participantes avaliaram como muito divertido ou divertido, enquanto 1 participante avaliou como regular. Ao aplicar o critério NP diretamente na escolha da solução, ações de jogo agressivas foram executadas pelo agente NPS. Por isso, acreditamos que esse tenha sido o motivo desse agente ter sido considerado um oponente mais fraco do que o agente PVCNPS. Porém, também acreditamos que as boas avaliações obtidas no quesito divertimento estejam relacionadas a agressividade de jogo proporcionada pela técnica de reuso implementada neste agente.

Figura 19 - Avaliações do Agente NPS

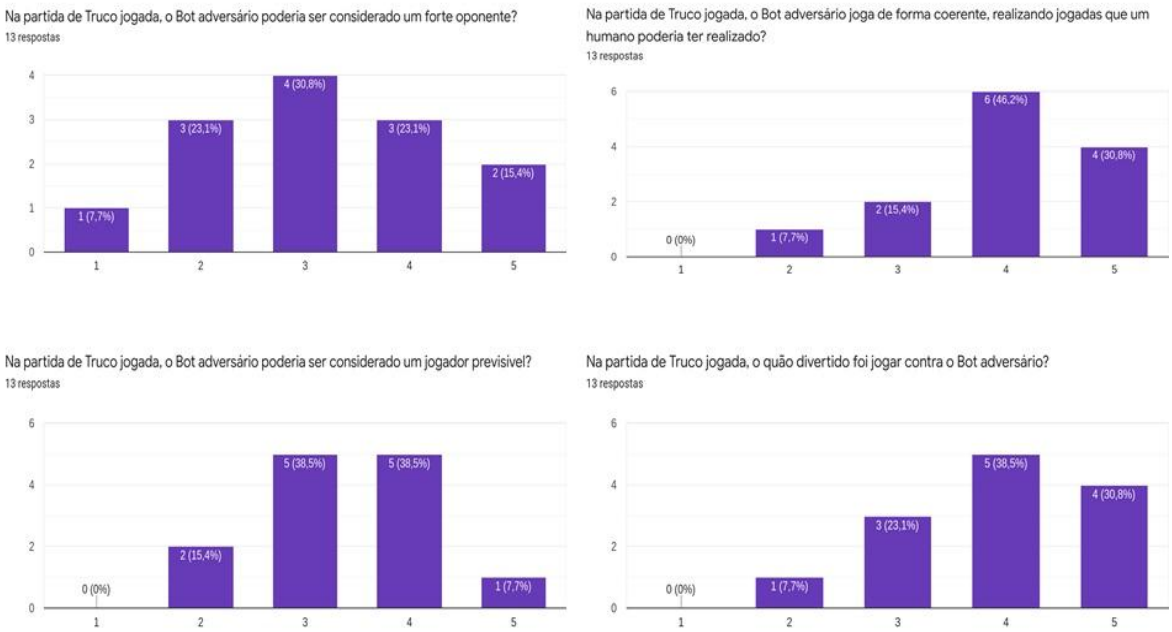


Fonte: do autor.

Os resultados das avaliações obtidas pelo agente PVCS são apresentados na Figura 20. Na avaliação de competência, 5 participantes avaliaram esse agente como oponente muito forte ou forte, 4 participantes consideraram o agente como regular, 3 participantes avaliaram ele como fraco ou muito fraco. Quanto a coerência, 10 participantes avaliaram o agente como um oponente muito coerente ou coerente, 2 participantes consideraram o agente como um jogador regular e 1 participantes avaliou ele como um jogador incoerente. No quesito previsibilidade, 6 participantes avaliaram o agente como muito imprevisível ou imprevisível, 5 participantes avaliaram ele como um jogador regular, 2 participantes avaliaram o agente como previsível. Ao serem questionados sobre quão divertido foi jogar contra esse oponente, 9 participantes consideraram que o jogo foi muito divertido ou divertido, 3 participantes consideraram o jogo regular e 1 participantes considerou o jogo chato. Acreditamos que as avaliações recebidas pelo agente PVCS estão relacionadas com o excesso de conservadorismo empregado pela técnica de reuso PVCS. A utilização do critério PV tanto para escolha do cluster quanto para escolha das ações de jogo permitiram implementar agentes competitivos de acordo com os resultados obtidos nas partidas disputadas contra outros agentes implementados. Porém, devido a maior capacidade interpretativa dos jogadores

humanos, foi possível perceber que eles foram capazes de explorar mais facilmente o excesso de conservadorismo resultante da utilização dessa política de reuso.

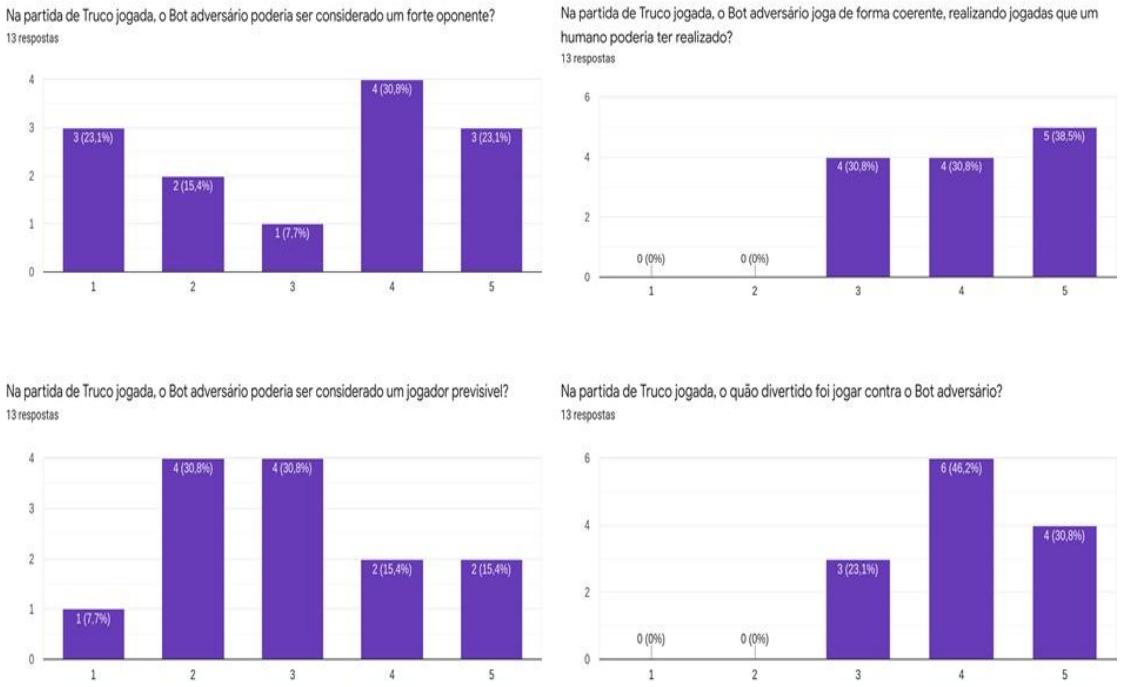
Figura 20 - Avaliações do Agente PVCS



Fonte: do autor.

Figura 21 apresenta as avaliações recebidas pela política PVS. Conforme as respostas coletadas, 7 participantes consideraram que o agente PVS é um oponente forte ou muito forte, 1 participante classificou esse agente como razoável, 5 participantes classificaram esse agente como fraco ou muito fraco. No quesito coerência, 9 participantes consideraram o agente coerente ou muito coerente, 4 participantes avaliaram o agente como razoavelmente coerente. Quanto a previsibilidade, 4 participantes avaliaram o agente como muito imprevisível ou imprevisível, 4 participantes avaliaram o agente como regular, 5 participantes classificaram o agente como muito previsível ou previsível. Em relação ao divertimento, 9 participantes consideraram muito divertido ou divertido jogar contra esse agente, 3 participantes avaliaram o agente como razoavelmente divertido. Acreditamos que a aplicação da probabilidade de vitória diretamente na escolha da solução também proporcionou um comportamento conservador para o agente PVS, onde esse comportamento conservador foi logo identificado pela maioria dos jogadores nas partidas disputadas.

Figura 21 - Avaliações do Agente PVS



Fonte: do autor.

5.3. DISCUSSÃO DOS EXPERIMENTOS E RESULTADOS

Com o objetivo de validar as propostas apresentadas nesta dissertação, diferentes tipos de experimentos foram executados. Em (Paulus *et al.*, 2019), o modelo de reuso proposto utilizou o mesmo critério de reuso na implementação da técnica de reuso em dois passos descrita neste trabalho. Naquela ocasião, os resultados foram promissores para uma situação onde o *threshold* de similaridade usado era baixo (75%), o que permitia explorar a pequena quantidade de casos disponíveis na base de casos. Devido a maior quantidade de agentes apresentados neste trabalho e a evolução da pesquisa desenvolvida, diferentes etapas de validação foram desenvolvidas.

Inicialmente, os 20 agentes implementados foram submetidos a partidas duplicadas seguindo o modelo da ACPC. Esta etapa de validação foi conduzida a partir da utilização de uma base de casos inicial contendo 3195 casos. Esta etapa de testes resultou em 44 vitórias a mais para o agente que implementou a política PVCNPS, em relação ao agente que implementou a política NPS, o qual foi o segundo colocado nesta competição. Dentre os 10 agentes classificados nesta etapa, 8 deles utilizaram o modelo

de reuso em 2 passos proposto nesta dissertação. Acreditamos que o melhor desempenho dos agentes que utilizaram o modelo de reuso proposto esteja relacionado ao fato de que o *threshold* de 75% não ser suficiente para distinguir ações de jogo suficientemente similares possivelmente empregadas em estados de jogo diferentes.

O segundo teste foi feito com o objetivo de validar o modelo de aprendizado proposto. Por isso, agentes implementados com as mesmas políticas de reuso se desafiaram utilizando as diferentes bases de casos (a base inicial com 3195 casos vs a base treinada com 27515 casos). Como nosso objetivo principal foi avaliar o desempenho que políticas de reuso obteriam em base de casos construídas por jogadores com tipos de comportamentos de jogo variados, a base de casos final utilizada foi construída a partir do emprego de diferentes agentes implementados. Como resultado, dentre as 10 políticas de reuso classificadas na etapa de testes anterior, 7 delas obtiveram evolução no desempenho quando a base de casos treinada foi utilizada. A combinação entre diferentes pares de jogadores durante a etapa de treinamento proporcionou diferentes resultados para diferentes estados de decisão. Porém, os novos casos criados não foram benéficos para políticas de reuso que aplicaram critérios diferentes do PV para a escolha do cluster, os quais foram combinados ao critério NP para a escolha da ação de jogo a ser tomada. Estes resultados indicam que a aplicação do critério NP em grupos de casos que não tem probabilidade de sucesso comprovada pode levar a escolha de ações de jogo que favorecem a perda de pontos em bases de casos onde existe maior quantidade de comportamentos de jogo agressivos (base treinada) comparada a utilização das mesmas políticas em bases de casos contendo ações de jogo mais conservadores (base inicial).

O terceiro teste permitiu avaliar o desempenho obtido pelos agentes quando um *threshold* de similaridade mais alto foi utilizado no processo de recuperação de casos da base de casos. Apesar da evolução obtida por políticas aplicadas ao modelo convencional de reuso de soluções (devido a menor incidência de instâncias diferentes de combinações estado – ação presentes nos casos recuperados), a combinação entre os critérios PVC e NPS permitiu uma maior diferença de pontos quando o agente estava na frente no placar e menor diferença quando o agente estava atrás no placar.

O quarto teste permitiu avaliar os agentes classificados na etapa de testes anterior em partidas disputadas contra jogadores humanos. Nesta etapa de validação, a combinação entre os critérios PVC e NPS alcançou uma diferença de pontos ainda maior

em vitórias do que alcançou na etapa de testes anterior. A diferença de pontos quando o agente estava atrás no placar também foi menor do que a apresentada na etapa de testes realizada anteriormente.

Em resumo, o primeiro conjunto de testes realizado reforçou os resultados apresentados em (Paulus *et al.*, 2019): a aplicação de critérios de reuso em resultados de agrupamentos de dados tende a reduzir a contabilização de ações de jogo como idênticas apesar de terem sido feitas em diferentes estados de jogo. Isso é particularmente observado quando um *threshold* relativamente baixo de recuperação foi usado, o qual era necessário para viabilizar o uso de bases de casos não tão grandes. Enquanto isso, no terceiro e quarto conjuntos de testes (onde as taxas de efetividade das políticas aplicadas no modelo convencional aumentaram devido ao uso de uma base de casos expandida e a possibilidade de utilizar um *threshold* de recuperação mais alto) a combinação entre os critérios PVC e NPS permitiram maximizar ganhos e minimizar perdas no jogo.

6. CONCLUSÕES

A natureza estocástica e a visão parcial presente em jogos de cartas, extensivamente observados no jogo de Truco, apresentam um cenário desafiador para a exploração de casos e clusters na construção de novas políticas de reuso para sistemas CBR. Neste contexto, a contribuição dessa dissertação é descrever como construir essas políticas de reuso a partir do emprego de resultados obtidos pela execução de algoritmos de *clustering*. Tais algoritmos permitem analisar e utilizar diferentes estados/cenários de decisão de jogo na computação de ações de jogo retornadas como resultados de consultas CBR. Em muitos sentidos, a computação de melhores respostas para essas consultas é fundamental em diferentes aplicações de CBR, assim como é fundamental no apoio ao processo de tomada de decisão de agentes capazes de jogar cartas em diferentes níveis de competência.

Em linhas gerais, o trabalho apresentado em (Paulus *et al.*, 2019) já apresenta uma contribuição para a área de pesquisa explorada nesta dissertação. Contudo, esse trabalho é expandido em diferentes sentidos, os quais são descritos a seguir.

(A) Emprego de um modelo otimizado de recuperação de casos: para diminuir o custo computacional e reduzir o tempo de tomada de decisão em sistemas CBR, exploramos resultados obtidos a partir da execução de algoritmos de *clustering* nos casos armazenados na base de casos na computação de um modelo de recuperação em dois passos tal como apresentado em (Chen *et al.*, 2018). Essa técnica é relevante para a construção de sistemas CBR que utilizam grandes bases de casos na computação de respostas para consultas dadas.

(B) Utilização de diferentes critérios de reuso em uma mesma política de reuso: diferente do que foi apresentado em (Paulus *et al.*, 2019), onde um único critério de reuso era aplicado nos dois passos de reuso computados, diferentes critérios de reuso são combinados no modelo de reuso proposto. Em particular, um determinado critério de reuso pode ser aplicado para escolher um grupo de comportamentos de jogo, enquanto outro utiliza os casos pertencentes ao grupo escolhido para a determinação da solução de jogo que deve ser executada por um agente. A combinação dos diferentes critérios aplicados no nosso modelo de reuso em dois passos possibilitou um aumento considerável no número de políticas de reuso propostas e testadas. Este modelo de reuso em dois passos utilizado permitiu que soluções empregadas em diferentes estados de decisões não fossem computadas como jogadas idênticas pelos mecanismos de reuso usados em sistemas CBR. Além disso, o modelo permitiu combinar critérios de reuso alternativos (PVCS, NPCS, MJCS, PLCS, NPCMJS, NPCPLS, NPCPVS, MJCNPS, MJCPVS, MJCPLS, PLCNPS, PLCMJS, PLCPVS, PVCNPS, PVCMJS e PVCPLS) na construção de uma única política de reuso explorada por agentes implementados. Além disso, é relevante observar que a dissertação apresenta critérios de reuso ajustados para jogos de cartas. Além de utilizarmos critérios de reuso de soluções descritos na literatura (Sandven e Tessem, 2006; Rubin e Watson, 2010), essa dissertação descreve como adaptar esses critérios as necessidades de agentes amplamente explorados na disputa de partidas de Truco.

(C) Revisão de jogadas consideradas suicidas (jogadas muitas vezes aleatórias e não consistentes com as regras de jogo do Truco): em bases de casos onde existem experiências construídas por jogadores de diversos níveis de habilidade, pode ocorrer a existência de decisões de jogo consideradas muito pouco efetivas por diversos motivos. Para criar agentes competitivos e validar o modelo proposto, utilizamos regras para

revisar algumas ações de jogo reusadas onde as decisões de jogo sugeridas são muito extremas (de acordo com nossa experiência, ações de jogo reusadas que têm uma grande chance de dar errado ou funcionar apenas em situações muito específicas).

(D) Emprego de uma técnica de aprendizado automático: em (Paulus *et al.*, 2019), a abordagem proposta foi analisada em um cenário de pesquisa onde existia grande dificuldade de coleta de novas experiências de solução de problemas. Para atacar este problema, esta dissertação explora um modelo de aprendizado automático baseado em imitação (Gómez-Martín *et al.*, 2005; Sandven e Tessem, 2006; Floyd *et al.*, 2008), o que permitiu aumentar consideravelmente o número de experiências de jogo armazenadas na base de casos. Para isso, foi necessária a adição de critérios capazes de permitir, em tempo de execução, a identificação do número de grupos existentes para cada cenário de jogo onde uma decisão de jogo era necessária. Além disso, a dissertação explora uma técnica baseada no uso de um *threshold* de similaridade variável. Neste caso, exploramos o uso de um *threshold* que inicia com um valor alto e diminuí gradativamente quando uma quantidade mínima de casos com esse valor mínimo de similaridade não é retornada como resultado de uma consulta dada. Essa variação do *threshold* de recuperação explorada foi particularmente importante no desenvolvimento do processo de treinamento que resultou no incremento gradual do número de casos armazenado na base de casos.

(E) Utilização de diferentes etapas de testes: além de testar um número maior de políticas de reuso, os agentes implementados são avaliados em diferentes etapas nesta dissertação. Estes testes foram feitos com o objetivo de analisar o desempenho das políticas propostas em diferentes situações. Tais testes foram divididos em:

- Testes com base de casos inicialmente coletada com jogadores humanos: semelhante aos testes apresentados em (Paulus *et al.*, 2019), todos os agentes implementados são submetidos a disputa de partidas duplicadas entre eles nestes testes.
- Validação de aprendizagem: as melhores políticas de reuso inicialmente selecionadas neste trabalho foram utilizadas para criar novas experiências de solução de problemas no domínio de aplicação sendo atacado. Para analisar o impacto que os novos casos poderiam ter na abordagem proposta, políticas de reuso idênticas foram aplicadas em disputas de partidas duplicadas com a utilização das diferentes bases de casos existentes.
- Disputa entre políticas de reuso com a base de casos expandida: para analisar nossas políticas de reuso em situações onde existe garantia de que os casos recuperados têm uma alta similaridade com a consulta dada, as políticas de reuso que obtiveram os melhores desempenhos nos testes anteriormente realizados foram submetidas a disputas de jogo via partidas duplicadas com o uso de uma base de casos armazenando uma quantidade razoável de casos (uma base de casos mais competente para resolver diferentes problemas no domínio de aplicação que a base é usada). Esta etapa de teste possibilitou avaliar as políticas de reuso em dois passos contra as políticas de reuso convencionais, em circunstâncias adversas. Isto é, a base de casos com uma maior quantidade de experiências, permitiu aumentar a similaridade mínima dos casos recuperados. A consideração de casos com similaridades maiores tendem a beneficiar políticas de reuso convencionais. Mesmo nestas circunstâncias a combinação entre os critérios PV e NP aplicados no nosso modelo de reuso em dois passos obteve melhores resultados.
- Validação com jogadores humanos: partidas disputadas contra jogadores humanos permitiram obter avaliações quantitativas e qualitativas dos agentes implementados nesta dissertação.

Na prática, o desenvolvimento da dissertação permitiu observar que quando não existe uma grande quantidade de casos disponíveis na base de casos, os agentes acabam

tomando decisões com base em experiências não tão similares recuperadas da base. Nesses casos, o mecanismo de reuso de soluções computa uma maior quantidade de ações de jogo possivelmente consideradas idênticas embora tendo sido realizadas em diferentes situações de jogo. Para solucionar esse problema, e realmente computar soluções a partir de ações de jogo executadas em cenário de jogo realmente similares, a exploração do modelo de reuso em dois passos viabilizou que os agentes conforme esse modelo obtivessem um número de vitórias consideravelmente maior em comparação com os demais agentes testados. Mais importante, a utilização de grupos de casos no processo de reuso de soluções permitiu aumentar a diferença de pontos obtidos quando os agentes estiveram à frente no placar e reduzir essa diferença quando eles estavam atrás no placar. Isso indica que os agentes implementados de acordo com as técnicas de reuso em dois passos propostas apresentaram um comportamento bastante competitivo. Em resumo, a aplicação de diferentes critérios nos modelos de reuso descritos nesta dissertação possibilitou observar comportamentos de jogo distintos nos agentes implementados. Dentre todas as políticas de reuso implementadas por esses agentes, o comportamento de jogo do critério PV para escolher o grupo de casos e do critério NP para selecionar a solução obteve os melhores resultados.

Como trabalhos futuros, o desenvolvimento de testes envolvendo um maior número de jogadores humanos e competições disputadas contra agentes implementados via outras técnicas de IA são elementos importantes para alcançar uma maior validação das técnicas de jogo de cartas desenvolvidas nesta dissertação. Além disso, acreditamos que os experimentos realizados contra jogadores humanos podem ser expandidos de forma a permitir a análise de outras características possivelmente observadas a partir dos agentes implementados. Mais ainda, julgamos que seria relevante a proposição e teste de novos comportamentos de jogo particularmente relacionados com o emprego e detecção de blefe em jogos de cartas. Outros trabalhos também podem investigar a construção de funções de similaridade mais ajustadas para as necessidades destes problemas de aplicação explorados, semelhante ao que foi realizado em (Lucca *et al.*, 2018) onde a análise de resultados de algoritmos de *clustering* permitiram construir funções de similaridade mais efetivas para sistemas CBR. A pesquisa desenvolvida nesta dissertação também pode ser expandida a partir da exploração de resultados que podem ser obtidos por outros tipos de algoritmos de *clustering*. Por fim, apesar de termos apresentado um modelo para construir políticas de reuso usadas por agentes em jogos de cartas, entendemos que essa abordagem de reuso em dois passos baseadas na utilização de clusters de casos pode trazer benefícios para outras aplicações onde decisões precisam ser tomadas em cenários de visão parcial.

REFERÊNCIAS BIBLIOGRÁFICAS

AAMODT, A.; PLAZA, E. Case-based reasoning: foundational issues, methodological variations, and system approaches. **AI communications,** v. 7, n. 1, p. 39-59, 1, March 1994. ISSN 0921-7126.

AMBEKAR, G.; CHIKANE, T.; SHETH, S.; SABLE, A.; GHAG, K. **Anticipation of winning probability in poker using data mining**. 2015 International Conference on Computer, Communication and Control (IC4). Indore, India: IEEE**:** 1-6 p. 2015.

AYED, S. B.; ELOUEDI, Z.; LEFÈVRE, E. **ECTD: Evidential Clustering and Case Types Detection for Case Base Maintenance**. 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA). Hammamet, Tunisia: IEEE**:** 1462-1469 p. 2017.

BACKHUS, J.; NONAKA, H.; YOSHIKAWA, T.; SUGIMOTO, M. **Application of reinforcement learning to the card game Wizard**. 2013 IEEE 2nd Global Conference on Consumer Electronics (GCCE). Tokyo, Japan: IEEE**:** 329-333 p. 2013.

BALAN, S.; OTTO, J. **Business Intelligence in Healthcare with IBM Watson Analytics**. 1. North Charleston, SC, USA: CreateSpace Independent Publishing Platform, 2017. 110 ISBN 1548829897.

BAUCKHAGE, C.; DRACHEN, A.; SIFA, R. Clustering Game Behavior Data. **IEEE Trans. on Comp. Intelligence and AI in Games,** v. 7, n. 3, p. 266-278, September 2015.

BELLAMY-MCINTYRE, J. **Applying Case-Based Reasoning to the Game of Bridge**. 2008. 61 (phd). PhD thesis, University of Auckland, Auckland, New Zealend.

BHOLOWALIA, P.; KUMAR, A. EBK-Means : A Clustering Technique based on Elbow Method and K-Means in WSN. **International Journal of Computer Applications,** v. 105, n. 9, p. 17-24, November 2014.

BROWN, N.; SANDHOLM, T. Superhuman AI for multiplayer poker. **Science,** v. 365, August 30 2019.

CANELLAS, M. C.; FEIGH, K. M.; CHUA, Z. K. Accuracy and Effort of Decision-Making Strategies With Incomplete Information: Implications for Decision Support System Design. **IEEE Transactions on Human-Machine Systems,** v. 45, n. 6, p. 686-701, December 2015.

CHEN, J.; DONG, W.; YANG, Y.; LIU, L. **The Application of Cluster Analysis Method in Case-Based Reasoning System**. 26th International Conference on Geoinformatics. Kunming, China: IEEE**:** 1-4 p. 2018.

FLOYD, M. W.; ESFANDIARI, B.; LAM, K. A Case-Based Reasoning Approach to Imitating RoboCup Players. FLAIRS Conference, 2008, Florida, USA. Aaai, May 15-17. p.251-256.

FUKUNAGA, K.; NARENDRA, P. M. A Branch and Bound Algorithm for Computing k-Nearest Neighbors. **IEEE Transactions on Computers,** v. 24, n. 7, p. 750-753, July 1975.

GÓMEZ-MARTÍN, M.; GÓMEZ-MARTÍN, P.; GONZALEZ-CALERO, P. **Game-Based Learning as a New Domain for Case-Based Reasoning**. 6th International Conference on Case-Based Reasoning, ICCBR. Chicago, IL, USA: Springer**:** 175-184 p. 2005.

GOW, J.; BAUMGARTEN, R.; CAIRNS, P.; COLTON, S.; MILLER, P. Unsupervised Modeling of Player Style With LDA. **IEEE Transactions on Computational Intelligence and AI in Games,** v. 4, n. 3, p. 152-166, September 2012. ISSN 1943-0698.

HARTIGAN, J. **Clustering Algorithms**. New York: John Wiley & Sons, 1975. 351.

HEINRICH, J.; SILVER, D. Self-Play Monte-Carlo Tree Search in Computer Poker. AAAI Workshops, 2014, North America. AAAI, June 2014.

HOOSHYAR, D.; YOUSEFI, M.; LIM, H. Data-Driven Approaches to Game Player Modeling: A Systematic Literature Review. **ACM Computer Survers,** v. 50, n. 6, p. 1-19, 6, January 2018. ISSN 0360-0300.

JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. Data clustering: a review. **ACM Computing Surveys,** v. 31, n. 3, p. 264-323, 3, September 1999.

JI, S.-H.; AHN, J.; LEE, E.-B.; KIM, Y. Learning method for knowledge retention in CBR cost models. **Automation in Construction,** v. 96, p. 65-74, 01, December 2018. ISSN 0926-5805. Disponível em: < http://www.sciencedirect.com/science/article/pii/S0926580517311020 >.

KHUSSAINOVA, G.; PETROVIC, S.; JAGANNATHAN, R. Retrieval with Clustering in a Case-Based Reasoning System for Radiotherapy Treatment Planning. **Journal of Physics: Conference Series,** v. 616, p. 012013, 2015/05/22 2015. ISSN 1742-6588

1742-6596. Disponível em: < http://dx.doi.org/10.1088/1742-6596/616/1/012013 >.

LI, H.; YU, J.-L.; YU, L.-A.; SUN, J. The clustering-based case-based reasoning for imbalanced business failure prediction: a hybrid approach through integrating unsupervised process with supervised process. **International Journal of Systems Science,** v. 45, n. 5, p. 1225-1241, May 2014. ISSN 0020-7721. Disponível em: < https://doi.org/10.1080/00207721.2012.748105 >.

LOPEZ DE MANTARAS, R.; MCSHERRY, D.; BRIDGE, D.; LEAKE, D.; SMYTH, B.; CRAW, S.; FALTINGS, B.; MAHER, M. L.; COX, M. T.; FORBUS, K. Retrieval, reuse, revision and retention in case-based reasoning. **The Knowledge Engineering Review,** v. 20, n. 03, p. 215-240, 2005. ISSN 1469-8005.

LORA, D.; SÁNCHEZ-RUIZ-GRANADOS, A. A.; GONZÁLEZ-CALERO, P. A.; GÓMEZ-MARTÍN, M. A. **Dynamic Difficulty Adjustment in Tetris**. Florida Artificial Intelligence Research Society Conference (FLAIRS 2016). Key Largo, Florida, USA: AAAI Press**:** 335-339 p. 2016.

LUCCA, M.; JUNIOR, A. L.; SILVA, L. A. D. L.; FREITAS, E. P. D. **A Case-Based Reasoning and Clustering Framework for the Development of Intelligent Agents in Simulation Systems**. Florida Artificial Intelligence Research Society Conference (FLAIRS 2018). Melbourne, Florida, USA: AAAI Press 399-402 p. 2018.

LUÍS FILIPE, T.; REIS, L. P. HoldemML: A framework to generate No Limit Hold'em Poker agents from human player strategies. 6th Iberian Conference on Information Systems and Technologies (CISTI 2011), 2011, Chaves, Portugal. IEEE, 15-18 June p.1-6.

MARKOFF, J. Computer wins on 'jeopardy!': trivial, it's not. **New York Times,** v. 16, 2011.

MOZGOVOY, M.; PURGINA, M.; UMAROV, I. Believable self-learning AI for world of tennis. 2016 IEEE Conference on Computational Intelligence and Games (CIG), 2016, Santorini, Greece. IEEE, 20-23 September. p.1-7.

NIKLAUS, J.; ALBERTI, M.; PONDENKANDATH, V.; INGOLD, R.; LIWICKI, M. Survey of Artificial Intelligence for Card Games and Its Application to the Swiss Game Jass. **6th Swiss Conference on Data Science (SDS),** v. abs/1906.04439, June, 14 2019. Disponível em: < http://arxiv.org/abs/1906.04439 >.

OH, I.-S.; CHO, H.; KIM, K.-J. Playing real-time strategy games by imitating human players' micromanagement skills based on spatial analysis. **Expert Systems with Applications,** v. 71, n. 1, p. 192-205, April, 1 2017.

OHIRA, S.; MAEDA, N.; NAGAO, K. **Player type classification based on activity logs in a gamified seminar setting**. International Conference on Game, Game Art, and Gamification (ICGGAG 2016). Jakarta, Indonesia: IEEE**:** 1-6 p. 2016.

PAULUS, G. B.; ASSUNCAO, J. V. C.; SILVA, L. A. D. L. Cases and Clusters in Reuse Policies for Decision-Making in Card Games. 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), 2019, Portland, OR, USA. 4-6 Nov. p.1361-1365.

RECIO-GARCÍA, J.; GONZALEZ-CALERO, P.; DÍAZ-AGUDO, B. jCOLIBRI2: A framework for building Case-based reasoning systems. **Science of Computer Programming,** v. 79, p. 126–145, January, 1 2014.

RICHTER, M. M.; WEBER, R. O. **Case-based reasoning**. Springer-Verlag Berlin Heidelberg, 2013. 546 ISBN 3662523779.

RUBIN, J.; WATSON, I. **Investigating the Effectiveness of Applying Case-Based Reasoning to the Game of Texas Hold'em**. Florida Artificial Intelligence Research Society Conference (FLAIRS 2007). Key West, Florida, USA: AAAI Press 2007.

RUBIN, J.; WATSON, I. **A Memory-Based Approach to Two-Player Texas Hold'em**. Australasian Conference on Artificial intelligence. Melbourne, Australia: Springer Berlin Heidelberg**:** 465-474 p. 2009.

RUBIN, J.; WATSON, I. **Similarity-Based Retrieval and Solution Re-use Policies in the Game of Texas Hold'em**. International Conference on Case-Based Reasoning (ICCBR 2010). Alessandria, Italy: Springer Berlin Heidelberg**:** 465-479 p. 2010.

RUBIN, J.; WATSON, I. Computer poker: A review. **Artificial Intelligence,** v. 175, n. 5, p. 958-987, April 01 2011. ISSN 0004-3702. Disponível em: < http://www.sciencedirect.com/science/article/pii/S0004370211000191 >.

RUBIN, J.; WATSON, I. Case-based strategies in computer poker. **AI Communications,** v. 25, n. 1, p. 19-48, January 2012.

SANDVEN, A.; TESSEM, B. **A case-based learner for poker**. The Ninth Scandinavian Conference on Artificial Intelligence (SCAI 2006). Helsinki, Finland 2006.

SIFA, R.; DRACHEN, A.; BAUCKHAGE, C.; THURAU, C.; CANOSSA, A. **Behavior evolution in Tomb Raider Underworld**. Conference on Computational Inteligence in Games (CIG 2013). Niagara Falls, ON, Canada IEEE**:** 1-8 p. 2013.

SILVER, D.; SCHRITTWIESER, J.; SIMONYAN, K.; ANTONOGLOU, I.; HUANG, A.; GUEZ, A.; HUBERT, T.; BAKER, L.; LAI, M.; BOLTON, A.; CHEN, Y.; LILLICRAP, T.; HUI, F.; SIFRE, L.; VAN DEN DRIESSCHE, G.; GRAEPEL, T.; HASSABIS, D. Mastering the game of Go without human knowledge. **Nature,** v. 550, n. 7676, p. 354-359, October 01 2017. ISSN 1476-4687. Disponível em: < https://doi.org/10.1038/nature24270 >.

SMITI, A.; ELOUEDI, Z. WCOID-DG: An approach for case base maintenance based on Weighting, Clustering, Outliers, Internal Detection and Dbsan-Gmeans. **J. Comput. Syst. Sci.,** v. 80, n. 1, p. 27-38, February 2014. ISSN 0022-0000.

TEÓFILO, L. F.; REIS, L. P. **Identifying Player's Strategies in No Limit Texas Hold'em Poker through the Analysis of Individual Moves**. The 16th Portuguese Conference on Artificial Intelligence (EPIA 2013). Angra do Heroísmo, Açores, Portugal: Springer-Verlag Berlin Heidelberg 2013.

UNGER, S. H. **Integrating CBR and BN for Decision Making with Imperfect Information: Exemplified by Texas Hold'em Poker**. 2011. (phd). Institutt for datateknikk og informasjonsvitenskap

VENTOS, V.; COSTEL, Y.; TEYTAUD, O.; VENTOS, S. T. Boosting a Bridge Artificial Intelligence. 2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI), 2017, Boston, MA, USA. IEEE, November 6-8. p.1280-1287.

WATSON, I.; RUBIN, J. Casper: A case-based poker-bot. Australasian Joint Conference on Artificial Intelligence, 2008, Auckland, New Zealand. Springer, Berlin, Heidelberg December 1-5. p.594-600.

WEHR, D.; DENZINGER, J. **Mining game logs to create a playbook for unit AIs**. Conference on Computational Intelligence and Games (CIG 2015). Tainan, Taiwan IEEE**:** 391-398 p. 2015.

WINNE, L. L. **Truco**. Ciudad Autónoma de Buenos Aires Ediciones Godot, 2017. ISBN 978-987-4086-27-3.

YANNAKAKIS, G. N.; TOGELIUS, J. **Artificial Intelligence and Games**. Springer Publishing Company, Incorporated, 2018. 337 ISBN 3319635182, 9783319635187.

APÊNDICE A – RECURSOS COMPUTACIONAIS UTILIZADOS

Para ser possível o desenvolvimento desta pesquisa, diferentes recursos computacionais foram utilizados em diferentes etapas:

- Desenvolvimento das soluções:
  - Descrição: 1 notebook asus com 8 GB de memória Ram, Hd ssd 250GB, Processador intel Core(TM) i7 -3610QM CPU @ 2.30GHz, Sistema operacional Linux Debian 8.10 64 - bits.
- Coleta de casos:
  - Descrição: 1 máquina virtual hospedada na Digital Ocean com 4 GB de memória ram, 80 GB de armazenamento, 2 vCPUs, Sistema Operacinal Linux Debian 8.10 64 – bits.
- Validação com seres humanos:
  - Descrição: 1 máquina virtual hospedada na Digital Ocean com 16 GB de memória ram, 320 GB de armazenamento, 6 vCPUs, Sistema Operacinal Linux Debian 8.10 64 – bits.
- Validação entre agentes:
  - Descrição: 3 computadores LG *all in one* com 4 GB de memória ram, 1TB de hd, Processador intel Core(TM) i5 – 4200M CPU @ 2.50GHz, Sistema Operacional Windows 10 home 64 – bits.
- Treinamento automático da base:
  - Descrição: 1 notebook hp com 4 GB de memória ram, Hd ssd 120 GB, processador intel Celeron (R) Dual- core CPU T3000 @ 1.80GHZ, Sistema operacional Linux Debian 8.10 64 - bits.

APÊNDICE B – DETALHAMENTO DOS AGRUPAMENTOS

Tabela 1 - Agrupamentos executados

Agrupamento	**Atributos**	**Número de grupos existentes na base final**
Primeira carta quando o agente é o primeiro a jogar	Carta Alta Recebida, Carta Média Recebida, Carta Baixa Recebida, Carta Jogada pelo agente.	4
Primeira carta quando o agente é o segundo a jogar	Carta Alta Recebida, Carta Média Recebida, Carta Baixa Recebida, Carta Jogada pelo oponente, carta jogada pelo agente.	4
Segunda carta quando ganhou a primeira	Segunda carta jogada pelo agente	4
Segunda carta quando perdeu a primeira	Segunda carta jogada pelo oponente, segunda carta jogada pelo agente	6
Terceira carta quando ganhou a segunda	Terceira carta jogada pelo agente	2
Terceira carta quando perdeu a segunda	Terceira carta jogada pelo oponente, Terceira carta jogada pelo agente	4

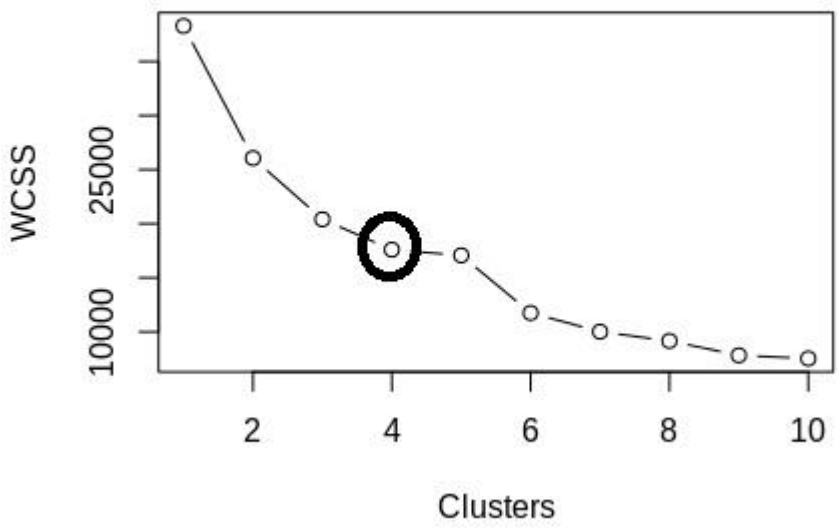
Quem ENVIDO quando o agente é o primeiro a apostar	Quem pediu envido, Quem pediu real envido, Quem pediu falta envido, pontos envido.	5
Quem ENVIDO quando o agente é o segundo a apostar	Quem pediu envido, Quem pediu real envido, Quem pediu falta envido, pontos envido.	5
Quem TRUCO na primeira rodada quando o agente é o primeiro a jogar	Quem pediu truco, Quem pediu retruco, Quem pediu vale quarto, Carta Alta, Carta Media, Carta Baixa.	8
Quem TRUCO na primeira rodada quando o agente é o Segundo a jogar	Quem pediu truco, Quem pediu retruco, Quem pediu vale quarto, Carta Alta, Carta Media, Carta Baixa.	4
Quem TRUCO na segunda rodada quando o agente ganhou a primeira	Quem pediu truco, Quem pediu retruco, Quem pediu vale quarto, Carta Alta, Carta Media, Carta Baixa.	6
Quem TRUCO na segunda rodada quando o agente perdeu a primeira	Quem pediu truco, Quem pediu retruco, Quem pediu vale quarto, Carta Alta, Carta Media, Carta Baixa.	4
Quem TRUCO na terceira rodada quando agente ganhou a segunda	Quem pediu truco, Quem pediu retruco, Quem pediu vale quarto, Carta Alta, Carta Media, Carta Baixa.	7

Quem TRUCO na terceira rodada quando agente perdeu a segunda	Quem pediu truco, Quem pediu retruco, Quem pediu vale quarto, Carta Alta, Carta Media, Carta Baixa.	4
Indexação da base para ações do tipo jogar cartas e apostas do tipo TRUCO	Carta Alta, Carta Média, Carta Baixa	8
Indexação da base para apostas do tipo ENVIDO	Pontos Envido	2

Fonte: do autor.

.

Figura 1 - *Plot elbow* da Primeira carta quando o agente é o primeiro a jogar



Fonte: do autor.

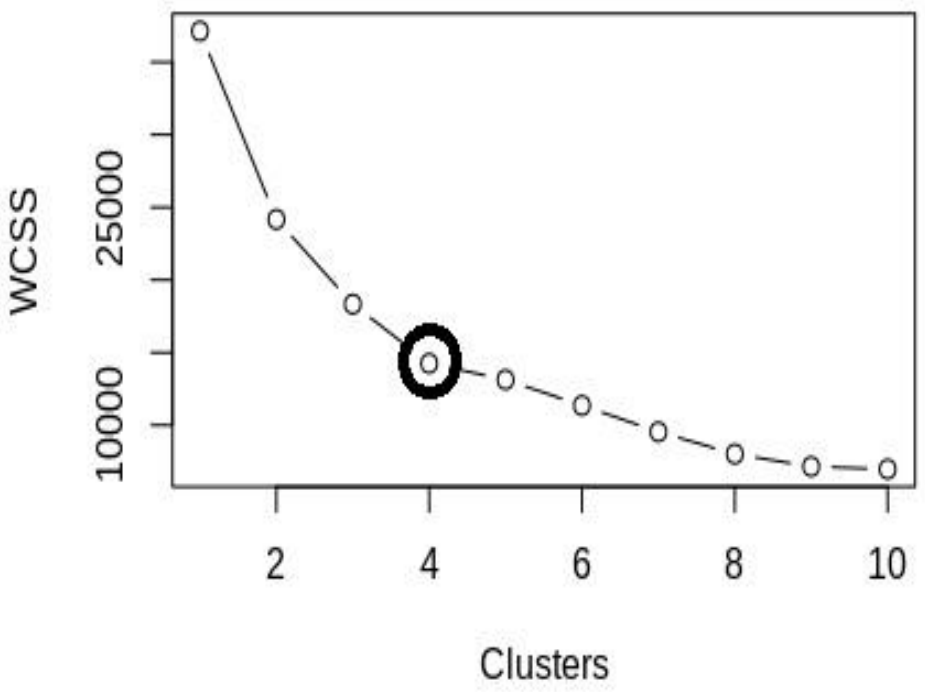
Tabela 2 – quantitativos referentes ao agrupamento da Primeira carta quando o agente é o primeiro a jogar

Grupo	Quantidade de casos pertencentes ao grupo	Quantidade totais de casos no cenário
1	4573	9573

2	1246	
3	1394	
4	2360	

Fonte: do autor.

Figura 2 - *Plot elbow* da Primeira carta quando o agente é o segunda a jogar



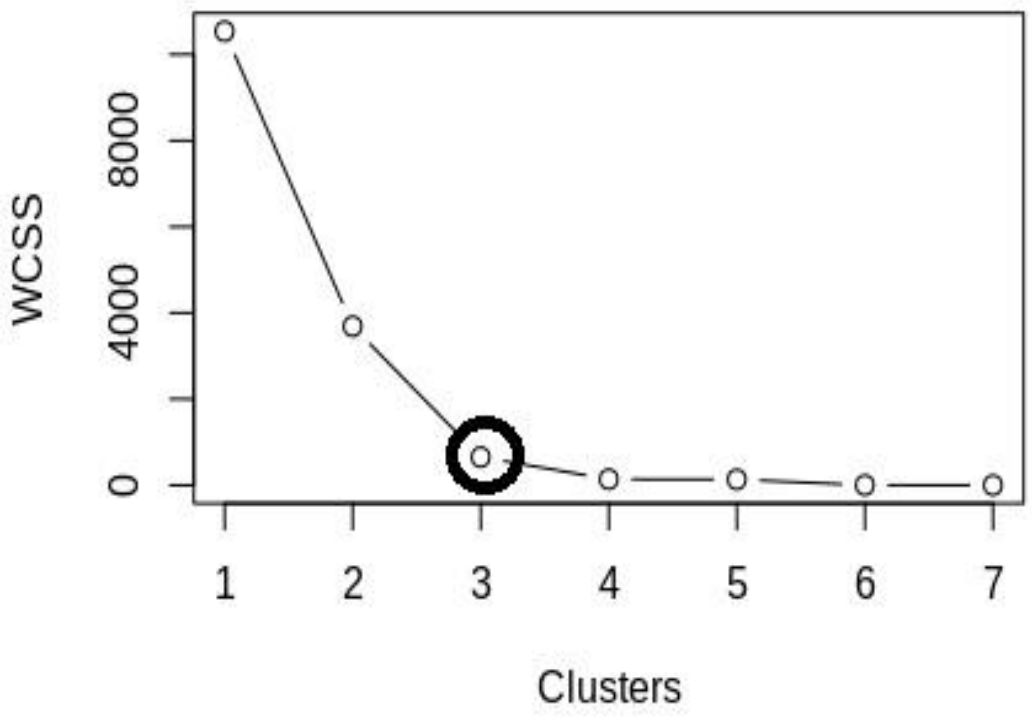
Fonte: do autor.

Tabela 3 - quantitativos referentes ao agrupamento da Primeira carta quando o agente é o primeiro a jogar

Grupo	Quantidade de casos pertencentes ao grupo	Quantidade totais de casos no cenário
1	1764	7427
2	2063	
3	484	
4	3116	

Fonte: do autor.

Figura 3 - *Plot elbow* da segunda carta quando o agente ganhou a primeira rodada



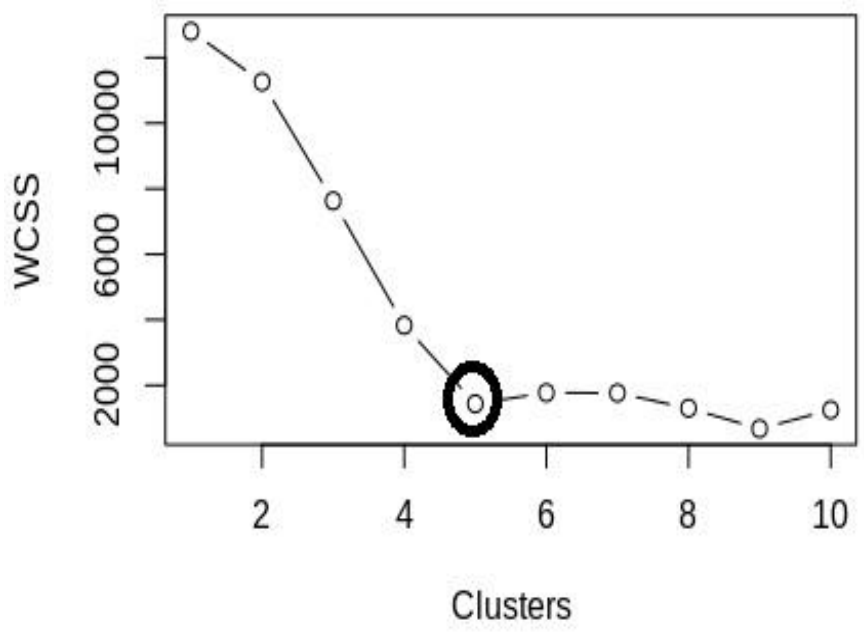
Fonte: do autor.

Tabela 4 - quantitativos referentes ao agrupamento da segunda carta quando o agente ganhou a primeira rodada

Grupo	Quantidade de casos pertencentes ao grupo	Quantidade totais de casos no cenário
1	2661	10539
2	1961	
3	5917	

Fonte: do autor.

Figura 4 - *Plot elbow* da segunda carta quando o agente perdeu a primeira rodada



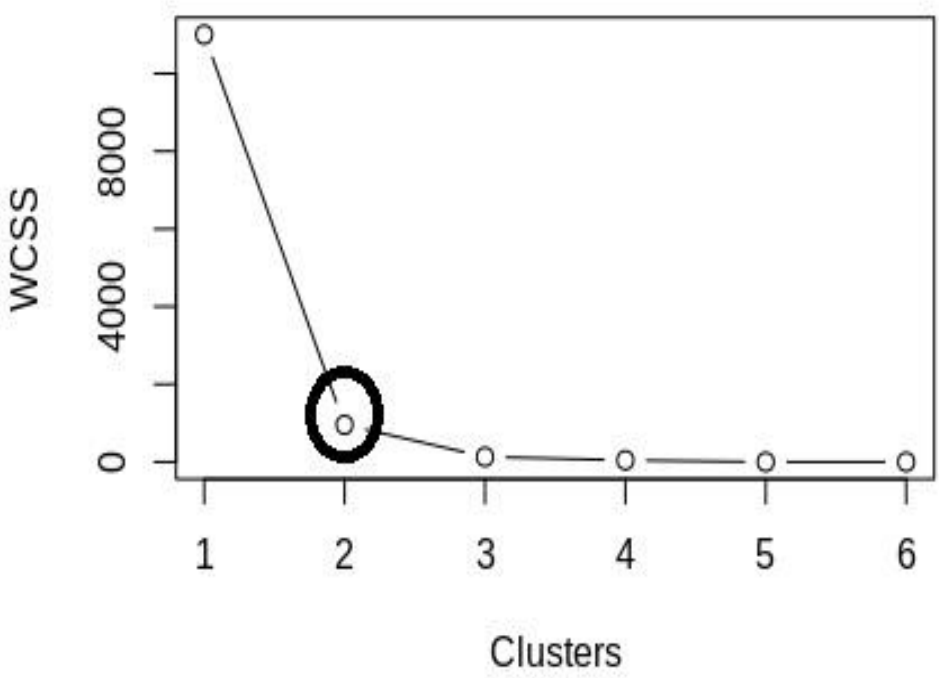
Fonte: do autor.

Tabela 5 - quantitativos referentes ao agrupamento da segunda carta quando o agente ganhou a primeira rodada

Grupo	Quantidade de casos pertencentes ao grupo	Quantidade totais de casos no cenário
1	56	6402
2	2342	
3	310	
4	27	
5	3667	

Fonte: do autor.

Figura 5 - *Plot elbow* da terceira carta quando o agente ganhou a segunda rodada



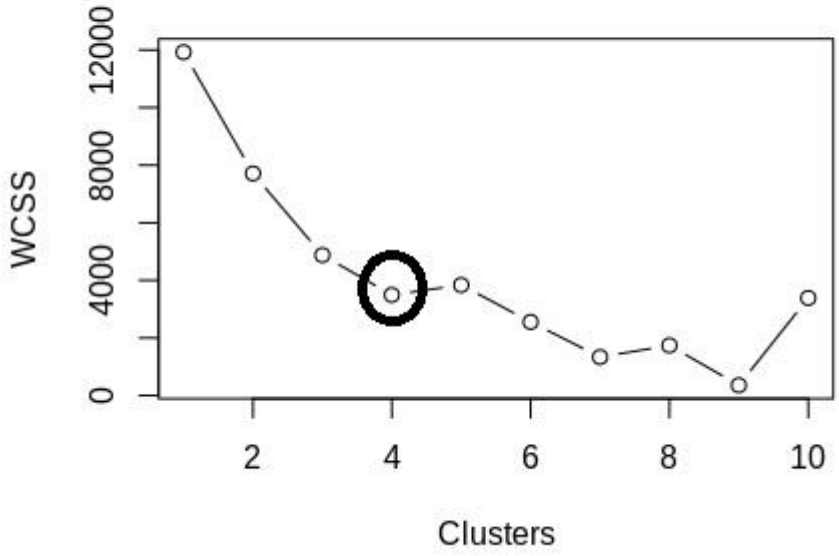
Fonte: do autor.

Tabela 6- quantitativos referentes ao agrupamento da terceira carta quando o agente ganhou a segunda rodada

Grupo	Quantidade de casos pertencentes ao grupo	Quantidade totais de casos no cenário
1	3747	11003
2	7256	

Fonte: do autor.

Figura 6 - *Plot elbow* da terceira carta quando o agente perdeu a segunda rodada



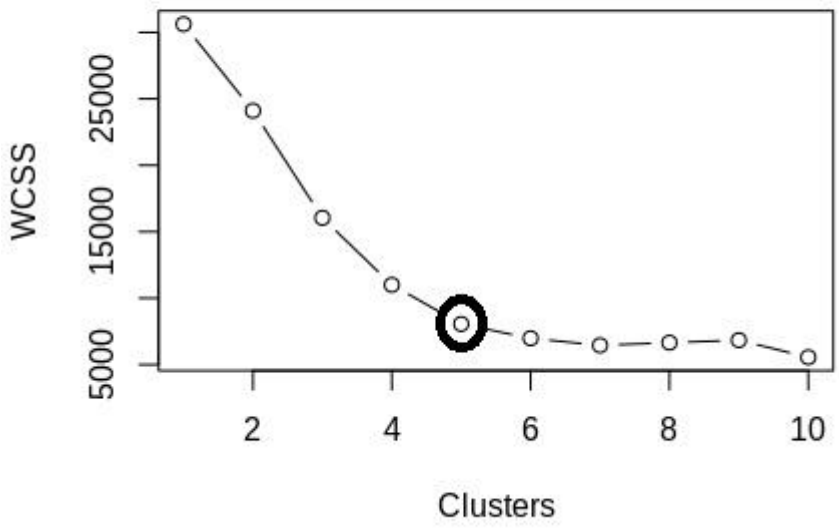
Fonte: Autor.

Tabela 7 - quantitativos referentes ao agrupamento da terceira carta quando o agente perdeu a segunda rodada

Grupo	Quantidade de casos pertencentes ao grupo	Quantidade totais de casos no cenário
1	593	5962
2	4581	
3	657	
4	131	

Fonte: do autor.

Figura 7 - *Plot elbow* quem ENVIDO quando o agente é o primeiro a apostar



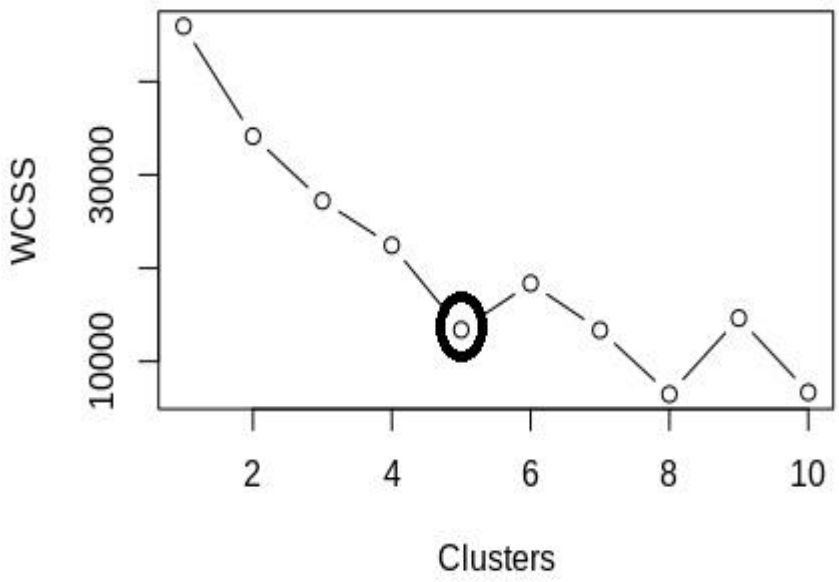
Fonte: do autor.

Tabela 8 - quantitativos referentes ao agrupamento quem ENVIDO quando o agente é o primeiro a apostar

Grupo	Quantidade de casos pertencentes ao grupo	Quantidade totais de casos no cenário
1	2017	7654
2	740	
3	943	
4	1332	
5	2622	

Fonte: do autor.

Figura 8 - *Plot elbow* quem ENVIDO quando o agente é o segundo a apostar



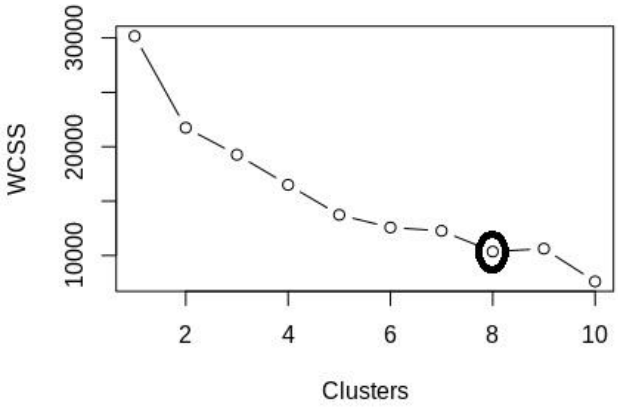
Fonte: do autor.

Tabela 9 - quantitativos referentes ao agrupamento quem ENVIDO quando o agente é o segundo a apostar

Grupo	Quantidade de casos pertencentes ao grupo	Quantidade totais de casos no cenário
1	1646	11486
2	3062	
3	2897	
4	1456	
5	2425	

Fonte: do autor.

Figura 9 - *Plot elbow* referente ao agrupamento quem TRUCO quando o agente é o primeiro a apostar



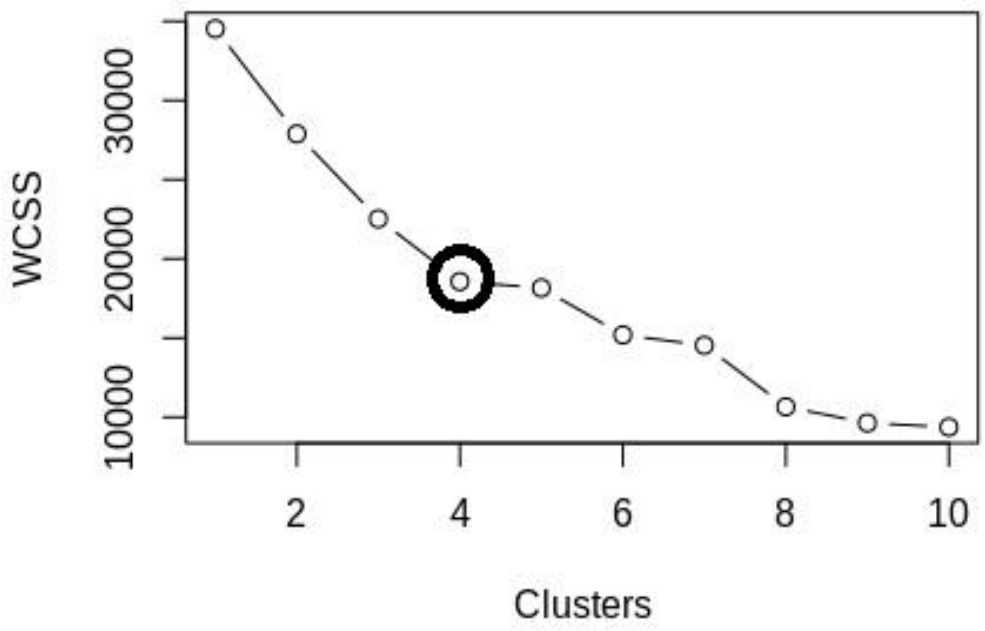
Fonte: do autor.

Tabela 10 - quantitativos referentes ao agrupamento quem TRUCO na primeira rodada quando o agente é o primeiro a jogar

Grupo	Quantidade de casos pertencentes ao grupo	Quantidade totais de casos no cenário
1	10	5029
2	130	
3	3	
4	73	
5	72	
6	1484	
7	55	
8	3202	

Fonte: do autor.

Figura 10 - *Plot elbow* referente ao agrupamento quem TRUCO na primeira rodada quando o agente é o segundo a apostar



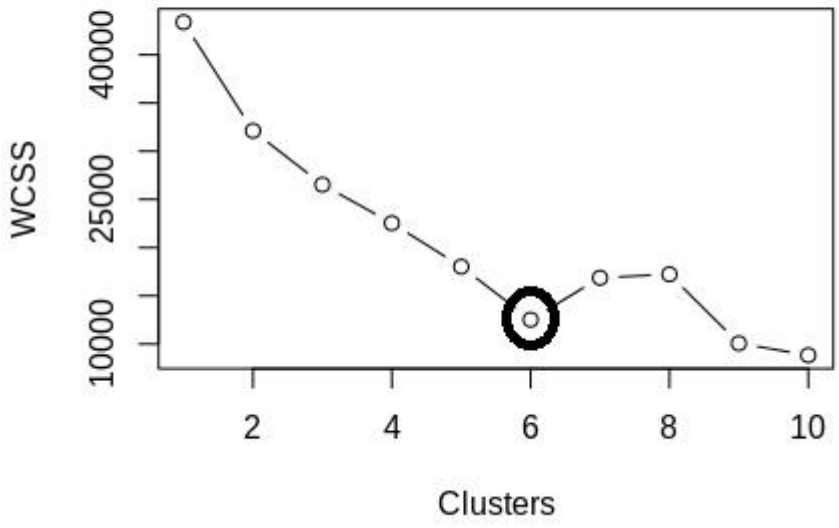
Fonte: do autor.

Tabela 11 - quantitativos referentes ao agrupamento quem TRUCO na primeira rodada quando o agente é o segundo a jogar

Grupo	Quantidade de casos pertencentes ao grupo	Quantidade totais de casos no cenário
1	784	5755
2	2639	
3	245	
4	2087	

Fonte: do autor.

Figura 11 - *Plot elbow* referente ao agrupamento quem TRUCO na segunda rodada quando o agente ganhou a primeira



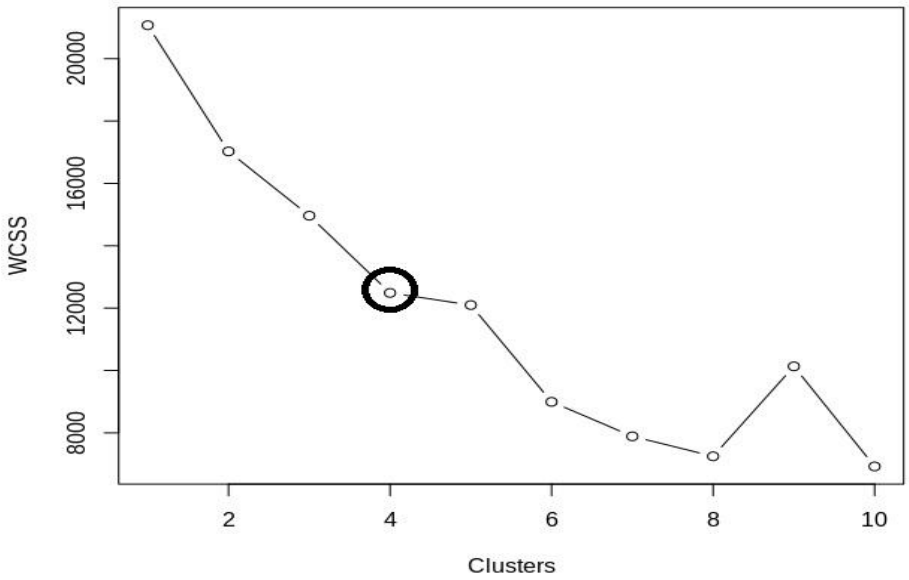
Fonte: do autor.

Tabela 12 - quantitativos referentes ao agrupamento quem TRUCO na segunda rodada quando o agente ganhou a primeira

Grupo	Quantidade de casos pertencentes ao grupo	Quantidade totais de casos no cenário
1	200	7231
2	1434	
3	1467	
4	2210	
5	1493	
6	427	

Fonte: do autor.

Figura 12 - *Plot elbow* referente ao agrupamento quem TRUCO na segunda rodada quando o agente perdeu a primeira



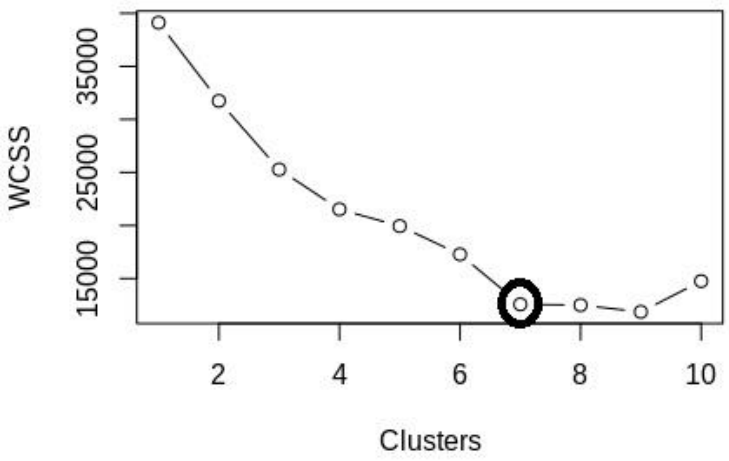
Fonte: do autor.

Tabela 18 - quantitativos referentes ao agrupamento quem *TRUCO* na segunda rodada quando o agente perdeu a primeira

Grupo	Quantidade de casos pertencentes ao grupo	Quantidade totais de casos no cenário
1	662	3513
2	1497	
3	1040	
4	314	

Fonte: do autor.

Figura 13 - *Plot elbow* referente ao agrupamento quem TRUCO na terceira rodada quando o agente ganhou a segunda



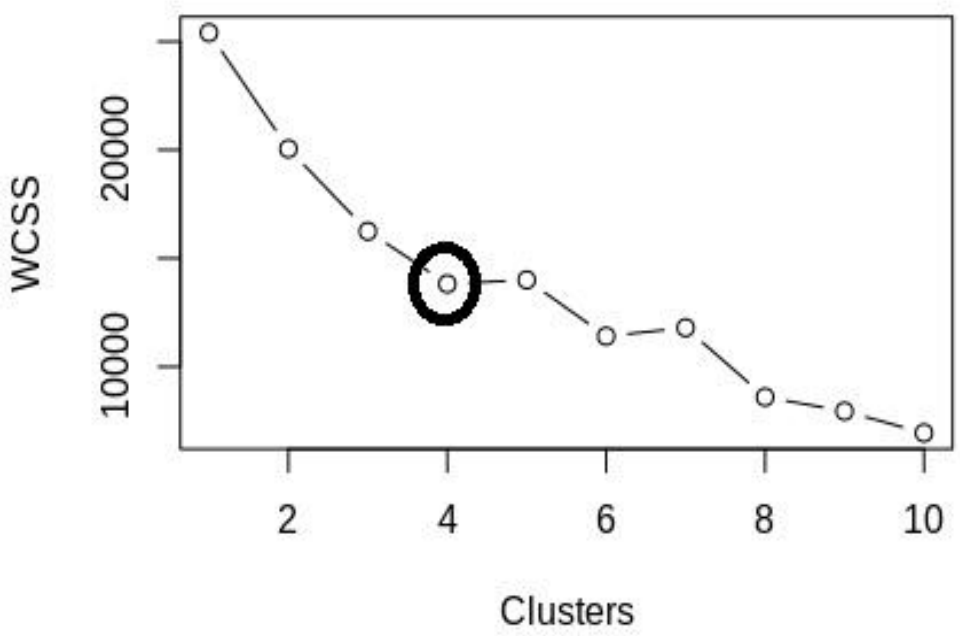
Fonte: do autor.

Tabela 19 - quantitativos referentes ao agrupamento quem TRUCO na terceira rodada quando o agente ganhou a segunda

Grupo	Quantidade de casos pertencentes ao grupo	Quantidade totais de casos no cenário
1	2177	6520
2	100	
3	308	
4	1100	
5	961	
6	1120	
7	754	

Fonte: do autor.

Figura 14 - *Plot elbow* referente ao agrupamento quem TRUCO na terceira rodada quando o agente perdeu a segunda



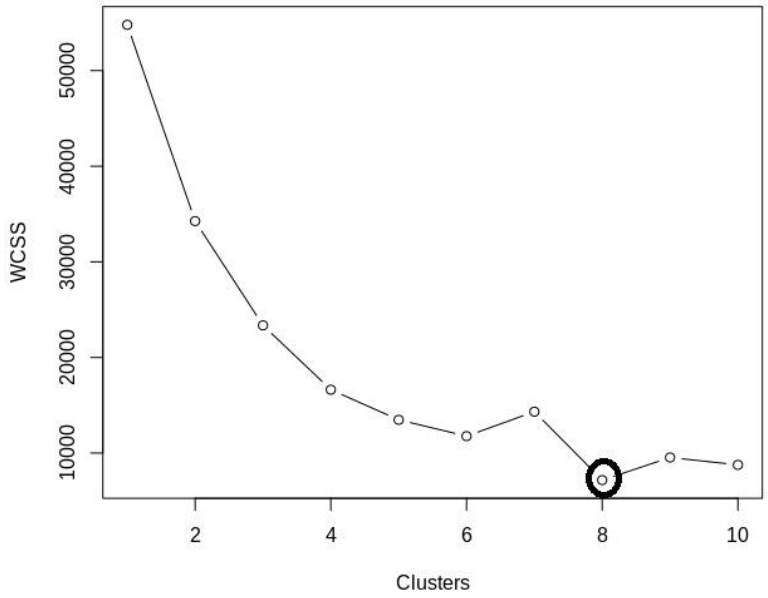
Fonte: do autor.

Tabela 15 - quantitativos referentes ao agrupamento quem TRUCO na terceira rodada quando o agente perdeu a segunda

Grupo	Quantidade de casos pertencentes ao grupo	Quantidade totais de casos no cenário
1	2690	4235
2	425	
3	904	
4	216	

Fonte: do autor.

Figura 15 - *Plot elbow* referente ao agrupamento para indexação de jogadas



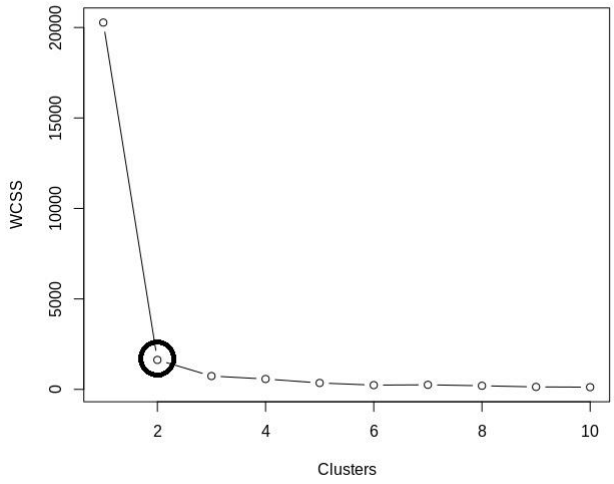
Fonte: do autor.

Tabela 16 - quantitativos referentes ao agrupamento para indexação de jogadas

Grupo	Quantidade de casos pertencentes ao grupo	Quantidade totais de casos uteis para ações do tipo jogada (apostas do tipo TRUCO e ações de jogar cartas)
1	377	18263
2	10511	
3	3415	
4	271	
5	222	
6	218	
7	3144	
8	105	

Fonte: do autor.

Figura 16 - Plot elbow referente ao agrupamento para indexação de pontos



Fonte: do autor.

Tabela 17 - quantitativos referentes ao agrupamento para indexação de pontos

Grupo	Quantidade de casos pertencentes ao grupo	Quantidade totais de casos uteis para apostas do tipo ENVIDO (apostas do tipo ENVIDO e do tipo FLOR)
1	7244	20273
2	13029	

Fonte: do autor.