UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS FLORIANÓPOLIS
CURSO DE GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO

Gustavo Emanuel Kundlatsch

HAIL: um modelo de revisão de percepções de agentes inteligentes baseado em alucinação e ilusão

Florianópolis
2021

Gustavo Emanuel Kundlatsch

HAIL: um modelo de revisão de percepções de agentes inteligentes baseado em alucinação e ilusão

Monografia submetida ao Curso de Graduação em Ciências da Computação da Universidade Federal de Santa Catarina para a obtenção do título de bacharel em Ciências da Computação.
Orientador: Dr. Thiago Ângelo Gelaim
Coorientador: Prof. Dr. Elder Rizzon Santos

Florianópolis
2021

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Kundlatsch, Gustavo Emanuel
HAIL: um modelo de revisão de percepções de agentes inteligentes baseado em alucinação e ilusão / Gustavo Emanuel Kundlatsch ; orientador, Thiago Ângelo Gelaim, coorientador, Elder Rizzon Santos, 2021.
115 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Ciências da Computação, Florianópolis, 2021.

Inclui referências.

1. Ciências da Computação. 2. Agentes. 3. Percepção. 4. Anomalias. I. Gelaim, Thiago Ângelo. II. Santos, Elder Rizzon. III. Universidade Federal de Santa Catarina. Graduação em Ciências da Computação. IV. Título.

Gustavo Emanuel Kundlatsch

HAIL: um modelo de revisão de percepções de agentes inteligentes baseado em alucinação e ilusão

O presente trabalho em nível de bacharel foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Dr. Thiago Ângelo Gelaim
Orientador

Prof. Dr. Elder Rizzon Santos
Coorientador e Responsável

Prof. Dr. Rafael de Santiago

Me. Rodrigo Rodrigues Pires de Mello

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de bacharel em Ciências da Computação.

_____

Coordenação do Programa de Graduação

_____

Dr. Thiago Ângelo Gelaim
Orientador

Florianópolis, 2021.

Para meus pais, que sempre souberam que o melhor
investimento para os filhos é a educação.

AGRADECIMENTOS

Quando fazemos uma escolha, ela recebe o peso de toda a potencialidade morta das alternativas que não foram escolhidas. Sou muito grato a minha família, que sempre me deu toda a liberdade do mundo para tomar minhas próprias escolhas e que também sempre me apoiou, no acerto ou no erro, independente da dificuldade enfrentada. Eu escolhi cursar computação, e mesmo que minha família não entenda muito bem o que isso significa, sempre esteve comigo me incentivando a seguir meus sonhos.

Este trabalho não representa o ciclo final de minha graduação, mas todo o caminho trilhado nesses anos de curso. Ele é resultado de um processo que começou no fim do meu primeiro semestre, um processo que me fez conhecer muitas pessoas incríveis e que me desenvolveu tanto profissionalmente quanto como pessoa. O laboratório sempre foi para mim uma forma de expressar minha criatividade, e esta monografia é resultado de apenas uma das muitas ideias que eu tive. Por essa liberdade de criar e por todas as reuniões de orientação que tive ao longo desses anos eu agradeço ao professor Elder e ao Thiago, que tiveram a paciência de me acolher na minha iniciação científica.

A maior benção que recebi na vida foram meus amigos, e não posso deixar de agradecê-los aqui. Seja para jogar, reclamar da vida, chorar as mágoas ou simplesmente desfrutar da companhia uns dos outros, meus amigos nunca me deixaram na mão. Eu sempre quis ser um contador de histórias, e compartilhar meus causos com meus amigos, distribuindo sorrisos e gargalhadas, é algo que dinheiro nenhum no mundo seria capaz da comprar.

Por fim, gostaria de agradecer cada servidor da UFSC e profissional da comunidade, que são responsáveis pela alimentação, limpeza e organização de nossa universidade pública e gratuita. Sem o trabalho de cada uma dessas pessoas, essa conquista não seria possível.

“Qualquer lugar é o paraíso desde que queira viver.
Afinal, todos estão vivos.
E, enquanto for assim, todos têm chance de ser feliz.”
(Anno, End of Evangelion, 1997)

RESUMO

Percepções são a principal maneira de uma entidade receber informações do ambiente. Cada pessoa possui uma maneira diferente de perceber e interpretar o mundo. Entretanto, sabe-se que na percepção humana existem anomalias, que são percepções incorretas que enganam a mente. Dito isso, como podemos saber se nossas percepções são reais ou se são apenas fruto de nossa imaginação? E a questão derivada disso é: e computadores? Agentes inteligentes são entidades computacionais autônomas capazes de tomar decisões baseadas no ambiente no qual estão inseridos, e utilizam sensores para reconhecerem o mundo a sua volta. Mas esses sensores podem falhar. Neste trabalho, apresentamos um modelo genérico de revisão de percepções, capaz de tratar anomalias recebidas pelo agente e criar novos planos a partir delas para se adaptar ao ambiente. Esse modelo foi implementado e submetido a experimentos para testar seu funcionamento. As simulações realizadas demonstraram que o modelo é capaz de detectar e classificar as anomalias, e que em ambientes com grande quantidade de percepções inválidas o modelo consegue ter maior impacto criando novos planos através do planejamento automatizado e aumentando a quantidade de percepções válidas recebidas pelo raciocínio do agente.

Palavras-chave: Agentes. Percepção. Anomalias.

ABSTRACT

Perceptions are the primary way for an entity to receive information from the environment. Each person has a different manner of percept and interpret the world. However, it is known that in human perception, there are anomalies, incorrect perceptions that deceive the mind. So, how can we know if our perceptions are real or just tricks from our minds? Furthermore, the follow-up question is: what about computers? Intelligent agents are computational entities capable of making decisions based on the environment and use sensors to perceive the world. However, sensors can fail. In the current work, we present a generic model of perception revision, capable of treating anomalies received by the agent and creating new plans to adapt to the environment. This model was implemented and submitted to experiments to test its behaviour. The simulations demonstrate that the model is capable of detecting and classifying anomalies. In environments with a high amount of invalid perceptions, the model can have more impact creating new plans through the automatic planning process and increasing the number of valid perceptions received by the agent's cognition.

Keywords: Agents. Perception. Anomalies.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BDI	*Belief-Desire-Intention*
EOM	*Environment Object Model*
IA	Inteligência Artificial
K-CoPMan	*Knowledge enabled Cognitive Perception for Manipulation*
NPC	Número de Percepções recebidas por Ciclo
OA-SMTRNN	*Object Augmented Supervised Multiple Timescale Recurrent Neural Network*
PDDL	*Planning Domain Definition Language*
PMK	*Perception and Manipulation Knowledge*
PPI	Porcentagem de Percepções Inválidas
RP	Relação Percentual
SMA	Sistema Multiagente
TMA	Tempo Médio gasto pelo planejamento Automatizado
TMC	Tempo Médio gasto em um Ciclo de raciocínio

LISTA DE SÍMBOLOS

Δ	Função de transição do modelo de revisão de percepções
Γ	Função de transição que especifica um sistema de transição de estados Σ
γ	Função de percepção do agente
θ	Função de refinamento
ρ	Conjunto de percepções refinadas
Σ	Sistema de transição de estados de um modelo conceitual de planejamento automatizado
Ψ	Conjunto união formado pelas pré-condições das ações que compõem um plano
ψ	Conjunto de pré-condições de uma ação
Ω	Conjunto união formado pelas pós-condições das ações que compõem um plano
ω	Conjunto de pós-condições de uma ação
A	Conjunto finito ou recursivamente enumerável de ações
Ab	Conjunto de blocos avaliadores
Ab_h	Bloco avaliador de alucinações
Ab_{i1}	Bloco avaliador de ilusões classe 1
Ab_{i2}	Bloco avaliador de ilusões classe 2
Ag	Agente
Ap	Conjunto de blocos de planejamento automatizado
Ap_h	Bloco de planejamento automatizado de alucinações
Ap_i	Bloco de planejamento automatizado de ilusões
c	Contexto do agente
Ce	Função equação de limpeza do bloco avaliador
Cf	Função de limpeza do bloco avaliador

D	Conjunto de decisores
d_a	Decisor de anomalias
d_h	Decisor de alucinações
d_i	Decisor de ilusões
E	Conjunto finito ou recursivamente enumerável de eventos
K	Conjunto de conhecimentos do agente
L	Lista ordenada
$\|L\|$	Número de elementos de uma lista ordenada
L_i	Elemento i da lista ordenada L
M_{ai}	Módulo de alucinação e ilusão
P	Conjunto de planos do agente
p	Conjunto de percepções iniciais
$P(L_i)$	Função peso da lista ponderada
Pf	Função de processamento do bloco avaliador
S	Conjunto finito ou recursivamente enumerável de estados
$T_m(x)$	Função tempo médio de x
W	Função peso de uma anomalia
Z	Descrição de sistema

SUMÁRIO

1 INTRODUÇÃO

Dentro da Inteligência Artificial (IA), agentes inteligentes são entidades capazes de raciocinar a respeito do ambiente em que estão inseridos e tomar decisões baseadas na situação em que se encontram (RUSSELL, S. J.; NORVIG, 2016). Dessa maneira, podemos descrever um agente pelos seus processos de percepção, raciocínio e atuação. O agente ocupa um ambiente, do qual recebe informações e no qual atua. O ambiente é o mundo em que o agente está inserido, podendo ser virtual ou uma parte do mundo real, no caso de um agente físico. Existem diversos tipos de ambientes que podem ser classificados de acordo com o seu fechamento (que determina se agentes de fora do ambiente podem afetar o sistema), dinamismo (a maneira como o ambiente evolui), determinismo (a consistência dos efeitos no ambiente) e cardinalidade (o número de objetos a serem afetados e percebidos) (MOYA; TOLK, 2007).

Uma das maneiras de um agente atualizar seu conhecimento a respeito do ambiente é a percepção, o processo de utilizar sensores para detectar o ambiente e transformar os dados coletados em informações úteis (WEYNS *et al.*, 2004). O raciocínio, por sua vez, é o processamento das percepções baseado nos objetivos do agente, que resulta em um conjunto ações a serem tomadas através dos atuadores. O processo do raciocínio é comandado pela arquitetura cognitiva do agente, um modelo computacional inspirado na estrutura da mente humana (DYACHENKO *et al.*, 2018). As arquiteturas cognitivas podem ser divididas em três categorias: simbólicas, emergentes e híbridas (YE *et al.*, 2018). Arquiteturas simbólicas descrevem o ambiente através de símbolos armazenados em memória em uma base de conhecimentos, e utilizam lógica simbólica para realizar o ciclo de percepção, raciocínio e ação. Arquiteturas emergentes se baseiam na estrutura biológica do cérebro e normalmente utilizam redes neurais em uma estrutura hierárquica para lidar com situações de incerteza. Por fim, arquiteturas híbridas combinam o comportamento emergente e o processamento simbólico para resolver problemas de diversos domínios.

Todavia, sensores podem apresentar problemas para o processo de percepção por razões como campo de visão, distância do objeto observado, resolução dos sensores e leituras não confiáveis (CHRISMAN *et al.*, 1991). Tratar deste problema normalmente é responsabilidade da arquitetura cognitiva do agente, pois a arquitetura precisa ser capaz de fazer a ponte entre o ambiente e o conhecimento do agente (LANGLEY *et al.*, 2009).

O objetivo deste trabalho é apresentar um modelo genérico (independente da arquitetura do agente) que pode ser acoplado entre o processo de percepção e raciocínio, capaz de detectar e tratar percepções inválidas para transformá-las em informações úteis através de um processo de criação de novos planos. Esse modelo pressupõe

um ambiente aberto (onde agentes externos podem influenciar o ambiente), dinâmico (mudanças no ambiente são causadas por eventos aleatórios) e não determinístico (ações do agente causam resultados diferentes no ambiente, mesmo em situações aparentemente idênticas, pois os resultados variam dependendo da percepção do agente daquele evento). Com isso, buscamos responder as perguntas de pesquisa:

- Como podemos diferenciar percepções válidas de percepções inválidas?
- Como podemos utilizar percepções inválidas para criar novos planos?

1.1 JUSTIFICATIVA

Para que um agente possa interagir com o ambiente, ele precisa atualizar seus estados internos, de acordo com regras estabelecidas por sua arquitetura cognitiva. A percepção é a capacidade do agente sentir o mundo, resultando em uma expressão que por si própria pode ser entendida e interpretada (WEYNS *et al.*, 2004). Em ambientes dinâmicos, há possivelmente centenas de percepções por segundo (HAYES-ROTH, B. *et al.*, 1992). Mas percepções não necessariamente precisam incluir representações corretas da realidade e podem variar de agente para agente (JANSSEN, 2005). Um agente com percepção incompleta pode ter problemas de percepções por conta de limitações de sua capacidade de perceber determinados objetos ou obstrução física dos sensores, por exemplo (CHRISMAN *et al.*, 1991).

Falhas na entrada de dados através de sensores pode ser perigoso, levando o agente a tomar decisões incorretas, especialmente caso o agente faça parte de um sistema crítico. Por exemplo, em um sistema controlado por um agente cujo objetivo é trocar a cor do semáforo sempre que detectar pedestres o suficiente vindo de diversas direções, o mal funcionamento de um sensor pode levar a cor do semáforo a trocar com muita frequência, trancando os carros e levando a um grande engarrafamento – ou pior, pode causar o atropelamento de um pedestre.

Para categorizar as percepções incorretas e diferenciá-las das percepções válidas, foram utilizados dois conceitos vindos do estudo de percepção da filosofia: (i) alucinações, que são percepção completamente válidas, mas que ocorrem em um momento errado e (ii) ilusões, percepções ou de objetos existentes no ambiente com características incorretas, ou de objetos inexistentes no ambiente mas com características comuns a objetos válidos (CRANE; FRENCH, 2017).

Com base no nome em inglês desses dois conceitos foi definido o nome do modelo proposto neste trabalho: HAIL (*hallucination* e *illusion*, alucinação e ilusão em inglês, respectivamente).

1.2 OBJETIVOS

1.2.1 Objetivo geral

Este trabalho possui como objetivo geral propor um modelo de revisão de percepções capaz de detectar percepções inválidas e transformá-las em informações úteis para o agente.

1.2.2 Objetivos específicos

1. Analisar o estado da arte relativo a percepções de agentes inteligentes;
2. Criar um modelo capaz de detectar, classificar e aprender com percepções inválidas;
3. Formalizar e implementar o modelo proposto em um ambiente genérico (sem contexto de aplicação);
4. Testar o modelo proposto através de diferentes simulações, utilizando o design fatorial de experimentos.

1.3 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado do seguinte modo:

- O capítulo 2 apresenta e define formalmente a fundamentação teórica necessária para a compreensão do trabalho, dentre eles inteligência artificial, agente inteligente, percepção e planejamento automatizado;
- O capítulo 3 é um compilado de alguns trabalhos relacionados, que exploram o problema de percepções inválidas de diferentes maneiras;
- O capítulo 4 apresenta e define formalmente o modelo HAIL;
- O capítulo 5 descreve os experimentos realizados para testar o modelo HAIL e apresenta seus resultados;
- O capítulo 6 contém as considerações finais e os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo é apresentada a fundamentação teórica utilizada para o desenvolvimento do trabalho. Ele apresenta os conceitos básicos de inteligência artificial e um estudo aprofundado sobre agentes e percepções. Além disso, este capítulo apresenta definições que serão utilizadas na formalização do modelo no Capítulo 4.

2.1 INTELIGÊNCIA ARTIFICIAL

A Inteligência Artificial é um campo de estudo que se difere, pois enquanto outros campos do conhecimento se limitam a buscar entender *como* o pensamento humano funciona, a IA se propõe a construir entidades pensantes (RUSSEL; NORVIG *et al.*, 2013). Apesar de ser uma área de pesquisa que surgiu na década de 50 (MOOR, 2006), não existe uma única definição de Inteligência Artificial nem consenso dentro da comunidade acadêmica.

Uma visão prática do que é IA é sua abordagem como a automatização de diversas tarefas humanas: “atividades que associamos ao pensamento humano, atividades como a tomada de decisões, a resolução de problemas, o aprendizado” (BELLMAN, 1978). Essa definição é derivada da pergunta “computadores podem pensar?”. Existem diversos exemplos de atividades do pensamento humano que os computadores são capazes de executar, como tratamento de incerteza, consciência e humor. Apesar disso, para Bellman, “o espírito humano se mantém muito acima de qualquer coisa que possa ser automatizada”. Essa é uma visão de que o objetivo da IA é replicar o pensamento humano.

Um outro ponto de vista coloca a IA como ferramenta de investigação da mente humana, ou seja, como “o estudo das faculdades mentais através do uso de modelos computacionais” (CHARNIAK, 1985). Para essa definição estar correta é necessário que exista uma equivalência entre o processo mental humano e o processamento de um computador. Por conta disso, Charniak define o dogma central da Inteligência Artificial: “O que o cérebro faz pode ser pensado em algum nível como um tipo de computação”. Caso o dogma se mostre verdadeiro, o uso de modelos computacionais para o estudo das faculdades mentais é valido. “Faculdades mentas”, dentro dessa definição, são os mecanismos internos que recebem imagens e palavras (através da visão e da linguagem) e os converte em saídas na forma de ações robóticas e fala. Esse processamento interno inclui dedução, planejamento, aprendizado e outras técnicas.

As definições de IA podem ser divididas entre aquelas que defendem que os computadores devem pensar como humanos, e aquelas que defendem que os computadores devem agir como humanos (RUSSEL; NORVIG *et al.*, 2013). As duas definições apresentadas anteriormente estão no primeiro grupo. As definições que levam

em conta que os computadores devem agir como humanos em geral possuem um aspecto mais prático. Por exemplo, a definição dos autores Rich e Knight diz que inteligência artificial é "o estudo de como os computadores podem desempenhar tarefas que hoje são melhor desempenhadas pelas pessoas" (RICH; KNIGHT, 1991). Com isso podemos notar que esse lado mais prático da IA não precisa se preocupar tanto com as questões filosóficas por trás do pensamento humano, pois foca em resolver problemas reais através dos métodos existentes.

Neste trabalho, utilizaremos uma definição nessa mesma linha de pensamento de que o computador deve agir como um ser humano, nesse caso específico, de maneira lógica: "inteligência computacional é o estudo do desenvolvimento de agentes inteligentes" (POOLE *et al.*, 1998). Essa definição foi escolhida pois o HAIL se trata de um modelo desenvolvido para agentes inteligentes. A Seção 2.2 se dedica a definir o que é um agente.

Independente da definição utilizada, podemos afirmar que a IA é um campo vasto que intriga muitos pesquisadores. Esse campo possui diversas técnicas, que são utilizadas para resolver todo o tipo de problemas. Kurzweil apresenta a seguinte visão sobre a pesquisa de Inteligência Artificial:

> É nosso destino como pesquisadores de Inteligência Artificial nunca alcançar a cenoura pendurada à nossa frente. A inteligência artificial é inerentemente definida como a busca de problemas difíceis da ciência dos computadores que ainda não foram resolvidos (KURZWEIL, 2000).

2.2 AGENTE INTELIGENTE

Um agente inteligente é uma entidade autônoma, capaz de tomar as próprias decisões para atingir seus objetivos (WOOLDRIDGE, M., 1999). Apesar da definição intuitiva ser simples, assim como no termo inteligência artificial não existe um consenso da comunidade sobre o que é um agente. Definições mais simples categorizam agente como algo que age, e agente inteligente como aquele que age buscando o melhor resultado possível (RUSSEL; NORVIG *et al.*, 2013), enquanto definições mais fechadas buscam contextualizar a entidade em um ambiente:

> Um agente inteligente é um sistema que age de maneira inteligente: o que faz é apropriado para as circunstâncias e seus objetivos, é flexível para mudar ambientes e mudar objetivos, aprende com a experiência e faz escolhas apropriadas, dadas as limitações perceptivas e a computação finita (POOLE *et al.*, 1998).

Tais definições são bastante amplas, e enquadram diversos tipos de programas computacionais. Apesar disso, agentes possuem características específicas, principal-

mente provenientes da capacidade de interagir uns com os outros. As capacidades e características dos agentes inteligentes podem ser divididas em cinco tópicos (LUGER, 2008):

1. **Agentes são autônomos ou semi-autônomos:** Os agentes são independentes, ou seja, cada agente é capaz de trabalhar em uma tarefa sem saber no que outros agentes estão trabalhando, ou sem saber como eles resolvem determinada tarefa. Além disso, eles podem tanto fazer algo efetivamente (agir) ou reportar seus resultados para outros agentes (se comunicar).

2. **Agentes possuem escopo localizado:** Cada agente é sensível ao ambiente, e normalmente não possui conhecimento sobre aquilo que todos os outros agentes estão realizando. Portanto o conhecimento de um agente é limitado às tarefas que ele deve realizar, sem conhecimento amplo sobre seus limites.

3. **Agentes são interativos:** Normalmente, agentes se agrupam em forma de sociedade, com o objetivo de colaborar para resolver um problema. E assim como na sociedade humana, o conhecimento, a responsabilidade, habilidades e outros recursos estão distribuídos entre os indivíduos.

4. **As sociedades dos agentes são estruturadas:** Na maioria das abordagens de solução de problema orientada a agentes, cada agente, mesmo possuindo seu próprio conjunto de habilidades e objetivos, se coordena com outros agentes para a resolução geral de problemas. Portanto, a solução final não é apenas coletiva, mas também cooperativa.

5. **O fenômeno da inteligência nesses ambientes é emergente:** A capacidade final da resolução de um problema por uma sociedade de agentes é maior do que a soma das capacidades individuais de trabalho. A inteligência é vista como um fenômeno residente e emergente de uma sociedade e não apenas uma propriedade de um agente individual.

A noção de agente inteligente ainda pode ser caracterizada como forte ou fraca (WOOLDRIDGE, M. J.; JENNINGS, 1995). A noção fraca de agente é utilizada para denominar hardware ou software que possui algumas características específicas, sendo elas autonomia, habilidade social, reatividade e pró-atividade. Já a noção forte de agente se refere a um sistema que, além das características citadas anteriormente, ou foi concebida ou foi implementada utilizando conceitos que normalmente se aplicam a humanos. O modelo explorado neste trabalho segue a noção forte de agente, pois se baseia em conceitos da psicologia e da filosofia para resolver um problema prático de agentes.

Com base na leitura das definições apresentadas anteriormente, para este trabalho vamos definir agente formalmente conforme apresentado na Definição 1, de maneira que facilite a manipulação e formalização do modelo proposto.

Definição 1. Um agente é uma tripla $Ag = \langle K, P, \gamma \rangle$, onde:

- K é uma base de conhecimentos, tal que $K = K_i \cup K_p$, onde K_i é o conjunto de conhecimentos inicias do agente e K_p os conhecimentos adquiridos através das percepções. K_i é iniciado com valores arbitrários de acordo com a necessidade do agente e K_p é iniciado vazio. Uma base de conhecimentos é uma estrutura que representa fatos a respeito do mundo e apresenta formas de raciocinar a respeito desses fatos para deduzir novos conhecimentos (HAYES-ROTH, F. *et al.*, 1983);
- P é o conjunto de planos do agente, sendo um plano definido como $plano = (\Psi, A, \Omega)$, onde Ψ é o conjunto união formado pelas pré-condições das ações que compõem o plano, A o conjunto de ações que compõe o plano e Ω o conjunto união formado pelas pós-condições das ações que compõem o plano. Por sua vez, uma ação é definida como $acao = (\psi, n, \omega)$, sendo ψ um conjunto de pré-condições, n um nome para a ação e ω um conjunto de pós-condições; e
- γ é a função de percepção, definida como $\gamma(p, K) \rightarrow P_i$, onde p é o conjunto de percepções recebidas, K a base de conhecimentos de Ag e P_i o retorno da função, que é um subconjunto próprio do conjunto P de planos do agente.

A partir dessa definição, podemos construir o conceito de contexto, que será amplamente utilizado na formalização do modelo de revisão de percepções. O contexto de um agente é o conjunto de todos os símbolos compreendidos pelo agente:

Definição 2. O contexto c de um agente Ag é o domínio de sua função γ.

2.3 PERCEPÇÃO

Existem diversas definições para o termo “percepção”. Podemos entender percepção como um conjunto de sensações que, através da maneira subjetiva que um dado agente o interpreta, representa determinadas entidades do ambiente (GIBSON, 1950). Ou seja, a percepção não é simplesmente a representação direta das entidades reais que existem no mundo, mas um processo complexo que varia para cada indivíduo.

Percepções podem ser divididas nos níveis baixo e alto (CHALMERS *et al.*, 1992). A percepção de baixo nível ocorre através de meios físicos, os órgãos ópticos para os humanos ou os sensores para os agentes. A percepção de alto nível trabalha com uma visão mais geral da informação, extraindo conceitos dos dados brutos,

podendo envolver diversas faculdades como o reconhecimento de objetos e o relacionamento de entidades. Nos trabalhos de Inteligência Artificial, em geral, estamos interessados na percepção de alto nível, pois a percepção de baixo nível está mais relacionada a robótica (no caso de agentes que possuem hardware próprio) ou a simulação (no caso de agentes que possuem apenas software).

Ainda segundo Chalmers et. al. uma das principais características da percepção de alto nível é a extrema flexibilidade. Um mesmo objeto do ambiente pode ser percebido de diversas maneiras, de acordo com as características do observador. Para os autores, algumas das fontes da flexibilidade das percepções são a capacidade de serem influenciadas pelas crenças, objetivos e contexto externo. Além disso, percepções de um mesmo objeto podem ser radicalmente alteradas conforme o necessário.

Para o modelo que iremos propor, baseado na Definição 1, o conceito de percepção pode ser simplesmente definido como a entrada *p* da função de percepção γ de um agente. Vale destacar a diferença entre percepção e contexto, pois o contexto é constituído pelas percepções que fazem parte do domínio da função γ, ou seja, uma percepção é toda informação produzida pelo ambiente que o agente recebe, e contexto é o subconjunto das percepções que o agente reconhece.

Em ambientes dinâmicos há possivelmente centenas de percepções por segundo (HAYES-ROTH, B. *et al.*, 1992). Mas percepções não necessariamente precisam incluir representações corretas da realidade e podem variar de agente para agente (JANSSEN, 2005). Percepções incorretas podem ocorrer por conta de limitações da capacidade do agente de perceber determinados objetos ou por conta de obstrução física dos sensores, por exemplo (CHRISMAN *et al.*, 1991).

2.3.1 Refinamento

Como o volume de percepções de um agente pode ser muito grande, e as percepções custam tempo para serem processadas, o número de percepções que chegam ao ciclo de raciocínio do agente pode ser reduzido para diminuir seu custo computacional. Neste trabalho, esse processo será chamado de refinamento, definido da seguinte maneira:

Definição 3. Refinamento de percepções é uma função θ tal que, dado o conjunto de entradas de percepções *p*, reduz tais percepções para um subconjunto próprio ρ.

Existem diversas maneiras de realizar refinamento. Uma das mais clássicas é o uso de filtros de percepções, que limitam que percepções serão processados pelo agente baseado em diversos critérios, como posição, distância e a velocidade do objeto percebido (BORDEUX *et al.*, 1999). Outra maneira de restringir as percepções é a percepção ativa. Um observador pode ser categorizado como ativo se ele ativamente pode executar uma ação que altere as configurações geométricas de seus sensores,

com objetivo de melhorar a qualidade de sua observação (ALOIMONOS *et al.*, 1988). O processo realizado pela percepção ativa pode ser definido através da tupla: por quê, o que, quando, onde e como perceber (BAJCSY *et al.*, 2018). Desta maneira, o agente pode perceber apenas quando é necessário (por quê), escolhendo o que perceber (o que) e otimizar fisicamente a percepção (quando, onde e como).

2.3.2 Anomalias

Anomalias são as percepções consideradas inválidas, geradas por alguma falha no processo de percepção. O conjunto de anomalias de um agente é o conjunto de todas as percepções possíveis que não fazem parte de seu contexto. A seguinte definição será utilizada:

Definição 4. Uma percepção p de um agente Ag com contexto c, é uma anomalia caso $p \notin c$ de Ag.

Anomalias podem ser divididas em dois tipos: alucinações e ilusões. Nas seções seguintes, será descrito o que as alucinações e ilusões são para os estudos clássicos do problema da percepção (RUSSELL, B., 1912) (PRICE, 1933) e como elas podem ser representadas dentro do processo de percepção de agentes inteligentes. A base desse estudo foi o artigo “The Problem of Perception” (CRANE; FRENCH, 2017).

2.3.2.1 Ilusão

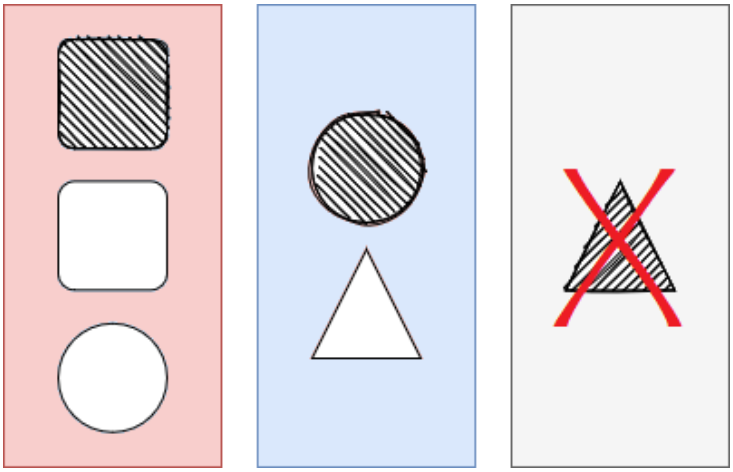
Uma ilusão, na definição clássica onde o objeto de estudo são os seres humanos, é qualquer situação perceptiva na qual um objeto físico é realmente percebido, mas ele aparenta ser outra coisa que ele não é (SMITH, 2002). O Exemplo 2.3.2 ilustra o que é uma ilusão.

Exemplo 2.3.1. Suponha um robô que possui a tarefa de empacotar diferentes itens do depósito de uma loja. Os itens passam por uma esteira, e o agente usa dois sensores para perceber o que os itens são. O primeiro sensor é uma câmera ligada ao topo do corpo do robô, usado para definir a forma do item. O segundo sensor é um detector de textura na lateral das esteiras. Assim, as informações advindas dos sensores formam predicados da forma `forma(textura)`, que descreve um item. Os itens podem ter a forma de um círculo, um quadrado ou um triângulo, e sua textura pode ser lisa ou listrada. O agente possui duas cores de papéis para empacotar, vermelho e azul. O papel vermelho é usado para quadrados (de ambas as texturas) e círculos lisos. O papel azul é usado para círculos listrados e triângulos lisos. A loja não vende triângulos listrados, então não há nenhum item desse tipo no depósito. Podemos descrever esse comportamento com as seguintes regras:

```
papel(vermelho) :- quadrado(_) OR circulo(liso)
```

```
papel(azul) :- circulo(listrado) OR triangulo(liso)
```

Figura 1 – Funcionamento do agente empacotador.



Fonte: Autor.

Exemplo 2.3.2. Agora vamos estender o exemplo 2.3.1 supondo que o sensor tátil não está funcionando corretamente, e ele irá sentir objetos lisos como se fossem ondulados. Para nosso exemplo, vamos considerar que o primeiro item será um triângulo liso. Nenhum erro ocorre quando a câmera percebe o triângulo, mas o sensor tátil indica que ele é ondulado. Nesse caso, teremos uma ilusão, pois triângulo é um objeto válido, mas ele foi percebido com uma propriedade inválida, que não existe no contexto do agente. Não há planos para quando o agente detecta esse tipo de erro, então ele pode executar um plano padrão para casos de erro, ou simplesmente não fazer nada.

Esse tipo de anomalia demonstrado no Exemplo 2.3.2 será chamado de ilusão classe 1, onde o corpo do predicado, ou o objeto da percepção, é válido, mas possui um argumento ou uma característica inválida.

Definição 5. Uma ilusão classe 1 é uma percepção do tipo `objeto(caracteristica)` ou equivalente, onde `objeto` é um elemento do contexto do agente e `característica` não é.

Exemplo 2.3.3. Se considerarmos que as percepções do agente possuem formas erradas por conta de um defeito na câmera ou o software de reconhecimento de padrões que atua sobre ela, um objeto como um círculo liso pode ser reconhecido como um estrela listrada. Estrela não é um objeto válido, mas listrado é.

Isso será chamado de ilusão classe 2, definido de maneira similar a ilusão classe 1.

Definição 6. Uma ilusão classe 2 é uma percepção do tipo `objeto(caracteristica)` ou equivalente, onde `objeto` não é um elemento do contexto do agente e `característica` é.

Portanto, podemos simplesmente definir ilusão da seguinte forma:

Definição 7. Uma ilusão é uma percepção do tipo `objeto(caracteristica)` ou equivalente, que se caracteriza como uma ilusão classe 1 ou uma ilusão classe 2.

2.3.2.2 Alucinação

O segundo tipo de anomalia é a alucinação. Uma alucinação é uma percepção recebida que poderia ser completamente válida, mas que na realidade não existe no ambiente.

Exemplo 2.3.4. Retornando ao exemplo 2.3.2 do agente responsável por empacotar os itens, mas que agora apresenta também o comportamento defeituoso do exemplo 2.3.3. Uma percepção formalmente correta, do tipo `objeto(caracteristica)`, por conta dos erros que os sensores possuem, pode resultar na percepção `estrela(ondulada)`. Essa percepção poderia ser processada pelo agente, entretanto ela não faz parte de seu contexto e resultaria na execução de um plano padrão para erros ou na inação do agente.

Assim, vamos definir alucinação como um tipo específico de ilusão classe 1 e classe 2, podendo acarretar os mais diversos tipos de erros dentro do raciocínio do agente, ou gerando problemas caso seja ignorada.

Definição 8. Uma alucinação é uma percepção do tipo `objeto(caracteristica)` ou equivalente, onde nem `objeto` nem `característica` são elementos do contexto do agente.

2.4 PLANEJAMENTO AUTOMATIZADO

Planejamento automatizado é um dos problemas fundamentais da Inteligência Artificial. As motivações para usar o planejamento automatizado são a capacidade de utilizar recursos de planejamento acessíveis e eficientes e reproduzir uma parte do processo cognitivo humano com um componente totalmente integrado de comportamento deliberativo (GHALLAB *et al.*, 2004). A maneira clássica de realizar planejamento automatizado é considerar esse um problema de dedução lógica, onde dado um estado inicial, ações que aferam esse estado e um conjunto de estados de objetivo, é necessário encontrar a sequência de ações que faziam com que o ambiente saísse de um estado inicial para um estado de objetivo (MADANI *et al.*, 2003).

Uma forma alternativa de tratar o problema de planejamento automatizado é utilizando planejamento probabilístico (KUSHMERICK *et al.*, 1995). Essa abordagem pode ser necessária por conta do fato de que o agente provavelmente não tem conhecimento completo do mundo ao seu redor. Outra saída para o problema do planejamento automatizado são os processos de decisão de Markov (CASSANDRA, 1998), (BOUTILIER *et al.*, 2011), (LITTMAN, 2009).

Na Definição 9 é apresentada a noção abstrata de planejamento automático, descrita como um modelo conceitual simples que contém os elementos principais do problema, tendo sido originalmente apresentada por Ghallab et. al. (GHALLAB *et al.*, 2004).

Definição 9. Um modelo conceitual de planejamento automatizado é descrito como a interação entre os seguintes três componentes:

- Um sistema de transição de estados Σ, especificado por uma função de transição de estados Γ, de acordo com os eventos e ações que ele recebe.
- Um *controlador*, que dado uma entrada de estados *s* do sistema, fornece como saída uma ação de acordo com algum plano.
- Um *planejador*, que dado uma entrada de uma descrição de sistema Z, uma situação inicial e alguns objetivos, sintetiza um plano para o controlador a fim de alcançar o objetivo.

Um sistema de transição de estados Σ é uma quádrupla $\Sigma = \langle S, A, E, \Gamma \rangle$, onde:

- $S = \{s_1, s_2, ..., s_n\}$ é um conjunto finito ou recursivamente enumerável de estados;
- $A = \{a_1, a_2, ..., a_n\}$ é um conjunto finito ou recursivamente enumerável de ações;
- $E = \{e_1, e_2, ..., e_n\}$ é um conjunto finito ou recursivamente enumerável de eventos; e
- $\Gamma : S \times A \times E \rightarrow 2^S$ é uma função de transição de estados.

2.5 RESUMO

Existem diversas definições de agente, mas para este trabalho será considerado que um agente inteligente é uma entidade, inserida em um ambiente, que possui autonomia para tomar suas decisões. Essa entidade possui um conjunto de conhecimentos a respeito do ambiente, e pode atualizar esses conhecimentos através da percepção, o processo de utilizar seus sensores para reconhecer o mundo ao seu redor. O funcionamento básico de um agente acontece através da entrada de percepções novas, o processamento delas para atualizar a base de conhecimentos e a escolher de quais

ações tomar. Como o agente pode receber muitas percepções de uma vez, pode ser necessário criar medidas para reduzir esse volume de acordo com certos parâmetros – chamamos isso de refinamento.

Por fim, as percepções que um agente recebe podem ser anomalias, i. e., inválidas ou corrompidas. Separamos essas anomalias entre ilusões e alucinações, de acordo com suas características semânticas. Neste trabalho, será proposto um modelo que recebe as percepções, refina, detecta e classifica possíveis anomalias e cria novos planos para lidar com essas percepções inválidas através de um processo de planejamento automatizado.

Esse processo de receber percepções, categorizá-las como percepções válidas ou anomalias e gerar novos artefatos de valor para o agente é o que chamamos neste trabalho de revisão de percepções. No Capítulo 3 alguns trabalhos que propõem processos similares a revisão de percepções serão apresentados.

3 TRABALHOS RELACIONADOS

Existem diversas abordagens para otimizar as percepções recebidas por um agente, isto é, garantir que todas as informações coletadas pelos sensores sejam utilizadas da melhor maneira possível. Diversos artigos sobre o assunto, com diferentes abordagens, são publicados todos os anos. Para definir os trabalhos relacionados apresentados nessa seção, foram utilizados diversos termos de busca, uma vez que os termos ilusão e alucinação podem não necessariamente se aplicarem às percepções da mesma maneira definida neste trabalho.

Nos mecanismos de busca Google Scholar e Scopus foram realizadas pesquisas que associam agentes inteligentes a percepções inválidas, anomalias ou ilusões e alucinações, além de termos auxiliares como aprendizado e otimização. As buscas aconteceram ao longo do desenvolvimento do trabalho, e diversas *strings* de busca foram utilizadas e combinadas. Conforme encontraram-se os artigos, suas introduções foram verificadas para averiguar se o conteúdo realmente estava relacionado ao processo de revisão de percepções. Depois disso foi realizada uma leitura inicial dos trabalhos selecionados, e os quatro mais adequados foram escolhidos para um estudo mais profundo.

Os artigos apresentados nas Seções 3.1 e 3.2 implementam modelos para tratar de percepções em ambientes onde é possível receber percepções que não são totalmente confiáveis, semelhante ao conceito de anomalia definido no Capítulo 2. Os artigos das Seções 3.3 e 3.4 são trabalhos relacionados a outros campos de estudo de percepção, mas que precisam resolver problemas relacionados a percepções imperfeitas. A maneira como esses artigos se conectam ao modelo proposto no trabalho atual está descrita em mais detalhes nas seções seguintes.

3.1 SCALABLE PERCEPTION FOR BDI-AGENTS EMBODIED IN VIRTUAL ENVIRONMENTS (OIJEN; DIGNUM, 2011)

Ambientes virtuais como jogos, simulações e treinamentos exigem cada vez mais complexidade dos agentes com os quais os participantes interagem. A arquitetura BDI (*belief-desire-intension*) provê a complexidade necessária para que os agentes virtuais desempenhem as tarefas avançadas necessárias. Porém, agentes BDI tradicionais possuem uma interface direta com o ambiente, enquanto os agentes dos jogos normalmente possuem um conjunto de sensores limitados para realizar suas percepções. O problema é que agentes BDI não possuem um mecanismo padrão para controlar seus sensores, decidindo quais percepções receber. Dessa maneira, o agente pode facilmente ficar sobrecarregado de informação. Para resolver esse problema, os autores desse trabalho criaram um *framework* que fornece habilidades sensoriais e e atenção perceptiva para agentes BDI incorporados em um ambiente

virtual, funcionando como um *middleware* que atua entre o modelo cognitivo do agente e o ambiente.

Nesse *framework* (representado na figura 2) toda informação do ambiente é representada pelo modelo de informação chamado *Environment Object Model* (Modelo de Objeto de Ambiente) ou EOM. Objetos são definidos por classes e características, e existe uma hierarquia de objetos para agregar semântica. O *middleware* é dividido entre a interface física e a interface cognitiva. A função da interface física é interagir com o ambiente para formar símbolos. Para tal, o processador sensorial primeiro recebe uma lista de possíveis percepções, que respeita filtros predeterminados implementados nos sensores. Após isso, esse processador determina se o agente está interessado nas informações recebidas, através do *Interest Subscription Manager* (gerenciador de inscrição de interesse). Estes símbolos são passados para o agente na foma de percepções, que alteram os objetivos do agente. Os novos objetivos, por sua vez, são repassados para o sistema de atenção, que atualiza os interesses do agente no gerenciador de inscrição de interesse. A comunicação entre a interface física e a interface cognitiva é realizada através de mensagens, e há interfaces para que o *middleware* possa se comunicar com o ambiente e com o agente, de maneira a mantê-lo independente de domínio.

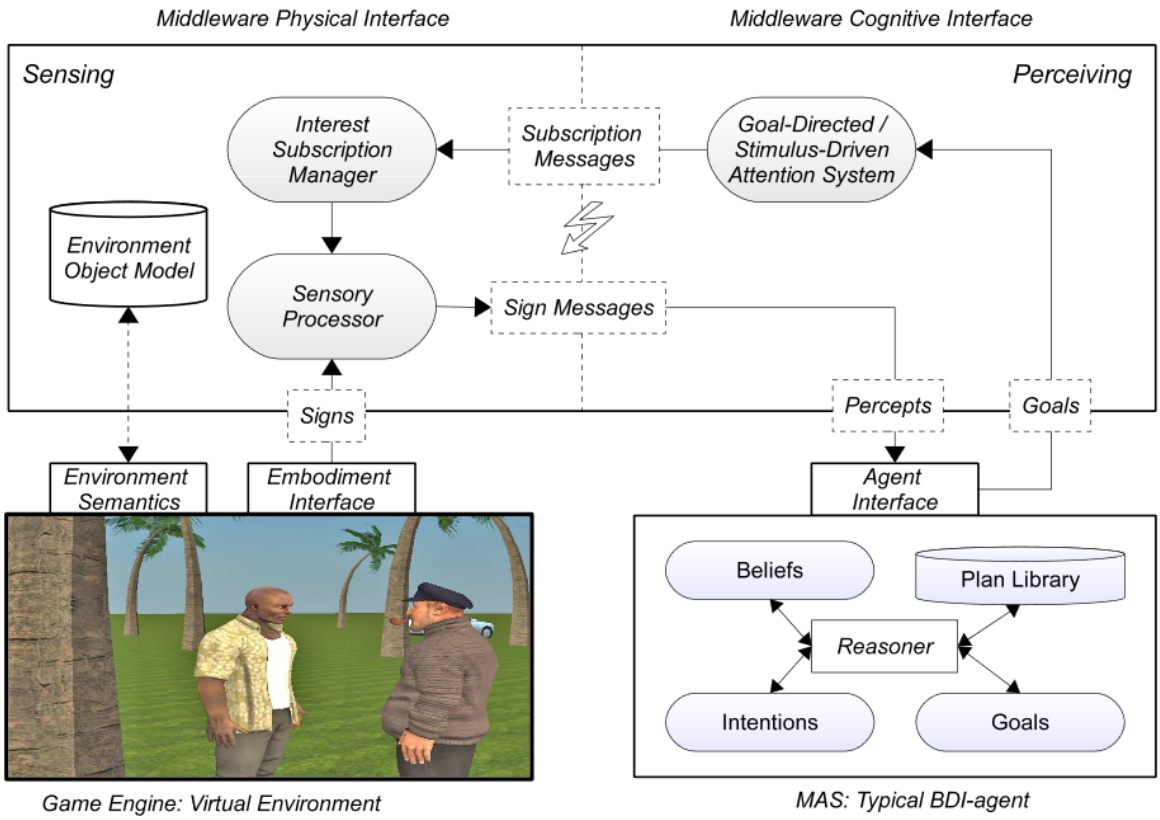
Para avaliar o modelo proposto, foram realizados dois experimentos. O primeiro consistiu no uso de sensores físicos em um teste de estresse, tanto em um sistema que utilizava o framework quando em um que não o utilizava. As percepções recebidas pelos agentes continham cinco atributos cada, sendo que a quantidade de entidades percebidas aumentava gradativamente. Cinco baterias de testes foram conduzidas para cada agente. O segundo experimento consistiu na implementação de um sistema multiagente (SMA) de agentes BDI baseado em Java e utilizando uma engine Prolog para realizar o raciocínio, e os mesmos testes do primeiro experimento foram realizados.

Esse modelo, que é capaz de decidir que tipo de percepções o agente deseja perceber de acordo com seus interesses, se assemelha ao HAIL porque muda dinamicamente ao longo do tempo, conforme as percepções são processadas. Além disso, ele possui um elemento de percepção ativa, pois o conjunto de percepções recebidas de maneira direta e massiva do ambiente é filtrado pelo processador de sensores.

3.2 PMK — A KNOWLEDGE PROCESSING FRAMEWORK FOR AUTONOMOUS ROBOTICS PERCEPTION AND MANIPULATION (DIAB *et al.*, 2019)

As tarefas executadas por robôs vêm se tornando cada vez mais complexas. Para realizar essas tarefas, os robôs precisam passar por uma etapa de planejamento, na qual decidem quais ações tomar baseados no estado atual do ambiente ao seu redor. Alguns dos mecanismos clássicos de planejamento utilizam a Linguagem de

Figura 2 – *Framework* de percepção baseado em EOM.



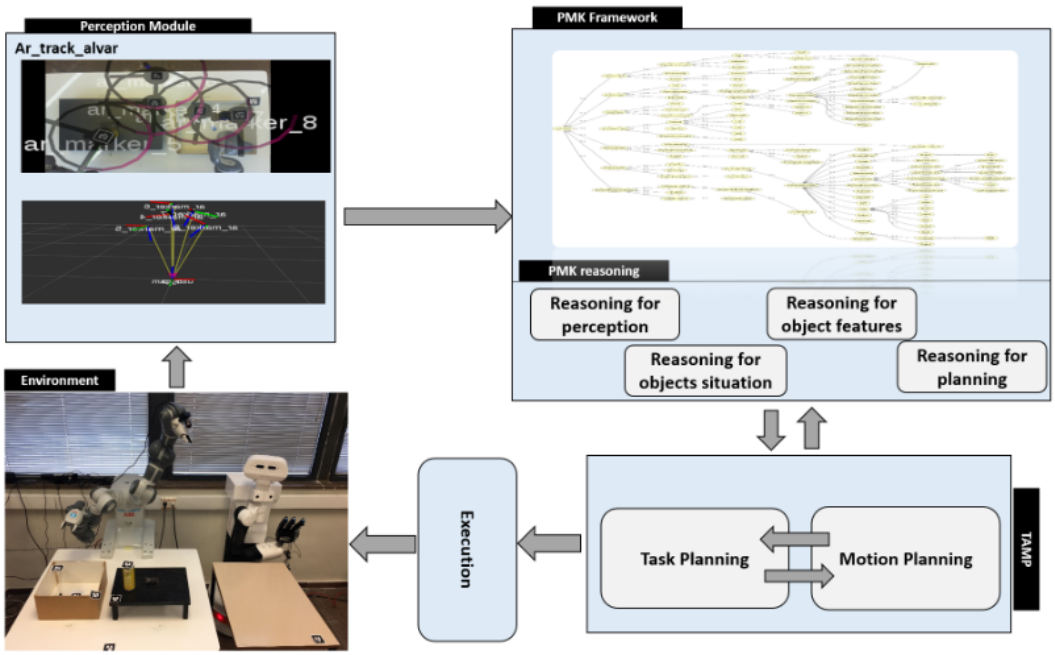
Fonte: Scalable perception for BDI-agents embodied in virtual environments (OIJEN; DIGNUM, 2011).

Definição de Domínio de Planejamento (*Planning Domain Definition Language* ou PDDL) para descrever o ambiente no qual o agente está inserido. O problema é que essa abordagem assume um mundo fechado, i.e., que todos os fatos sobre o mundo são conhecidos, caso contrário o planejador pode falhar. Com essa limitação, um robô não é capaz de começar uma tarefa a não ser que todos os objetos do ambiente tenham sido reconhecidos e as ações que ele deve executar tenham sido definidas. Em outras palavras, a existência de alucinações e ilusões limita o funcionamento de tais sistemas.

Para resolver esse problema em situações onde o robô precisa realizar tarefas complexas de manipulação, foram criadas abordagens de planejamento baseadas no conhecimento, que utilizam reconhecimento semântico do cenário, conhecimento a respeito do comportamento físico de objetos e raciocínio sobre as possíveis ações de manipulação.

O trabalho de Diab et al. propõem um *framework* de representação de conhecimento baseado em ontologias (uma especificação formal de conhecimento) chamado PMK (*Perception and Manipulation Knowledge*), apresentado na figura 3. Esse modelo é genérico, para que possa ser utilizado em diversos domínios, e incorporado

Figura 3 – *Framework* PMK.



Fonte: PMK - a knowledge processing framework for autonomous robotics perception and manipulation (DIAB *et al.*, 2019).

com outras ontologias. O PMK permite associar dados de percepção de baixo nível (proveniente dos sensores, na camada física do sistema) com conhecimento de alto nível (camada de raciocínio do agente). Uma dos principais contribuições do artigo é criar um *framework* que funcione como uma caixa preta para um planejador qualquer: o PMK é capaz raciocinar sobre os recursos do robô, suas restrições de ação, a viabilidade de ação e os comportamentos de manipulação. Para isso, o modelo utiliza análise situacional, avaliando a situação a situação dos objetos no ambiente com base em posicionamento espacial, acessibilidade do robô aos objetos, potencial área na qual o objeto será colocando entre outros.

A abordagem que é proposta pelo PMK oposta ao HAIL, pois tenta mapear todas as percepções possíveis em baixo nível a priori. Dessa forma, mesmo que o agente não tenha sido projetado para tratar determinadas percepções, ele é capaz de entender o significado semântico delas através do *framework*. Em outras palavras, a ideia do PMK é criar um mapa extenso para que nenhuma percepção recebida pelo agente seja uma anomalia.

3.3 COMBINING PERCEPTION AND KNOWLEDGE PROCESSING FOR EVERYDAY MANIPULATION (PANGERCIC *et al.*, 2010)

Robôs autônomos implementados para realizar tarefas de manipulação de objetos do dia a dia precisam tomar diversas decisões que requerem a combinação

de percepção e processamento de conhecimento. Esse artigo de Panger et al. apresenta um sistema de programação lógica chamado K-CoPMan (*Knowledge enabled Cognitive Perception for Manipulation*, ou Percepção Cognitiva Ativada pelo Conhecimento para Manipulação). Esse modelo é capaz de testar e satisfazer pré-condições de conhecimento para manipulações do dia a dia. Para isso, ele fornece ao agente o conhecimento simbólico abstrato sobre as cenas percebidas, usa conhecimento simbólico abstrato para realizar tarefas de percepção e responde a novos tipos de consultas que exigem a combinação de percepção e processamento de conhecimento.

Um dos principais mecanismos do K-CoPMan é o componente de percepção passiva. Para se tornar consciente do ambiente, o agente que utiliza tal sistema pode escanear a cena em busca de áreas de interesse, como mesas ou cadeiras, utilizar os sensores para detectar objetos. Cada objeto recebe um identificador único, para então ser guardado na base de conhecimentos, juntamente com o contexto do momento em que a percepção foi realizada. O identificador é utilizado para que mais tarde seja possível examinar mais a fundo objeto, e possivelmente classificá-lo ou categorizá-lo. Portanto, o K-CoPMan permite que agentes inteligentes estejam conscientes do ambiente ao seu redor fazendo uma varredura completa do ambiente, uma vez que utiliza tanto percepção ativa quanto passiva, e guardando as anomalias detectadas para que possam ser tratadas mais tarde por um módulo próprio (o servidor de percepção).

A abordagem desse trabalho para evitar os efeitos de percepções inválidas é similar ao PMK, mas ao invés de utilizar uma base de conhecimentos prévia para evitar que alguma percepção não possa ser tratada pelo agente, o próprio sistema cria sua base através da varredura do ambiente pela percepção passiva.

3.4 UNDERSTANDING HUMAN INTENTION BY CONNECTING PERCEPTION AND ACTION LEARNING IN ARTIFICIAL AGENTS (KIM *et al.*, 2017)

Para desenvolver agentes capazes de realizar comportamentos complexos similares aos de seres humanos, é primeiro preciso entender como os seres humanos aprendem a perceber, pensar e agir em um mundo dinâmico. Diversos campos da inteligência artificial buscam replicar esses comportamentos, além de outros como a emoção e a cooperação. Essas habilidades parecem ser intrínsecas aos seres humanos, e tornam nossas relações mútuas únicas. Em particular, a capacidade de entender a intenção dos outros tem sido considerada a base da comunicação entre humanos. Nesse artigo, Kim, Yu e Lee propõem um modelo, chamado OA-SMTRNN (*Object Augmented Supervised Multiple Timescale Recurrent Neural Network*), para entender a intenção do usuário e responder ativamente da maneira mais adequada, através do uso de redes neurais. Para implementar o reconhecimento de intenção, são focados dois processos cognitivos, a percepção da disponibilidade de objetos e a previsão da ação humana.

Nos experimentos realizados pelos autores, diversos objetos precisaram ser percebidos pelo agente. Entretanto, alguns objetos poderiam estar sobrepostos, conforme demonstra a Figura 4. Nesses casos, as percepções recebidas pelo agente poderiam estar incorretas. Para resolver este problema, o módulo responsável pelas ações foi implementado com a capacidade de relacionar a ação e os objetos. No artigo, é exemplificada a relação entre “encher um copo d’água” e “fazer um café mocha”. Ou seja, o modelo, que foi previamente treinado, se mostrou capaz de associar a intenções como “beber leite” a determinadas ações (segurar um objeto, leva algo para a boca) para inferir que determinada anomalia (uma caixa de leite sobreposta por uma caneca) era uma caixa de leite.

Figura 4 – Exemplo de sobreposição de imagens na percepção.

Fonte: Understanding human intention by connecting perception and action learning in artificial agents (KIM *et al.*, 2017).

Enquanto no modelo de revisão de percepções apresentado buscamos resolver o problema de percepção de anomalias através de uma abordagem simbólica, que classifica, trata e aprende com ilusões e alucinações, o modelo OA-SMTRNN utiliza uma abordagem conexionista, com o uso de redes neurais, para conseguir extrair semântica de percepções que seriam inicialmente inválidas para o agente. Além disso, na conclusão do trabalho é mencionado que “implementar aprendizagem de percepção-ação conectada pode desempenhar um papel importante no desenvolvimento de agentes artificiais que podem inferir a intenção humana e interagir melhor”. No HAIL, essa integração é realizada uma vez que as anomalias percebidas são utilizadas pelos módulos de planejamento automatizado para criar novos planos.

3.5 DISCUSSÃO

Os quatro trabalhos que foram selecionados possuem diversas similaridades e diferenças. Para visualizar isso melhor, eles foram separados de acordo com as seguintes categorias, para então comporem o Quadro 1:

1. Apresenta ou não um *framework* genérico que pode ser aplicado em qualquer agente, independente de arquitetura;
2. A abordagem que o trabalho utiliza para tratar percepções inválidas: limitando as percepções, de maneira que o agente realize a percepção apenas sobre as entidades que ele está pronto para tratar; definindo o ambiente, através de alguma forma de representação de conhecimento que descreve o mundo do agente para que ele possa tratar todas as percepções recebidas; ou tratar as anomalias recebidas através de algum sistema que processa as percepções inválidas recebidas;
3. Tipos de experimentos realizados: se foram feitas simulações de software ou se foi construído um agente físico para testar o modelo;
4. Tipos de percepções que o agente recebia: completas, no caso de ambientes simulados, ou físicas, no caso de agentes físicos;
5. Forma de avaliação que foi utilizada para mensurar a eficácia do método proposto;
6. Paradigma do agente (simbólico ou conexionista);
7. Se o modelo apresenta ou não alguma ferramenta de aprendizado que utiliza ilusões e alucinações para gerar novos planos ou conhecimento.

Através dessa classificação, é possível destacar em quais pontos nosso modelo se assemelha e diverge dos trabalhos relacionados que foram selecionados. O principal referencial do HAIL está no processo de revisão de percepções simbólico e na capacidade de permitir que qualquer arquitetura possua um processo de aprendizagem através do planejamento automatizado. Isso se reflete principalmente na forma de avaliação (tempo de processamento, como no trabalho 3.1, e também pela taxa de aprendizado do modelo) e no tipo de percepção utilizada nos experimentos (completas, pois o agente recebe a informação diretamente da simulação, mas sem semântica atrelada e sem contexto de aplicação). Além disso, três dos quatro trabalhos selecionados possuem um ambiente físico no mundo real: os trabalhos 3.2 e 3.3 na forma de implementação física e o trabalho 3.4 na forma de percepção de imagens que foram extraídas de um ambiente real. Isso é um indicativo de que trabalhos futuros podem implementar nosso modelo fisicamente, em robôs, por exemplo.

Quadro 1 – Trabalhos relacionados categorizados.

Trabalho	**Framework Genérico**	**Abordagem**	**Experimentos**	**Tipos de Percepções**	**Forma de Avaliação**	**Paradigma**	**Ferramenta de Aprendizado**
OIJEN; DIGNUM, 2011	Sim (BDI)	Limitar percepções	Simulação de SMA com ambiente gráfico	Completas (semântica no ambiente)	Tempo de processamento	Simbólico	Não
DIAB et al., 2019	Sim	Definir o ambiente	Implementação física	Física (câmeras)	Corretude da tarefa executada	Simbólico	Não
Pangercic et. al., 2010	Não	Definir o ambiente	Implementação física	Física (câmeras)	Corretude da tarefa executada	Simbólico	Não
KIM; YU; LEE, 2017	Não	Tratar anomalias	Implementação de modelo computacional	Dataset de imagens	Corretude da predição	Conexionista	Sim
HAIL (trabalho atual)	**Sim**	**Tratar anomalias**	**Simulação de agente único**	**Completas (aleatórias e independentes de contexto)**	**Tempo de processamento e capacidade de aprendizado**	**Simbólico**	**Sim**

Fonte: Autor.

4 MODELO DE REVISÃO DE PERCEPÇÕES

Neste capítulo, será descrito e formalizado o modelo proposto nesse trabalho. Ele foi inspirado pelos conceitos de ilusão e alucinação apresentados no Capítulo 2, que o nomeiam – o nome HAIL vem da junção das palavras *hallucination* e *illusion*, alucinação e ilusão em inglês, respectivamente. Seu objetivo é identificar anomalias nas percepções recebidas por um agente qualquer e torná-las informação úteis na forma de novos planos.

4.1 VISÃO GERAL DO MODELO

O modelo HAIL foi desenvolvido para que seja possível adicioná-lo a qualquer agente, independente de arquitetura cognitiva, como um componente que conecta as percepções vindas do ambiente ao agente. O HAIL pode ser separado em dois módulos, como mostra a Figura 5. De maneira geral, o funcionamento e a comunicação desses módulos se dá da seguinte maneira:

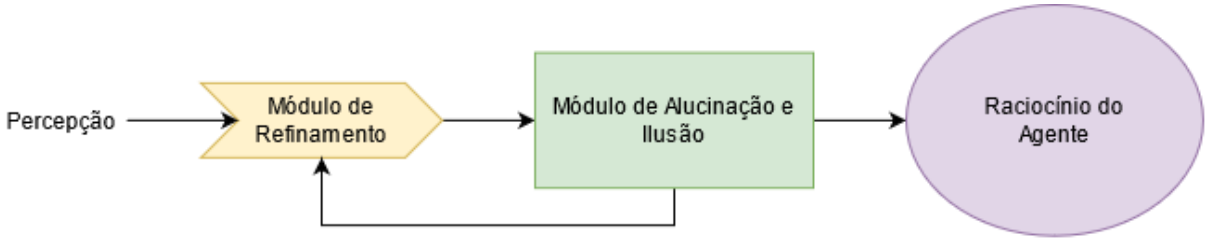
1. As percepções recebidas pelo modelo são refinadas pelo módulo de refinamento;
2. As percepções refinadas passam pelo módulo de alucinação e ilusão onde são categorizadas entre: percepções válidas, alucinações, ilusões classe 1 e ilusões classe 2;
3. As percepções válidas são encaminhadas para o raciocínio do agente, enquanto as anomalias continuam no módulo de alucinação e ilusão armazenadas em estruturas chamadas de bloco avaliador;
4. Quando os requisitos estabelecidos pelo bloco avaliador são cumpridos, as anomalias são selecionadas para passarem pelo processo de planejamento automatizado, alimentando o agente com novos planos.

Para ajudar a compreender o funcionamento do HAIL integrado ao raciocínio de um agente, nós usaremos uma versão estendida do Exemplo 2.3.1, com algumas adições para podermos demonstrar passo a passo como funciona o modelo.

Exemplo 4.1.1. Partindo do exemplo 2.3.1, vamos supor que agora a loja vende estrelas lisas e listradas, mas elas não devem ser empacotadas. Além disso, o mesmo robô responsável por empacotar os itens que passavam por uma esteira é responsável por empacotar itens de três diferentes esteiras. As percepções são as mesmas que antes, mas agora ele é capaz de perceber os itens nas três esteiras, através de novos sensores táteis e de uma câmera que capta uma imagem aberta o suficiente para isso. As percepções continuam sendo do tipo `forma(textura)`.

O Exemplo 4.1.1 será estendido em casos específicos nas seções seguintes.

Figura 5 – Visão geral do modelo HAIL.



Fonte: Autor.

4.2 MÓDULO DE REFINAMENTO

O módulo de refinamento funciona como uma primeiro filtro para que percepções indesejadas pelo agente não cheguem até seu ciclo de raciocínio. O processo de refinamento é descrito pela Definição 3.

O processo de refinamento não é obrigatório. Caso não seja de interesse de uma determina implementação do HAIL refinar suas percepções, basta que a função do módulo de refinamento seja a função identidade $f(x) = x$, possuindo assim $\rho = p$.

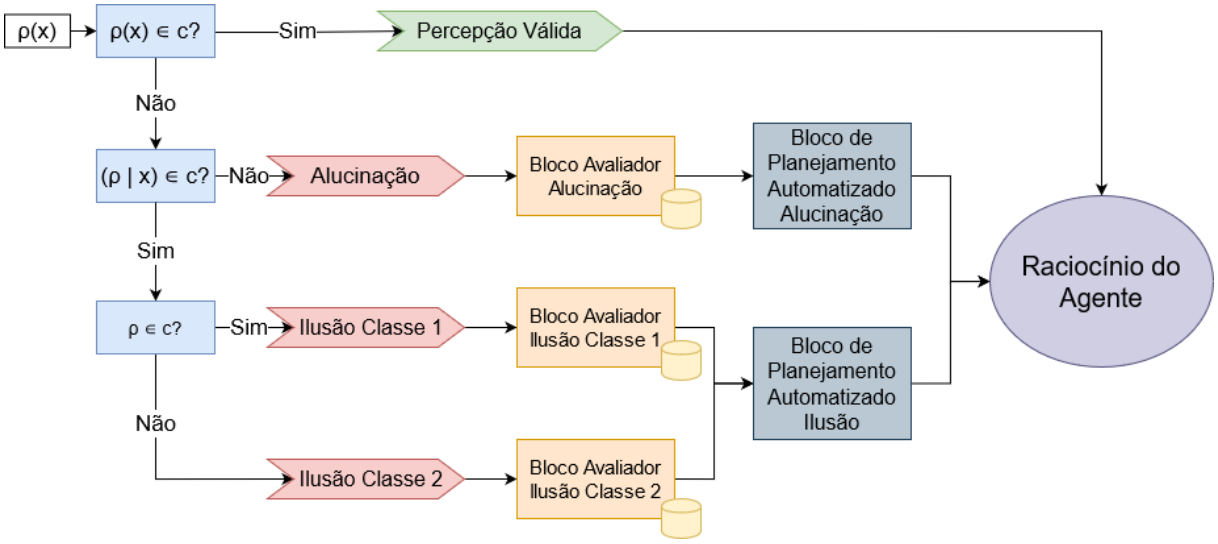
Exemplo 4.2.1. Continuando o exemplo 4.1.1, as estrelas não fazem parte da área de atuação desse robô, e portanto para otimizarmos o processo de empacotamento é possível utilizar o módulo de refinamento para reduzir a informação desnecessárias enviadas para o raciocínio do agente. Nesse exemplo, o HAIL pode ser implementado com uma função θ que realiza uma filtragem simbólica, removendo as percepções que não fazem parte do contexto. O conjunto de percepções p passa pelo processo de filtragem e retorna ρ. Neste exemplo, sendo s o conjunto de percepções possíveis envolvendo estrelas, a função de refinamento possui um comportamento tal que o conjunto p sob a operação θ retorna $\rho = p \cap \overline{s}$, ou seja, o agente possui um filtro que remove as percepções que envolvem estrelas.

Por exemplo, vamos supor que o agente recebe o conjunto p_i de percepções, composto por {*circulo*(*listrado*), *triangulo*(*liso*), *estrela*(*amarela*)}. A operação θ vai remover de p_i os elementos contidos no conjunto s de possíveis percepções envolvendo estrelas, conforme foi descrito anteriormente. Portanto, a saída do bloco de percepções será ρ = {*circulo*(*listrado*), *triangulo*(*liso*)}.

4.3 MÓDULO DE ALUCINAÇÃO E ILUSÃO

A Figura 6 apresenta um diagrama do funcionamento do módulo de alucinação e ilusão. Sua função é receber todas as percepções que passaram pelo processo de refinamento, e detectar quais delas são anomalias. Para isso, primeiro cada percepção $\rho(x)$ da entrada ρ (o conjunto de percepções filtradas) é dirigida para o decisor 1. O primeiro decisor responde a pergunta: “A percepção recebida faz parte do contexto

Figura 6 – Módulo de alucinação e ilusão.



Fonte: Autor.

do agente?". Caso a resposta for sim, consideramos a percepção como válida, e ela é enviada para o raciocínio do agente. Caso a resposta seja não, consideramos $\rho(x)$ uma anomalia, e enviamos ela para o segundo decisor, que responde a pergunta: "O corpo ou o argumento do predicado $\rho(x)$ faz parte do contexto do agente?". Caso a resposta seja não, concluímos que a anomalia é uma alucinação. Caso contrário, ela é considerada uma ilusão e é enviada para o terceiro decisor. O terceiro decisor responde a pergunta: "O corpo do predicado $\rho(x)$ faz parte do contexto do agente?". Caso a resposta for sim, a ilusão é considerada uma ilusão classe 1, caso contrário, é considerado uma ilusão classe 2. Essa cadeia de decisores pode ser representada pelo Algoritmo 1.

Algoritmo 1 Funcionamento dos decisores do módulo de alucinação e ilusão.

Entrada: contexto c do agente, percepção $\rho(x)$

```
1  início
2      se ρ(x) está em c então
3          ρ(x) é uma percepção válida
4      senão se nem ρ nem x estão em c então
5          ρ(x) é uma alucinação
6      senão se ρ está em c então
7          ρ(x) é uma ilusão classe 1
8      senão
9          ρ(x) é uma ilusão classe 2
10     fim
11 fim
```

Exemplo 4.3.1. Para entender como as percepções são tratadas pelos decisores, vamos supor algumas entradas possíveis para o agente de nossos exemplos. Vamos

analisar dois casos, uma percepção válida e uma anomalia:

- `quadrado(riscado)` – Essa é uma percepção completamente válida dentro do contexto do agente, pois ele possui um plano específico para tratá-la, que é empacotar o objeto com o papel vermelho, portanto vai fazer parte do conjunto de percepções refinadas ρ. Após sair do módulo de refinamento percepções, a percepção é recebida como entrada pelo decisor 1, que detecta que existe um plano específico para tratar da entrada, portanto a percepção é considerada válida e é diretamente enviada para o raciocínio do agente.
- `lua(serrilhada)` – Lua não é um item que deveria ser percebido pelo agente, pois não faz parte dos itens que deveriam ser inseridos na esteira. Todavia, a função de refinamento de nosso agente simplesmente remove as percepções que envolvem estrelas, fazendo com que a percepção `lua(serrilhada)` chegue ao módulo de alucinação e ilusão. Dentro desse módulo, quando ela chega ao decisor 1, é classificada como anomalia, pois não faz parte do contexto do agente. Em seguida, a percepção é enviada para o decisor 2, que verifica que nem `lua` nem `serrilhada` fazem parte do contexto do agente, ou seja, é uma alucinação. Uma vez detectada a alucinação, a percepção segue para o bloco avaliador.

4.3.1 Bloco avaliador

Após uma anomalia ser classificada, ela é enviada para um bloco avaliador, que tratará de decidir qual anomalia pode ser considerada relevante para ser usada no planejamento automatizado e quais podem ser descartadas. De acordo com a implementação do módulo de refinamento, as percepções que são classificadas como anomalias podem ser utilizadas para otimizar o processo de refinamento.

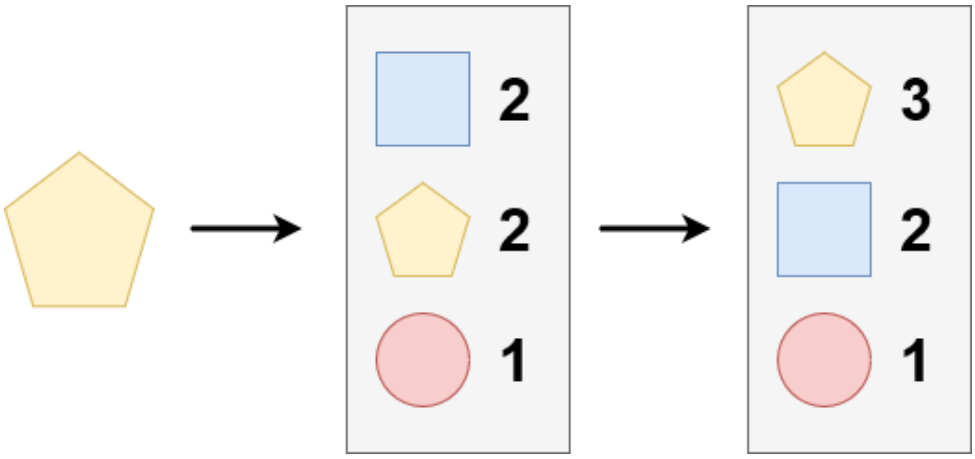
Em nosso modelo, utilizamos três blocos avaliadores: um para alucinações, um para ilusões classe 1 e outro para ilusões classe 2. Eles são separados para permitir que a prioridade de tratamento de alucinações seja definida individualmente, de acordo com a necessidade do agente que implementa o modelo.

O objetivo do bloco avaliador é decidir quando alucinações e ilusões que foram recebidas devem ser processadas, evitando que o planejamento automatizado que será realizado em seguida tenha impacto no tempo de execução de um ciclo de raciocínio do agente. Para isso, utilizamos uma lista ordenada por peso como escalonador. O principio do funcionamento da lista ordenada por peso é o mesmo de uma fila, *first in first out* (FIFO), mas atribui um peso a cada entrada que aumenta quando novos elementos iguais são inseridos. Quando um elemento é inserido pela primeira vez na fila, ele recebe o peso 1, e quando uma cópia do mesmo elemento é inserida o elemento tem seu peso aumentado em 1, como mostra a Figura 7. Quando uma

operação de remoção é executada, o elemento de maior peso é removido. Se dois ou mais elementos tiverem o maior peso, aquele que foi inserido primeiro é removido.

O bloco avaliador seleciona quando uma percepção deve ser tratada através de uma função matemática, levando em conta o tempo médio de processamento de uma percepção válida e de uma anomalia. Além desse funcionamento básico, o bloco avaliador ainda contém um mecanismo para remover anomalias classificadas com irrelevantes para o sistema, através de uma função de limpeza. Caso essa função retorne verdadeiro, todos os elementos de peso 1 da sua respectiva lista são removidos. Essas duas funções são descritas com mais detalhes na Seção 4.5.

Figura 7 – Exemplo do funcionamento de uma lista ordenada por peso.



Fonte: Autor

4.3.2 Bloco de planejamento automatizado

O bloco de planejamento automatizado é potencialmente a parte mais custosa computacionalmente, o que pode ser um gargalo do sistema, principalmente caso o agente funcione em tempo real e receba um volume muito elevado de percepções por segundo. Um planejamento automatizado implementado de maneira puramente simbólica tende a ser complexo computacionalmente, uma vez que pode considerar milhares de alternativas para o estado de mundo atual, tentando chegar mais perto de seu objetivo. Um processo de planejamento automatizado conexionista é uma alternativa, uma vez que estamos tratando de uma análise incompleta do mundo. Caso seja possível, teorias com maior custo computacional (como criatividade computacional (COLTON; WIGGINS *et al.*, 2012)) podem ser aplicadas aqui para um resultado ainda mais preciso.

Uma percepção chega ao bloco de planejamento automatizado uma vez que ela seja a primeira na fila ponderada e a função de processamento retorne verdadeiro em sua verificação. De um ciclo para outro, as percepções permanecem na fila, a não

ser que sejam descartadas pelo mecanismo de limpeza. Nosso modelo não explicita qual é a ordem que os blocos avaliadores devem processar suas filas para mandar anomalias para o planejamento automatizado (isto é, se deve primeiro ser priorizada as anomalias do bloco avaliador de alucinações, ilusões classe 1 ou ilusões classe 2), ficando a cargo da implementação em questão tomar essa decisão.

Exemplo 4.3.2. Para mostrar o caminho que faz uma ilusão, vamos considerar que o agente recebe em duas esteiras a percepção `triangulo(listrado)`. Não existem triângulos listrados na loja, mas como θ não filtra essa percepção, ela vai chegar ao decisor 1.

Os planos do agente são: `papel(vermelho) :- quadrado(_) OR circulo(liso)` e `papel(azul) :- circulo(listrado) OR triangulo(liso)`, e como não existe um plano específico para tratar a percepção `triangulo(listrado)`, ela é considerada uma anomalia, e é encaminhada ao decisor 2. Como existe um plano para tratar de triângulos lisos, a percepção é então classificada como uma ilusão, e vai para o terceiro decisor. Nele, como o plano trata triângulos, a percepção é considerada uma ilusão classe 1.

Após a percepção ter sido classificada corretamente, ela é enviada para o bloco avaliador, e é inserida na fila ponderada com peso 1. Depois disso, a percepção da segunda esteira, que é igual a que já foi tratada, chega ao bloco de alucinação e ilusão. Essa percepção vai fazer o mesmo caminho até o bloco avaliador e o peso da anomalia já presente na fila é aumentada para 2. Como duas percepções foram consideradas anomalias, vamos considerar que a função de processamento seja satisfeita, e o bloco avaliador passe essa anomalia para o bloco de planejamento automatizado.

Após uma percepção ser considerada relevante para ser enviada ao planejamento automatizado, ela deve gerar um novo plano para aquela percepção a ser adicionada ao conjunto de planos do agente, e essa percepção então deixará de ser uma anomalia. Ilusões e alucinações tem blocos de planejamento automatizado separados para permitir que duas implementações completamente diferentes sejam utilizadas de acordo com a função do agente e suas particularidades.

Retomando o exemplo 4.3.1, no segundo caso, quando o planejamento automatizado de alucinações recebe a percepção `lua(serrilhada)`, um processo de planejamento automatizado deve ser executado. Para esse exemplo, consideremos um planejamento automatizado puramente simbólico, que analisa uma grande quantidade de estados futuros possíveis para o ambiente em que o robô está inserido e seleciona aquele que será mais eficiente para que o agente se aproxime de seu objetivo principal (terminar de empacotar os itens). Isso é extremamente custoso, mas como alucinações são muito raras para esse agente em questão, pois ele já elimina as percepções inválidas reconhecidas no projeto durante o refinamento, não terá um impacto grande no desempenho do agente. No exemplo, é possível que a lua seja um

objeto novo que está sendo vendido, portanto, o agente precisa embalar esse item. Ao detectar uma semelhança entre a lua serrilhada e o circulo listrado, o planejamento automatizado pode determinar que esse novo objeto deve ser embalado com o papel azul, criando assim um novo plano da forma `papel(azul) :- lua(serrilhada)` é adicionado, e `lua(serrilhada)` deixa de ser uma alucinação.

No caso da percepção `triângulo(listrado)`, o agente poderia ter um bloco de planejamento automatizado baseado em uma rede bayesiana, uma vez que ilusões podem ser muito mais comuns e o agente já tem uma breve noção do que deve ser feita com objetos do tipo. `triângulo(listrado)` pode ser um novo item a venda na loja, e como já foi verificado em duas esteiras diferentes, faz sentido que ele não seja uma mera falha de sensores. Assim, o planejamento automatizado pode inferir o plano `triângulo(listrado) -> empacotar`, permitindo que o agente tenha um aprendizado dinâmico resultado da adição de um novo plano em seu conjunto de planos. Assim como no caso da ilusão, uma vez que esse plano novo foi adicionado a percepção original deixa de ser uma anomalia, uma vez que faz parte do contexto em que o agente está trabalhando.

4.4 RACIOCÍNIO DO AGENTE

O raciocínio do agente se refere a todos os fatores externos do modelo de revisão de percepções pertencentes ao agente, principalmente a arquitetura cognitiva utilizada para implementá-lo. Em um agente qualquer, o raciocínio é alimentado principalmente pelas percepções. Outras fontes de conhecimento, como comunicação, podem existir também, mas para o modelo HAIL consideramos toda forma de entrada como percepção. Em um agente que implemente o modelo HAIL, o raciocínio recebe tanto as percepções consideradas válidas pelo módulo de alucinação e ilusão quanto os novos planos criados a partir do planejamento automatizado.

4.5 FORMALIZAÇÃO

Nesta seção vamos formalizar o modelo HAIL através de um modelo de cascata, partindo dos conceitos mais gerais e afunilando para as definições mais específicas e fórmulas matemáticas utilizadas. Para isso, começamos com uma única tupla que define de maneira geral o que é o modelo que se desdobra para os demais conceitos. O objetivo disso é criar camadas de abstração, as quais podem ser modificadas de acordo com a necessidade de implementações específicas ou da integração com arquiteturas cognitivas ou outros modelos.

O bloco básico do modelo de revisão de percepções proposto, chamado de HAIL, é composto por um módulo para alucinação e ilusão M_{ai} e uma função de refinamento θ, conforme descrito na Definição 10. O módulo de ilusão e alucinação é

uma quádrupla, apresentada na definição 11. A função de refinamento é uma função abstrata, cuja entrada é obtida através dos sensores do agentes e a saída é a entrada do módulo de alucinação e ilusão, conforme já foi descrito anteriormente na Seção 4.2.

Definição 10. O modelo de revisão de percepções HAIL é uma dupla $HAIL = \langle M_{ai}, \theta \rangle$, onde:

- M_{ai} é o módulo de ilusão e alucinação; e
- θ é a função de refinamento $\theta(p) = \rho$, onde p é um conjunto de percepções e ρ é um subconjunto próprio de p.

Após ter passado pela função θ, as percepções ρ irão passar pelo Algoritmo 1, e serão encaminhadas de acordo com sua classificação. O bloco de ilusão e alucinação é descrito por uma quádrupla, com conjuntos de decisores, blocos e uma função de transição.

Definição 11. O bloco de ilusão e alucinação é uma quádrupla $M_{ai} = \langle D, Ab, Ap, \Delta \rangle$, onde:

- D é o conjunto de decisores $D = \{d_a, d_h, d_i\}$, onde:
  - d_a é o decisor de anomalias, definido pela função:

$$d_a = \begin{cases} 0 & \text{se } \rho(x) \text{ está em } c^1; \\ 1 & \text{se } \rho(x) \text{ não está em } c. \end{cases}$$

  - d_h é o decisor de alucinação, definido pela função:

$$d_h = \begin{cases} 0 & \text{se nem } \rho \text{ nem } (x) \text{ está em } c; \\ 1 & \text{se } \rho \text{ ou } (x) \text{ está em } c. \end{cases}$$

  - d_i é o decisor de ilusão, definido pela função:

$$d_i = \begin{cases} 0 & \text{se } \rho \text{ está em } c; \\ 1 & \text{se } (x) \text{ está em } c. \end{cases}$$

- Ab é o conjunto de blocos avaliadores $Ab = \{Ab_h, Ab_{i1}, Ab_{i2}\}$, onde Ab_h é o bloco avaliador de alucinações, Ab_{i1} é o bloco avaliador de ilusões classe 1 e Ab_{i2} é o bloco avaliador de ilusões classe 2.
- Ap é o conjunto de blocos de planejamento automatizado $Ap = \{Ap_h, Ap_i\}$, onde Ap_h é o bloco de planejamento automatizado de alucinações e Ap_i é o bloco de planejamento automatizado de ilusões.

[1] c é o contexto do agente, de acordo com a definição 2.

- Δ é a função de transição definido pela tabela abaixo, onde *out* é um estado final, que leva a percepção para fora do modelo de revisão de percepções, ou seja, pode tanto significar o encaminhamento de uma percepção válida para o raciocínio do agente quanto o fim da execução de um ciclo de revisão.

Tabela 1 – Função de transição Δ do módulo de ilusão e alucinação.

Estado	**0**	**1**
d_a	*out*	d_h
d_h	Ab_h	d_i
d_i	Ab_{i1}	Ab_{i2}
Ab_h	*out*	Ap_h
Ab_{i1}	*out*	Ap_i
Ab_{i2}	*out*	Ap_i

Fonte: Autor.

O módulo de ilusão e alucinação é o artefato principal do modelo. Ele recebe uma entrada ρ, que é a saída da função de refinamento apresentada na definição 1, e processa cada um dos elementos $\rho(x)$ desse conjunto, através de decisores e blocos de avaliação, percorrendo o modelo de acordo com as transições descritas pela função de transição Δ. Os três decisores do conjunto D fazem a triagem para detectar se a percepção $\rho(x)$ é uma anomalia, e que tipo de anomalia é. Após passar pelos três decisores, saberemos se essa percepção é valida, é uma alucinação, é uma ilusão tipo 1 ou uma ilusão tipo 2. Após ter passado pelos decisores, a percepção ou é levada para a cognição do agente, caso seja válida, ou fica armazenada nos blocos avaliadores, caso seja uma anomalia.

Definição 12. Um bloco avaliador é uma tripla $Ab_X = \langle L, Pf, Cf) \rangle$, $x \in \{h, i1, i2\}$, onde:

- L é uma lista ordenada pelo número de vezes que uma mesma anomalia é dada como entrada;
- Pf é a função de processamento, definida abaixo:

$$Pf = \begin{cases} 1 & \text{se } T_m(A) \leq T_m(V) * (|A| - |A_{pr}|); \\ 0 & \text{caso contrário.} \end{cases}$$

  Onde:

  - T_m é a função que retorna a média do tempo gasto para processar as percepções de um conjunto;
  - A é o conjunto de anomalias, $A(x)$ é um elemento específico x e $|A|$ o número de anomalias do conjunto;

  - A_{pr} é o conjunto de anomalias que já foram validadas para serem processadas pela função de processamento neste ciclo de raciocínio (A_{pr} é instanciada vazia a cada ciclo de raciocínio), e $|A_{pr}|$ o número de anomalias desse conjunto.
  - V é o conjunto de percepções válidas.

- Cf é a função de limpeza definida abaixo com auxílio da função equação de limpeza Cf, sendo α um coeficiente de limpeza variável que precisa ser definido pela instância implementada do modelo (por padrão, toma-se $\alpha = 1$):

$$Cf = \begin{cases} 1 & \text{se } Ce = \textit{Verdadeiro}; \\ 0 & \text{caso contrário.} \end{cases}$$

$$Ce = \sum_{i=1}^{|L|} W_n(L_i) > \alpha \sum_{j=1}^{|L|} W_1(L_j)$$

  Onde:

  - L é a lista ordenada do bloco, sendo $|L|$ seu número de anomalias e L_i a anomalia i da lista.
  - W é a função peso da anomalia L_i definida como $W(L_i) = |L_i|$, sendo $|L_i|$ o peso da anomalia especificada (número de entradas recebidas dessa mesma anomalia na lista). A função W é utilizada para especificar as seguintes funções:

(i) $$W_1(L_i) = \begin{cases} 1 & \text{se } W(L_i) = 1; \\ 0 & \text{caso contrário.} \end{cases}$$

(ii) $$W_n(L_i) = \begin{cases} W(Li) & \text{se } W(L_i) > 1; \\ 0 & \text{caso contrário.} \end{cases}$$

A terceira definição é a de bloco de avaliação (*Ab*). Um *Ab* é um módulo do modelo que é responsável por armazenar as anomalias detectadas e decidir se elas serão processadas pelo agente ou não. É descrito por uma tripla, constituída por uma lista ordenada L, uma função de processamento Pf e uma função de limpeza Cf. L é uma lista organizada pela recorrência de elementos inseridos nela, onde cada elemento só aparece uma vez e contém um número de vezes que o mesmo elemento já foi inserido nela, chamado de peso. Nesse modelo, os elementos são as anomalias percebidas pelo agente, e o peso é o número de vezes que o agente percebeu a anomalia.

Pf é uma função que avalia se uma anomalia será processada nesse ciclo de raciocínio ou se será armazenada para ser processada no futuro. Para isso, ela precisa de uma função que retorne o tempo médio previsto para o processamento de uma percepção T_m, seja ela uma percepção válida ou uma anomalia. Com base nessa função, *Pf* retorna 1 caso o tempo médio de processamento de uma anomalia seja menor que o tempo médio de processamento de uma percepção válida multiplicado pelo número de anomalias que fazem parte desse ciclo de raciocínio menos o número de anomalias que já foram aprovadas pelo bloco avaliador, e zero caso contrário.

De maneira simplificada, o objetivo dessa função é evitar que o modelo de revisão de percepção gaste mais tempo de processamento do que ele gastaria caso não estivesse sendo utilizado e todas as percepções fossem válidas. Para isso, o bloco avaliador permite processar apenas as anomalias que aparecem de maneira mais recorrente para o agente.

Cf é uma função que realiza a limpeza de *L*. Conforme os ciclos de raciocínio forem passando, *L* tende a possuir diversas anomalias que foram percebidas apenas uma única vez. Dessa maneira, uma grande quantidade de memória seria necessária para armazenar as possíveis centenas de anomalias que podem nunca ser processadas. Assim, a função *Cf* verifica se a equação *Ce* é verdadeira ou falsa. Ela é verdadeira quando o número de anomalias que apareceram uma única vez é maior que a soma dos pesos das anomalias que apareceram mais de uma vez (o peso é o número atrelado a cada anomalia, que representa quantas vezes elas já foram inseridas na lista). Quando a função for verdadeira, o bloco avaliador remove todas as anomalias de peso 1 da lista.

O coeficiente de limpeza α presente em *Ce* pode ser utilizado na implementação do HAIL quando é necessário que a limpeza seja mais ou menos recorrente. Por padrão, caso a instância do HAIL não queira alterar a frequência da limpeza das filas, utiliza-se 1. Utilizando valores menores que 1 a função será verdadeira mais facilmente, e o contrário caso o valor seja maior que isso.

Quando P_f retorna 1, a anomalia do topo da lista de algum dos três blocos avaliadores é removida, e usada como entrada do bloco de planejamento automatizado. Qual bloco avaliador tem a prioridade para ser utilizado é uma decisão que deve ser tomada pela implementação.

Definição 13. Um bloco de planejamento automatizado é uma instância do modelo conceitual de planejamento automatizado (Definição 9).

Por fim, a saída do planejamento automatizado, que é um plano ou um conjunto de planos novos que o agente deve adicionar ao conjunto de planos que possui, é enviada para o raciocínio do agente, dando fim ao processo de revisão de percepções daquele ciclo.

4.6 GENERALIDADE

O HAIL foi um modelo construído para ser genérico. Sua generalidade pode ser avaliada em dois quesitos: a capacidade de ser acoplado a qualquer arquitetura cognitiva ou modelo de agente inteligente e a capacidade de ser modificado de acordo com as necessidades da arquitetura à qual será acoplado.

Quanto a capacidade de acoplamento, podemos considerar o HAIL um modelo genérico uma vez que ele atua sobre as percepções do agente e não sobre seu processo cognitivo, ou seja, sua entrada é composta apenas por um subconjunto das percepções recebidas e os planos que podem ser gerados pelo módulo de planejamento automatizado.

No Capítulo 2, descrevemos um agente como uma entidade que recebe percepções do ambiente e o modifica através de ações. A Figura 8 demostra a diferença entre a interação normal de um agente com o ambiente e quando há a adoção do modelo HAIL. Portanto, contanto que o agente utilize mecanismos de percepção e raciocínio que permitam definir bem seu contexto, permitindo que o HAIL possa ser implementado conforme definido na Seção 4.5, o modelo pode ser inserido e removido do processo de percepção do agente livremente.

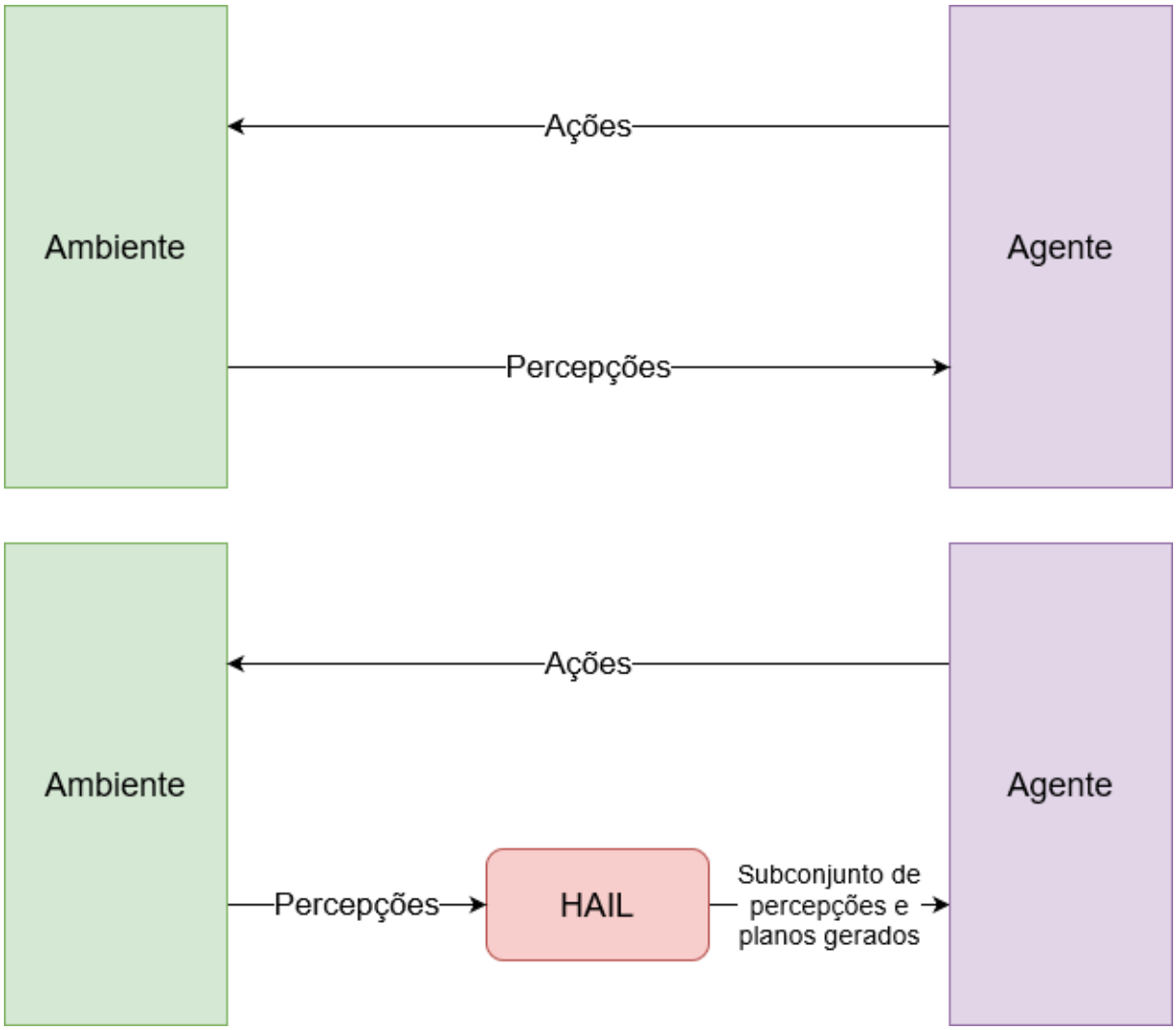
O outro aspecto da generalidade do modelo diz respeito a sua alterabilidade. As principais partes modificáveis do HAIL são o módulo de revisão de percepções, o coeficiente de limpeza α utilizado pela função de limpeza dos blocos avaliadores e os blocos avaliadores.

O módulo de revisão de percepções pode ser implementado para utilizar qualquer técnica de refinamento de percepções, ou então não utilizar técnica alguma. Isso foi definido pois dependendo do tipo de ambiente no qual o agente está inserido pode ser necessário analisar todas as percepções recebidas ou filtrar de maneira bastante rígida quais percepções serão processadas.

O coeficiente de limpeza foi criado para que seja possível ajustar a frequência com a qual as listas ponderadas são limpas, tornando possível que agentes em situações com muitas anomalias não utilizem tanta memória e agentes em situações críticas possam analisar anomalias dentro de um maior período de tempo.

Por fim, o bloco de planejamento automatizado não teve uma estratégia ou arquitetura definida pois esse tipo de mecanismo é bastante complexo e não era o foco do trabalho. Além disso, como foi mostrado no Exemplo 4.3.2, deixar a implementação do planejamento automatizado em aberto permite que o desenvolvedor adéque o modelo conforme a necessidade.

Figura 8 – Interação entre o ambiente e um agente sem e com o modelo HAIL.



Fonte: Autor

5 EXPERIMENTOS

Para este trabalho foram realizados três experimentos com o objetivo de averiguar se o modelo HAIL é funcional ou não. Esses experimentos não possuem um domínio específico, ou seja, não há um ambiente real onde o agente está inserido, ele recebe percepções aleatórias geradas pelo simulador e os planos inicias do agente não são relevantes para os testes.

Os dois primeiros experimentos foram implementados utilizando o design 2^k fatorial (JAIN, 1990). Esse tipo de design consiste em variar k fatores em 2 níveis diferentes, -1 e 1, que são extremos opostos. Por exemplo, em uma pesquisa ligada a um processador, um fator pode ser o número de núcleos, e seus níveis serem 1 núcleo e 8 núcleos. O fator é uma variável livre, utilizada para analisar a variação de uma variável dependente qualquer. Os fatores e as variáveis livres utilizadas estão nas Tabelas 2 e 3, respectivamente. A análise do impacto dos fatores foi realizada utilizando a equação de regressão não linear do design 2^k fatorial.

Tabela 2 – Fatores utilizados nos experimentos realizados com o modelo HAIL.

Fatores	**Sigla**	**Nível -1**	**Nível 1**
Porcentagem de Percepções Inválidas	PPI	5%	95%
Tempo Médio gasto pelo planejamento Automatizado	TMA	1/2 T	64 T
Tempo Médio gasto em um Ciclo de raciocínio	TMC	01 T	32 T
Número de Percepções recebidas por Ciclo	NPC	01	16

Fonte: Autor.

Tabela 3 – Variáveis dependentes analisadas nos experimentos realizados com o modelo HAIL.

Variáveis	**Motivação**
Tempo Virtual Decorrido	Medir desempenho geral do modelo
Planos Criados	Avaliar potencial do modelo de inserir aprendizado em arquiteturas que não o possuem
Percepções Válidas Processadas	Analisar a capacidade do modelo de ganhar desempenho ao longo do tempo

Fonte: Autor.

O terceiro experimento não utiliza o design 2^k fatorial. Após as simulações do segundo experimento, foi identificada a demanda de analisar como o HAIL era impactado com o processo contínuo de aprendizado fornecido pelo bloco de planejamento auto-

matizado. Este experimento não utiliza um design específico, e será melhor explicado na Seção 5.4.

Os experimentos consistem em diversas simulações, que submetem o modelo proposto ao processo de revisão de um grande número de percepções. Uma simulação possui 5000 ciclos de percepção, sendo que cada ciclo pode possuir uma ou várias percepções. Essas percepções podem ser válidas (pertencentes ao contexto do agente) ou inválidas (não pertencentes ao contexto do agente), sendo que a proporção entre o tipo de percepções é definido pela PPI. As percepções são produzidas aleatoriamente por um gerador de percepções.

O gerador de percepções cria as percepções válidas sorteando símbolos que pertencem ao contexto do agente, e as percepções inválidas são criadas utilizando o pacote `RandomWords` (ŚWIECICKI, 2012), que gera palavras em inglês aleatórias. As percepções são símbolos compostos da forma `corpo(argumento)`, e a quantidade de percepções válidas e inválidas geradas é baseado no PPI e TMA da simulação executada.

Cada ciclo da simulação segue os seguintes passos:

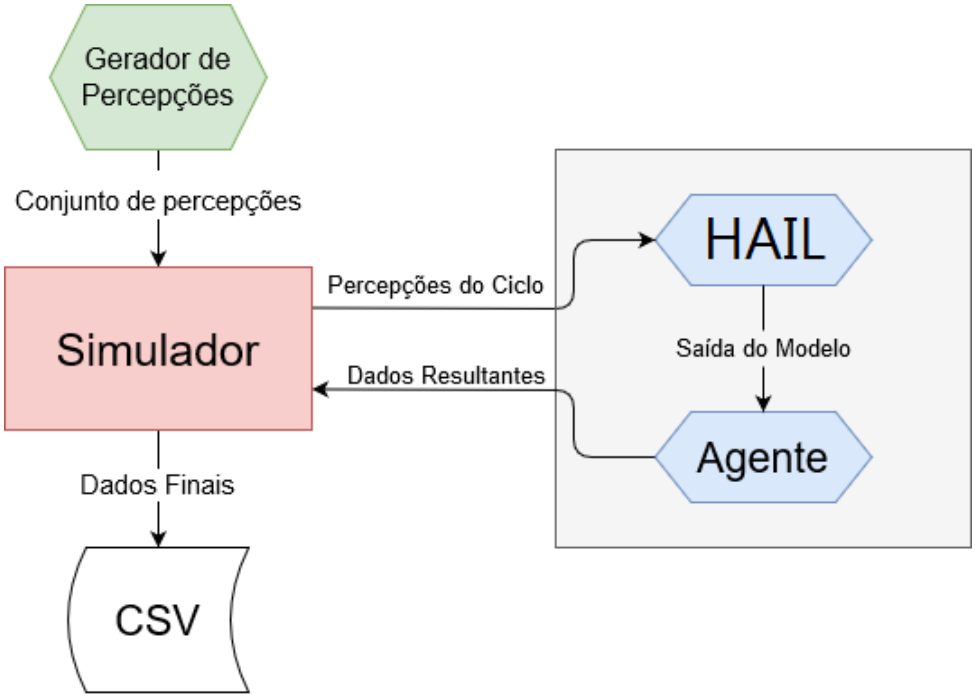
1. Gerar as percepções da simulação;
2. Iterar sobre cada um dos ciclos, passando as percepções para o modelo implementado;
3. Salvar os resultados, o agente final (com novos planos gerados pelo módulo de planejamento automatizado) e as percepções de cada ciclo em arquivos CSV.

Além da configuração dos valores dos fatores, é possível configurar se um agente será recarregado (ou seja, se ele deve manter os planos gerados por uma simulação anterior) e se o simulador deve gerar novas percepções ou não, pulando a etapa 1 de cada ciclo.

Para mensurar o tempo gasto pelos processos do agente, será considerada uma unidade de tempo genérica T, e iremos considerar o tempo virtual da execução das simulações com base nessa unidade. Portanto em uma simulação que possui TMC de 1T e TMA de 64T, o tempo médio do planejamento automatizado é 64 vezes maior do que o tempo médio do ciclo de raciocínio do agente.

Foi utilizado o agente do Exemplo 2.3.1 nas simulações, mas ele não foi descrito de acordo com nenhuma linguagem de programação de agentes real. Suas regras utilizam o formato de notação *percepcao*– > *acao*1; *acao*2; ...; *acaoN*. As ações desta implementação, assim como as percepções, são símbolos da forma `corpo(argumento)`. O módulo de refinamento não foi implementado, ou seja, sua entrada e saída são equivalentes.

Figura 9 – Diagrama da execução de uma iteração de cada simulação.



Fonte: Autor.

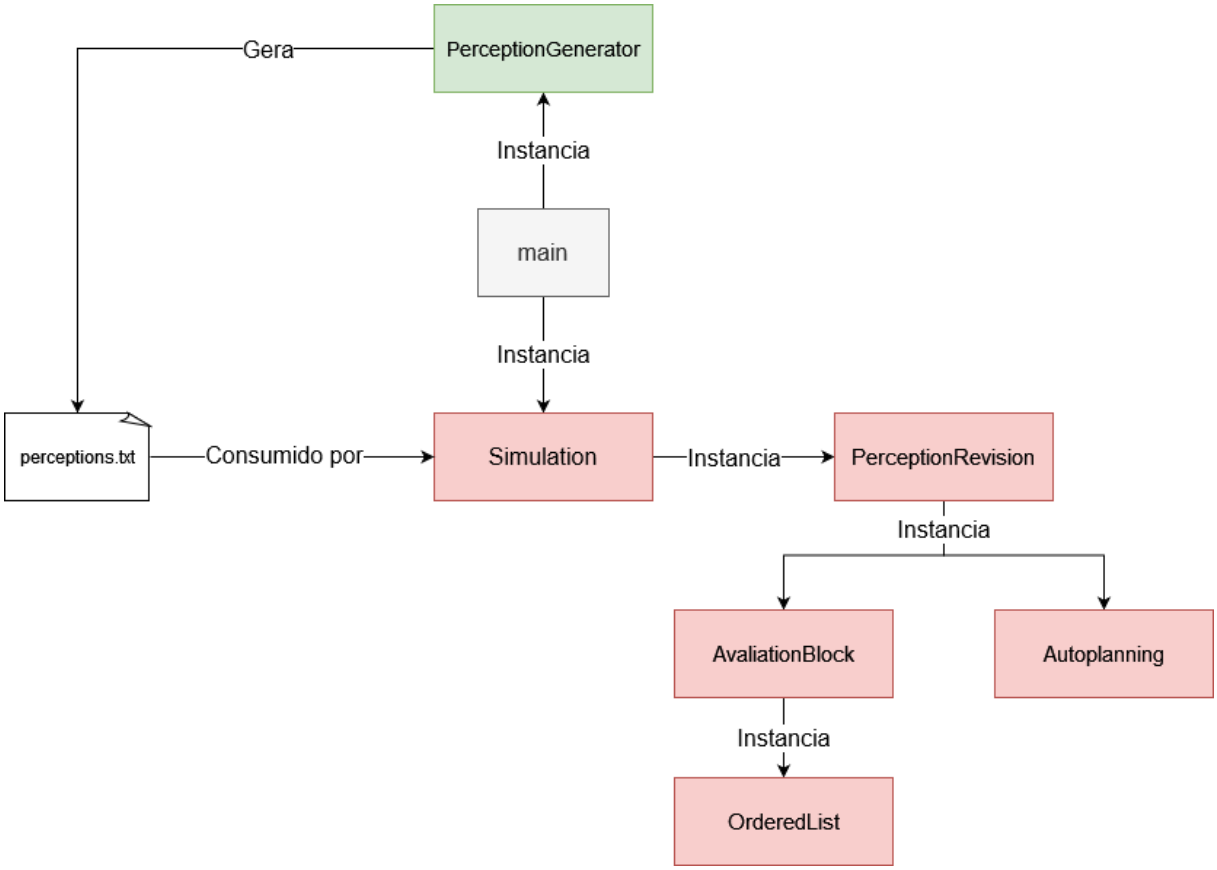
5.1 IMPLEMENTAÇÃO

A implementação do simulador foi feita em Python 3.7, e conta com três classes principais: `PerceptionGenerator`, `PerceptionRevision` e `Simulator`. As interações das classes do simulador a partir do método `main` estão ilustradas na Figura 10.

O gerador de percepções foi abstraído em um pacote chamado de `generator`, e sua classe é implementada no arquivo `perceptions.py`. Esta classe deve ser instanciada passando como argumento o número de ciclos de percepção que serão utilizados na simulação, a porcentagem de percepções inválidas e o número de percepções por ciclo. Depois de instanciado, é possível utilizar o objeto para chamar a função `generate`, que utiliza a configuração recebida para criar as percepções e escrevê-las num arquivo `perceptions.txt`. As percepções inválidas são geradas aleatoriamente com o pacote `RandomWords` e as percepções válidas são geradas a partir da especificação do arquivo `perceptions_options.py`, onde são definidas o corpo e o argumento das percepções válidas.

O modelo HAIL foi implementado na classe `PerceptionRevision`, que está contido no arquivo `pr_system.py`. Ele implementa cada um dos processos descritos no Capítulo 4. Quando esta classe é instanciada, ela primeiro extrai o contexto do agente através do método auxiliar `get_agent_context`, que percorre o arquivo do agente buscando os símbolos que compõem o contexto no corpo e no argumento dos símbolos dos planos. Depois disso, são instanciados os blocos de avaliação como atributos

Figura 10 – Diagrama demonstrando o relacionamento entre as classes do simulador.



Fonte: Autor.

da classe do revisor, um para cada tipo de anomalia. Eles são implementados em uma classe chamada `AvaliationBlock`. Cada bloco guarda uma instância da classe `OrderedList`, que representa a lista ordenada definida pelo modelo. Por último, o bloco de planejamento `Autoplanner` é instanciado.

Como o bloco de planejamento automatizado não é o foco do trabalho, e possui uma grande complexidade de implementação, foi criado uma versão que cria um plano novo juntando a anomalia recebida com um conjunto aleatório de ações que o agente possui. Além disso, nessa implementação alucinações e ilusões utilizam o mesmo método de planejamento automatizado, portanto apenas um bloco é instanciado.

A anomalia `studies(distortion)` submetida ao planejamento automatizado pode gerar a percepção `studies(distortion) -> wrap_up(circle)`, por exemplo. A implementação do `Autoplanner` gera um número aleatório de ações por plano, podendo ser uma ou no máximo o número de ações que o agente possui (nesse caso dois).

O revisor de percepções possui um método principal utilizado nas simulações chamado `process_perceptions`, que recebe uma lista de percepções, submete cada percepção ao fluxo do HAIL e então retorna o tempo virtual decorrido e o número

de percepções válidas processadas. Para isso, cada percepção é classificada pelos decisores do módulo de alucinação e ilusão e então adicionada ao bloco avaliador designado. Quando todas as percepções foram classificadas, o modelo escolhe se vai processar as ilusões ou alucinações primeiro. Nessa implementação, essa escolha é feita aleatoriamente. Os blocos avaliadores passam pelo processo de validação da função de processamento, e anomalias são enviadas para o planejamento automatizado enquanto a função retornar verdadeiro.

Por fim, a classe `Simulation` utiliza as duas classes descritas anteriormente para executar a simulação. As percepções do arquivo `perceptions.txt` são lidas pelo método `run`, executando cada um dos ciclos de percepção utilizando uma instância da classe `PerceptionRevision`. Os valores finais de tempo virtual, percepções processadas e planos criados são retornados e guardados em um arquivo CSV.

Além da simulação, outro projeto foi criado para analisar o impacto dos fatores, medir o erro dos experimentos e criar os gráficos. Ele também foi implementado em Python 3.7 e utiliza as ferramentas `pandas` (REBACK *et al.*, 2020), `matplotlib` (HUNTER, 2007) e `seaborn` (WASKOM, 2021).

5.2 EXPERIMENTO 1

O objetivo do primeiro experimento é analisar o impacto da mudança dos fatores no desempenho do modelo através da analise da variação no tempo virtual decorrido e da quantidade de planos criados pelo módulo de planejamento automatizado.

Esse experimento foi o primeiro realizado com a metodologia do 2^k fatorial. Cada um dos fatores foi avaliado em ambos os níveis apresentados na Tabela 2, resultando em dezesseis combinações diferentes, e portanto dezesseis simulações. O tempo virtual foi escolhido para termos uma métrica geral de desempenho do sistema, enquanto a quantidade de planos criadas foi escolhida por ser um dos principais diferenciais do sistema, e um de seus aspectos mais importantes.

Cada simulação foi repetida dez vezes, e a média dos resultados foi obtida. Isso foi feito para garantir que os valores obtidos não foram afetados pela geração aleatória de percepções. Quando repetições são adotadas no experimento normalmente utiliza-se a metodologia $2^k r$ fatorial, que leva em conta o erro obtido entre as diferentes execuções da simulação. Todavia, ao fazer a análise de erros o resíduo obtido foi descartável, com ordem de grandeza e^{-12}. Portanto, a metodologia de análise adotada foi a do 2^k fatorial.

5.2.1 Resultados

As médias dos resultados de cada simulação estão apresentadas na Tabela 4. O valor final do tempo virtual decorrido varia bastante, principalmente nas simulações

com alto nível de percepções inválidas. Em dois momentos o tempo virtual foi 5000T. Esse valor exato se repetiu pois nessas simulações o TMC é 1, o TMA é 0.5 e o NPC é 16. Nessa configuração os ciclos de percepções válidas consumiam 1 unidade de tempo, totalizando um tempo virtual de 4750T. O bloco avaliado sempre permitia o processamento de 2 percepções inválidas (pois cada uma toma 0.5 unidades de tempo), e quase sempre há percepções inválidas disponíveis na fila pois entram 16 percepções inválidas por ciclo de percepções inválidas. Portanto, cada vez que o modelo processava percepções inválidas, ele consumia 1 unidade de tempo, e isso aconteceu 250 vezes (5% de 5000).

Tabela 4 – Resultados do experimento 1.

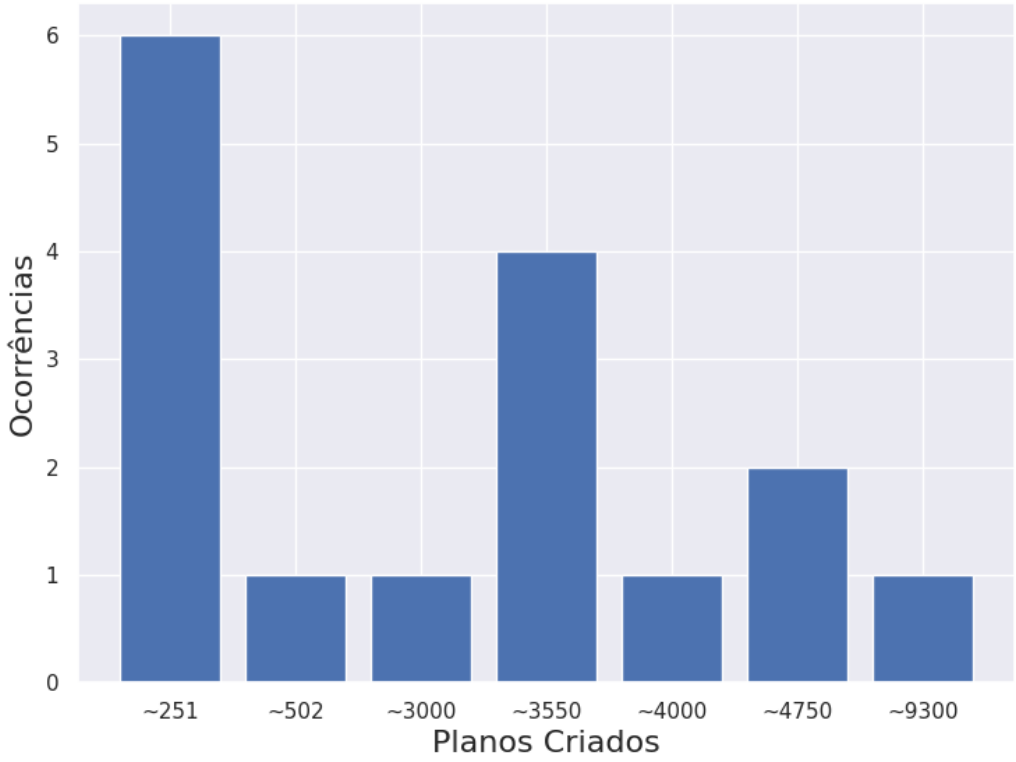
Simulação	PPI	TMC	TMA	NPC	Tempo Virtual	Planos Criados
1	5%	1	1/2	1	4874.6T	250.8
2	5%	1	1/2	16	5000.0T	502.0
3	5%	1	64	1	20806.7T	250.9
4	5%	1	64	16	20813.0T	251.0
5	5%	32	1/2	1	152093.5T	251.0
6	5%	32	1/2	16	153437.3T	2932.2
7	5%	32	64	1	168028.8T	250.9
8	5%	32	64	16	168032.0T	251.0
9	95%	1	1/2	1	3229.2T	3541.6
10	95%	1	1/2	16	5000.00T	9303.8
11	95%	1	64	1	228668.9T	3550.3
12	95%	1	64	16	304300.4T	4750.8
13	95%	32	1/2	1	48521.5T	3539.0
14	95%	32	1/2	16	140683.3T	4073.8
15	95%	32	64	1	273609.6T	3550.3
16	95%	32	64	16	312032.0T	4751.0

Fonte: Autor.

O número de planos criados se agruparam em certos conjuntos de valores, de acordo com os níveis dos fatores. Como esperado, as simulações com menor porcentagem de percepções inválidas criaram menos planos, pois houveram menos anomalias, logo menos material para o módulo de planejamento automatizado trabalhar. A Figura 11 mostra a ocorrência de valores próximos de planos criados para cada uma das faixas obtidas nas simulações. Por exemplo, as simulações 12 e 16 tiveram aproximadamente 4750 percepções criadas, portanto foram agrupadas na mesma faixa no gráfico. Essas faixas de valores indicam que há uma discrepância entre as simulações com PPI de 5% (com 6 ocorrências de aproximadamente 250 planos criados) e as simulações com PPI 95%. Com isso, é possível notar que a principal condição para que o bloco de planejamento automatizado crie uma quantidade elevada de planos novos

é um grande percentual de anomalias em relação ao total de percepções recebidas.

Figura 11 – Recorrência do valor final de Planos Criados nas simulações realizadas.



Fonte: Autor.

5.2.2 Análise de fatores

Conforme mostra a Tabela 5, na variável dependente de Tempo Virtual, os fatores TMC e TMA isolados tiveram uma alta porcentagem de impacto. Isso porque a contagem do tempo virtual acontece em função do tempo médio do ciclo de raciocínio e do tempo médio do planejamento automatizado. Portanto, a interferência nesses fatores influencia diretamente o resultado final. Além disso, a combinação dos fatores PPI + TMA teve um impacto de 24.13%. Isoladamente, PPI e TMA possuem porcentagens altas (12.69 e 31.64, respectivamente) e quanto mais percepções inválidas há, maior o tempo gasto no planejamento automatizado, pois mais anomalias podem ser processadas. Como cada fator possui impacto alto e seus valores interferem diretamente um com o outro, a combinação de ambos acaba sendo relevante para o resultado das simulações. Por outro lado, TMC possui 22.20% de impacto, mas a combinação PPI + TMC possui apenas 4.16, pois a porcentagem de percepções inválidas não interage diretamente com o tempo gasto no ciclo de raciocínio.

Tabela 5 – Análise dos fatores do experimento 1.

Fator	**Efeito Tempo Virtual**	**Efeito Planos Criados**
PPI	12.69%	66.10%
TMC	22.20%	0.50%
TMA	31.64%	2.95%
NPC	1.44%	8.67%
PPI + TMC	4.16%	3.76%
PPI + TMA	24.13%	0.05%
PPI + NPC	1.39%	2.13%
TMC + TMA	0.55%	0.50%
TMC + NPC	0.10%	0.50%
TMA + NPC	0.01%	2.99%
PPI + TMC + TMA	0.53%	3.76%
PPI + TMC + NPC	0.09%	3.75%
PPI + TMA + NPC	0.02%	0.06%
TMC + TMA + NPC	0.54%	0.50%
PPI + TMC + TMA + NPC	0.52%	3.76%

Fonte: Autor.

Isso demonstra como uma combinação de alta taxa de percepções inválidas com um tempo de planejamento automatizado alto podem ser impactantes no NPC. Portanto, em situações com um grande volume de percepções, implementações que possuem um bloco de planejamento automatizado que consome muito tempo para criar novos planos podem prejudicar o desempenho do agente caso ele seja muito sensível ao tempo.

Na variável dependente Planos Criados, o fator que mais impactou o resultado foi o PPI. Isso era o esperado, pois quanto mais percepções inválidas o modelo recebe, mais planos podem ser criados. Os demais fatores tiveram impactos mais baixos. O NPC teve 8.67% de influência pois simulações com mais percepções inválidas por ciclo preenchem a fila ponderada mais rápido, permitindo que o bloco avaliador sempre possua elementos na fila ponderada para enviar ao bloco de planejamento automatizado.

5.3 EXPERIMENTO 2

O segundo experimento foi realizado para analisar o ganho de performance de um agente ao utilizar os planos criados com o bloco de planejamento automatizado. Esse segundo experimento segue a metodologia do primeiro, porém foi realizado apenas uma iteração para cada configuração de fatores (ao invés das dez repetições). Após ser realizada uma simulação com uma configuração de fatores foi realizada uma nova simulação com a mesma configuração, mas usando o agente resultante

da primeira. Assim, os planos criados pelo agente foram reaproveitados. Ao invés de analisar quais são os fatores que mais influenciam nas variáveis dependentes, comparamos os resultados dessas variáveis antes e depois da criação de novos planos. As variáveis dependentes analisadas foram novamente o tempo virtual decorrido e o número de planos criados.

Esse experimento pode ter sua configuração de fatores separadas em dois grupos: (I) o tempo de processamento de um ciclo de raciocínio é menor do que o tempo do planejamento automatizado; (II) o tempo do planejamento automatizado é menor que o tempo de processamento de um ciclo de raciocínio. Essa separação pode ser feita pois quanto mais planos são criados, menos percepções inválidas serão recebidas pelo agente. Portanto, se o tempo de processamento de um ciclo for maior que o da geração de novos planos, o tempo virtual gasto total usando um agente que já aprendeu vários planos será maior. Isso é uma troca realizada pois agora essas novas percepções (que eram antes anomalias) estão sendo recebidas pelo agente e resultando na execução de um plano.

5.3.1 Resultados

Os resultados estão separados em tabelas por variável dependente e por grupo, como descrito anteriormente. As tabelas contém o índice da simulação, os níveis dos fatores, os resultados da primeira e da segunda iteração (R1 e R2, respectivamente) e a relação percentual (RP) entre os resultados, ou seja, a proporção de R2 em relação a R1, obtido através do cálculo $(R2 * 100)/R1$.

Em todas as tabelas, podem ser observadas um menor efeito na relação percentual nas simulações que utilizaram o nível –1 do fator PPI (5% de percepções inválidas). Isso acontece pois o modelo possui menor impacto em ambientes de baixo volume de percepções inválidas, uma vez que o bloco de planejamento automatizado apenas cria novos planos quando há percepções inválidas na fila ponderada do bloco avaliador.

Na Tabela 6, a média da relação percentual foi de 75.26%, se destacando a simulação 5 com uma relação percentual de 4.77%. Com o tempo de planejamento automatizado alto, tempo de processamento de um ciclo de raciocínio baixo e apenas uma percepção por ciclo, há um grande ganho de desempenho, conforme explicado anteriormente. Um comportamento similar (porém invertido) pode ser observado na Tabela 7, cuja média da relação percentual foi de 136.33%. Nessas simulações, o esperado era que o tempo virtual subisse conforme a quantidade de planos criados aumentasse, pois o tempo gasto no planejamento automatizado é menor que o tempo gasto no ciclo de raciocínio.

A outra variável analisada foi o número de percepções válidas processadas. A média de cada configuração de fatores foi de 30563.0625 percepções, enquanto nas segundas simulações foi de 40599.75, totalizando um ganho aproximado de 32.84%

Tabela 6 – Alteração no valor de tempo virtual nas simulações do grupo I.

Simulação	PPI	TMC	TMA	NPC	R1	R2	RP
1.1	5%	1	64	1	20750T	20687T	99.70%
1.2	5%	1	64	16	20813T	20750T	99.70%
1.3	5%	32	64	1	168032T	167968T	99.96%
1.4	5%	32	64	16	168032T	168000T	99.98%
1.5	95%	1	64	1	227579T	10859T	4.77%
1.6	95%	1	64	16	304313T	177998T	58.49%
1.7	95%	32	64	1	273536T	162400T	59.37%
1.8	95%	32	64	16	312032T	250048T	80.14%

Fonte: Autor.

Tabela 7 – Alteração no valor de tempo virtual nas simulações do grupo II.

Simulação	PPI	TMC	TMA	NPC	R1	R2	RP
2.1	5%	1	1/2	1	4874.5T	4876.5T	100.04%
2.2	5%	1	1/2	16	5000T	5000T	100%
2.3	5%	32	1/2	1	152093.5T	152219.5T	100.08%
2.4	5%	32	1/2	16	153435.5T	152887T	99.64%
2.5	95%	1	1/2	1	3228.5T	4955T	153.48%
2.6	95%	1	1/2	16	5000T	4944T	98.88%
2.7	95%	32	1/2	1	48458.5T	157385.5T	324.78%
2.8	95%	32	1/2	16	160000T	140631T	113.77%

Fonte: Autor.

de desempenho. Ou seja, apesar do ganho de tempo virtual em algumas situações, um maior número de percepções foi processada e resultou na execução de um plano do agente.

As Tabelas 8 e 9 mostram a variação do ganho de planos criados entre as simulações. Ambas apresentam uma queda drástica nessa variável nas simulações com 95% de percepções inválidas, pois foram nessas simulações que o modelo pode criar mais planos (pois o planejamento automatizado é utilizado mais vezes). As simulações 1.6 e 1.8 possuem um resultado acima das simulações 1.5 e 1.7, apesar das quatro possuírem PPI de 95%, pois possuem mais percepções (16 vezes mais por conta do NPC) e processam no máximo uma percepção inválida por ciclo (pois o TMA é maior que o TMC). A Tabela 9 possui valores ainda menores na relação percentual pois muitas percepções inválidas são processadas por ciclo, em especial na simulação 2.8, onde nenhum plano novo foi criado na segunda simulação. Isso ocorreu pois o TMC é alto, o TMA é baixo e ela possui o maior número possível de percepções inválidas, resultando em um uso constante do planejamento automatizado.

Tabela 8 – Alteração no valor de planos criados nas simulações do grupo I.

Simulação	PPI	TMC	TMA	NPC	R1	R2	RP
1.1	5%	1	64	1	250.0	249.0	99.60%
1.2	5%	1	64	16	251.0	250.0	99.60%
1.3	5%	32	64	1	251.0	249.0	99.20%
1.4	5%	32	64	16	251.0	250.0	99.60%
1.5	95%	1	64	1	3533.0	93.0	2.63%
1.6	95%	1	64	16	4751.0	2746.0	57.80%
1.7	95%	32	64	1	3548.0	75.0	2.11%
1.8	95%	32	64	16	4751.0	2814.0	59.23%

Fonte: Autor.

Tabela 9 – Alteração no valor de planos criados nas simulações do grupo II.

Simulação	PPI	TMC	TMA	NPC	R1	R2	RP
2.1	5%	1	1/2	1	251.0	247.0	98.41%
2.2	5%	1	1/2	16	502.0	500.0	99.60%
2.3	5%	32	1/2	1	251.0	247.0	98.41%
2.4	5%	32	1/2	16	2935.0	878.0	29.91%
2.5	95%	1	1/2	1	3543.0	90.0	2.54%
2.6	95%	1	1/2	16	9312.0	260.0	2.79%
2.7	95%	32	1/2	1	3541.0	83.0	2.34%
2.8	95%	32	1/2	16	4078.0	0.0	0.0%

Fonte: Autor.

5.4 EXPERIMENTO 3

O terceiro experimento foi executado para observar a capacidade de aprendizado do agente ao longo do tempo. Nele foram executadas 5 simulações com os mesmos fatores dos experimentos anteriores, mas com valores diferentes (como demonstra a Tabela 10). Entre uma simulação e outra o agente não foi recarregado, ou seja, manteve o aprendizado das simulações anteriores.

Tabela 10 – Valor dos fatores no experimento 3.

Fator	Valor
PPI	50%
TMC	16
TMA	32
NPC	8

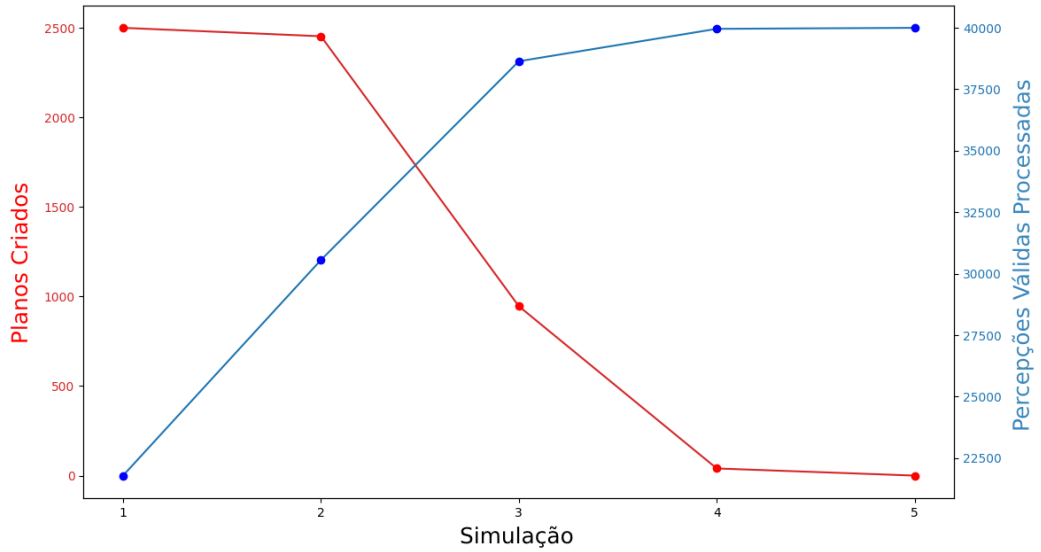
Fonte: Autor.

Tabela 11 – Resultados obtidos no experimento 3.

Simulação	Planos Criados	Percepções Válidas Processadas	Tempo Virtual
1	2501	21788	120016T
2	2454	30558	119264T
3	946	38641	95136T
4	40	39959	80640T
5	0	40000	80000T

Fonte: Autor.

Figura 12 – Evolução das percepções válidas processadas e dos planos novos criados ao longo das simulações.



Fonte: Autor.

Os resultados estão expostos na Tabela 11, mostrando o comportamento dos três fatores observados. Enquanto os planos criados diminuem a cada simulação, pois menos percepções são inválidas (uma vez que o agente aprende com as simulações anteriores), a quantidade de percepções válidas processadas aumenta. Esse comportamento é demonstrado na Figura 12. Além disso, o tempo virtual cai na mesma proporção que os planos criados diminuem, uma vez que a criação de um plano é duas vezes mais custosa em tempo do que o processamento de uma percepção válida nesse experimento.

5.5 ANÁLISE DO MODELO

O principal objetivo dos experimentos era validar as ideias principais que compõem a proposta do modelo HAIL. Como as simulações foram realizadas sem ambiente

e com um agente qualquer, os dados obtidos não podem ser usados como parâmetro de análise direta das qualidades do modelo, mas sim de seu funcionamento.

O primeiro experimento demonstrou que o módulo de ilusão e alucinação funciona como previsto: simulações com mais percepções inválidas e menor tempo médio de planejamento automatizado resultam em menos tempo virtual gasto e mais planos criados. Os grupos representados na Figura 11 demostram isso. As seis ocorrências do valor aproximado de 250 planos criados possuem PPI de 5%. A única simulação com esse PPI que possui uma quantidade de planos criados aproximada aos valores das simulações com PPI 95% possui TMA baixo e TMC e NPC altos, ou seja, muitos processos de planejamento automatizado são executados todos os ciclos, aumentando o número de planos criados. As demais simulações com esse PPI ou possuem poucas percepções por ciclo, ou o bloco avaliador não consegue enviar tantas anomalias para o planejamento automatizado devido aos valores de TMC e TMA.

Um dos objetivos do trabalho é criar um modelo capaz de aprender com as anomalias recebidas pelo agente através da percepção, e por isso o segundo e o terceiro experimentos abordam o mecanismo de aprendizado do HAIL. Com esses experimentos foi possível observar que o processo de seleção de anomalias que serão enviadas para o bloco de planejamento automatizado, desempenhado pelo bloco avaliador, é funcional.

O cenário ideal representado pelo experimento 3, onde todas as anomalias da primeira simulação se tornam parte do contexto do agente após poucas iterações, não tem como existir em uma situação real na qual a possibilidade de anomalias não é limitado por um gerador. Apesar disso, o experimento serve para demonstrar o funcionamento do módulo de ilusão e alucinação, pois os planos que são adicionados ao longo das simulações reduzem o tempo virtual e aumentam o número de percepções válidas processadas.

Com os resultados obtidos com os experimentos é possível afirmar que o modelo se comporta da maneira esperada, mas não é possível chegar a nenhuma conclusão relacionada à efetividade do modelo. Alguns cenários se mostraram favoráveis ao HAIL, como as simulações que possuem um grande volume de anomalias e baixo tempo de planejamento automatizado. Apesar disso, experimentos com cenários realistas e implementações acopladas a arquiteturas reais são necessários para validar que o modelo se comporta bem sob tais circunstâncias.

6 CONSIDERAÇÕES FINAIS

Agentes são entidades autônomas que, inseridas em um ambiente, são capazes de tomar decisões de maneira autônoma. Para se comunicar com o ambiente no qual estão inseridos, os agentes realizam o processo de percepção, utilizando sensores para receber diversos tipos de informações do mundo ao seu redor. Todavia, as percepções recebidas podem ser incorretas por diversos motivos.

Neste trabalho, foi apresentado um modelo de revisão de percepções, capaz de detectar anomalias entre as percepções, classificá-las e transformá-las em conhecimento útil para o agente. Esse modelo foi inspirado pelos conceitos de alucinação e ilusão: percepções possíveis mais deslocadas de seu contexto ou percepções parcialmente possíveis, respectivamente. Esses dois conceitos dão nome ao modelo, HAIL (*hallucination* e *illusion*). A base teórica de conceitos para a construção do HAIL está presente no Capítulo 2. Uma revisão simples do estado da arte está presente no Capítulo 3, no qual outros trabalhos, que abordam o problema das percepções de diferentes maneiras, foram apresentados.

O funcionamento do modelo foi descrito através de exemplos e formalizado no Capítulo 4. Ele é composto por um módulo de refinamento, responsável por reduzir as percepções iniciais recebidas através de um processo de filtragem, e um módulo de alucinação e ilusão, que classifica as anomalias através de decisores e as encaminha para blocos de avaliação. Nesses blocos, as anomalias são guardadas em listas ponderadas, e aguardam a sua vez de serem processadas. Quando o bloco libera uma anomalia, ela é enviada para um bloco de planejamento automatizado, onde será utilizada para gerar um novo plano para o agente.

No Capítulo 5 apresentamos uma implementação do modelo HAIL (em um ambiente sem contexto de aplicação) e os experimentos realizados para demonstrar que o funcionamento do modelo é como esperado, ou seja, que ele é capaz de categorizar anomalias e melhorar o funcionamento do agente com elas.

A partir da comparação inicial realizada com outros trabalhos que tentam resolver problemas de percepções inválidas, da definição formal do modelo e dos experimentos realizados, é possível identificar que o HAIL possui algumas características que definem quando ele é adequado:

- É um modelo genérico, que pode ser utilizado com diversos tipos de agentes e configurado de acordo com a necessidade do projeto.
- Pode adicionar aprendizado a qualquer arquitetura cognitiva, mesmo aquelas que não preveem esse tipo de mecanismo.
- Não necessita de uma base de conhecimentos prévia, tratando apenas das anomalias que são recebidas pelo módulo de alucinação e ilusão após o processo

de refinamento.

- Possui uma abordagem padrão totalmente simbólica.

6.1 TRABALHOS FUTUROS

Apesar dos experimentos terem resultados positivos, as simulações foram implementadas em um cenário genérico sem objetivo no mundo real. Por conta disso, quase não haviam fatores aleatórios, tornando a análise extremamente precisa (com erros da ordem de grandeza de e^{-12}) mas também bastante limitada. Portanto, é necessário validar o modelo através de duas etapas: acoplando-o a uma arquitetura cognitiva e submetendo-o a experimentos mais complexos, através de simulações ou implementação física com um problema real para ser resolvido.

Esses novos experimentos podem ser feitos aos moldes de experimentos de trabalhos similares, para que possa haver uma comparação direta com o HAIL. Os trabalhos apresentados no Capítulo 3, apesar de abordarem problemas similares ao que o HAIL se propõem a resolver, possuem domínios particulares que dificultam uma análise relacionando os resultados.

Além disso, o HAIL ainda pode ter cada um de seus componentes testados para validar se seu comportamento é o melhor possível, alterando os decisores do módulo de alucinação e ilusão, as equações do bloco avaliador, o método utilizado para armazenar as anomalias e o funcionamento do bloco de planejamento automatizado.

REFERÊNCIAS

ALOIMONOS, John; WEISS, Isaac; BANDYOPADHYAY, Amit. Active vision. **International Journal of Computer Vision**, v. 1, n. 4, p. 333–356, jan. 1988. ISSN 1573-1405. DOI: 10.1007/BF00133571. Disponível em: https://doi.org/10.1007/BF00133571.

BAJCSY, Ruzena; ALOIMONOS, Yiannis; TSOTSOS, John K. Revisiting active perception. **Autonomous Robots**, v. 42, n. 2, p. 177–196, fev. 2018. ISSN 1573-7527. DOI: 10.1007/s10514-017-9615-3. Disponível em: https://doi.org/10.1007/s10514-017-9615-3.

BELLMAN, Richard. **An introduction to artificial intelligence: Can computers think?** [*S.l.*]: Thomson Course Technology, 1978.

BORDEUX, Christophe; BOULIC, Ronan; THALMANN, Daniel. An efficient and flexible perception pipeline for autonomous agents. *In*: WILEY ONLINE LIBRARY, 3. COMPUTER Graphics Forum. [*S.l.*: *s.n.*], 1999. P. 23–30.

BOUTILIER, Craig; DEAN, Thomas L.; HANKS, Steve. Decision-Theoretic Planning: Structural Assumptions and Computational Leverage. **CoRR**, abs/1105.5460, 2011. arXiv: 1105.5460. Disponível em: http://arxiv.org/abs/1105.5460.

CASSANDRA, Anthony Rocco. **Exact and Approximate Algorithms for Partially Observable Markov Decision Processes**. 1998. Tese (Doutorado) – Providence, RI, USA. AAI9830418. ISBN 0-591-83322-0.

CHALMERS, David J; FRENCH, Robert M; HOFSTADTER, Douglas R. High-level perception, representation, and analogy: A critique of artificial intelligence methodology. **Journal of Experimental & Theoretical Artificial Intelligence**, Taylor & Francis, v. 4, n. 3, p. 185–211, 1992.

CHARNIAK, Eugene. **Introduction to artificial intelligence**. [*S.l.*]: Pearson Education India, 1985.

CHRISMAN, Lonnie; CARUANA, Rich; CARRIKER, Wayne. Intelligent agent design issues: Internal agent state and incomplete perception. *In*: CITESEER. PROCEEDINGS of the AAAI Fall Symposium on Sensory Aspects of Robotic Intelligence. AAAI Press/MIT Press. [*S.l.*: *s.n.*], 1991.

COLTON, Simon; WIGGINS, Geraint A *et al.* Computational creativity: The final frontier? *In*: MONTPELIER. ECAI. [*S.l.*: *s.n.*], 2012. P. 21–26.

CRANE, Tim; FRENCH, Craig. The Problem of Perception. *In*: ZALTA, Edward N. (Ed.). **The Stanford Encyclopedia of Philosophy**. Spring 2017. [*S.l.*]: Metaphysics Research Lab, Stanford University, 2017.

DIAB, Mohammed; AKBARI, Aliakbar; UD DIN, Muhayy; ROSELL, Jan. PMK—A Knowledge Processing Framework for Autonomous Robotics Perception and Manipulation. **Sensors**, MDPI AG, v. 19, n. 5, p. 1166, mar. 2019. ISSN 1424-8220. DOI: 10.3390/s19051166. Disponível em: http://dx.doi.org/10.3390/s19051166.

DYACHENKO, Yuriy; NENKOV, Nayden; PETROVA, Mariana; SKARGA-BANDUROVA, Inna; SOLOVIOV, Oleg. Approaches to cognitive architecture of autonomous intelligent agent. **Biologically Inspired Cognitive Architectures**, v. 26, p. 130–135, 2018. ISSN 2212-683X. DOI: https://doi.org/10.1016/j.bica.2018.10.004. Disponível em: http://www.sciencedirect.com/science/article/pii/S2212683X18301300.

GHALLAB, Malik; NAU, Dana; TRAVERSO, Paolo. Chapter 1 - Introduction and Overview. *In*: GHALLAB, Malik; NAU, Dana; TRAVERSO, Paolo (Ed.). **Automated Planning**. Burlington: Morgan Kaufmann, 2004. (The Morgan Kaufmann Series in Artificial Intelligence). P. 1–16. ISBN 978-1-55860-856-6. DOI: https://doi.org/10.1016/B978-155860856-6/50004-1. Disponível em: http://www.sciencedirect.com/science/article/pii/B9781558608566500041.

GIBSON, James J. The perception of the visual world. Houghton Mifflin, 1950.

HAYES-ROTH, Barbara; WASHINGTON, Richard; ASH, David; HEWETT, Rattikorn; COLLINOT, Anne; VINA, Angel; SEIVER, Adam. Guardian: A prototype intelligent agent for intensive-care monitoring. **Artificial Intelligence in Medicine**, Elsevier, v. 4, n. 2, p. 165–185, 1992.

HAYES-ROTH, Frederick; WATERMAN, Donald A; LENAT, Douglas B. Building expert system. CumInCad, 1983.

HUNTER, J. D. Matplotlib: A 2D graphics environment. **Computing in Science & Engineering**, IEEE COMPUTER SOC, v. 9, n. 3, p. 90–95, 2007. DOI: 10.1109/MCSE.2007.55.

JAIN, Raj. **The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling**. [*S.l.*]: John Wiley & Sons, 1990.

JANSSEN, Marco A. Agent-based modelling. **Modelling in ecological economics**, p. 155–172, 2005.

KIM, Sangwook; YU, Zhibin; LEE, Minho. Understanding human intention by connecting perception and action learning in artificial agents. **Neural Networks**, Elsevier, v. 92, p. 29–38, 2017.

KURZWEIL, Ray. **The age of spiritual machines: When computers exceed human intelligence**. [*S.l.*]: Penguin, 2000.

KUSHMERICK, Nicholas; HANKS, Steve; WELD, Daniel S. An algorithm for probabilistic planning. **Artificial Intelligence**, v. 76, n. 1, p. 239–286, 1995. Planning and Scheduling. ISSN 0004-3702. DOI: https://doi.org/10.1016/0004-3702(94)00087-H. Disponível em: http://www.sciencedirect.com/science/article/pii/000437029400087H.

LANGLEY, Pat; LAIRD, John E; ROGERS, Seth. Cognitive architectures: Research issues and challenges. **Cognitive Systems Research**, Elsevier, v. 10, n. 2, p. 141–160, 2009.

LITTMAN, Michael. Algorithms for Sequential Decision Making, ago. 2009.

LUGER, George F. **Artificial Intelligence: Structures and Strategies for Complex Problem Solving**. 6th. USA: Addison-Wesley Publishing Company, 2008. ISBN 0321545893.

MADANI, Omid; HANKS, Steve; CONDON, Anne. On the undecidability of probabilistic planning and related stochastic optimization problems. **Artificial Intelligence**, v. 147, n. 1, p. 5–34, 2003. Planning with Uncertainty and Incomplete Information. ISSN 0004-3702. DOI: https://doi.org/10.1016/S0004-3702(02)00378-8. Disponível em: http://www.sciencedirect.com/science/article/pii/S0004370202003788.

MOOR, James. The Dartmouth College artificial intelligence conference: The next fifty years. **Ai Magazine**, v. 27, n. 4, p. 87–87, 2006.

MOYA, Lisa Jean; TOLK, Andreas. Towards a taxonomy of agents and multi-agent systems. *In*: SPRINGSIM (2). [*S.l.*: *s.n.*], 2007. P. 11–18.

OIJEN, Joost van; DIGNUM, Frank. Scalable perception for bdi-agents embodied in virtual environments. *In*: IEEE. 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology. [*S.l.*: *s.n.*], 2011. P. 46–53.

PANGERCIC, D.; TENORTH, M.; JAIN, D.; BEETZ, M. Combining perception and knowledge processing for everyday manipulation. *In*: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems. [*S.l.*: *s.n.*], 2010. P. 1065–1071.

POOLE, David; MACKWORTH, Alan; GOEBEL, Randy. Computational Intelligence, 1998.

PRICE, H. H. Perception. **Journal of Philosophy**, Journal of Philosophy, v. 30, n. 12, p. 330–334, 1933.

REBACK, Jeff *et al.* **pandas-dev/pandas: Pandas**. [*S.l.*]: Zenodo, fev. 2020. DOI: 10.5281/zenodo.3509134. Disponível em: https://doi.org/10.5281/zenodo.3509134.

RICH, E.; KNIGHT, K. **Artificial Intelligence**. [*S.l.*]: McGraw-Hill, 1991. (Artificial Intelligence Series). ISBN 9780071008945. Disponível em: https://books.google.com.br/books?id=6P6jPwAACAAJ.

RUSSEL, Stuart; NORVIG, Peter *et al.* **Artificial intelligence: a modern approach**. [*S.l.*]: Pearson Education Limited, 2013.

RUSSELL, Bertrand. **The Problems of Philosophy**. [*S.l.*]: Home University Library, 1912.

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

SMITH, A. D. **The Problem of Perception**. [*S.l.*]: Harvard University Press, 2002.

ŚWIECICKI, Tomek. **RandomWords · PyPI**. [*S.l.*: *s.n.*], 2012. https://pypi.org/project/RandomWords/. Acesso 21/07/2020, versão 0.3.0.

WASKOM, Michael Lawrence. seaborn: statistical data visualization. **Journal of Open Source Software**, v. 60, n. 6, abr. 2021. DOI: 10.21105/joss.03021. Disponível em: https://joss.theoj.org/papers/10.21105/joss.03021.

WEYNS, DANNY; STEEGMANS, ELKE; HOLVOET, TOM. TOWARDS ACTIVE PERCEPTION IN SITUATED MULTI-AGENT SYSTEMS. **Applied Artificial Intelligence**, Taylor & Francis, v. 18, n. 9-10, p. 867–883, 2004. DOI: 10.1080/08839510490509063. eprint: https://doi.org/10.1080/08839510490509063. Disponível em: https://doi.org/10.1080/08839510490509063.

WEYNS, Danny; STEEGMANS, Elke; HOLVOET, Tom. Towards active perception in situated multi-agent systems. **Applied Artificial Intelligence**, Taylor & Francis, v. 18, n. 9-10, p. 867–883, 2004.

WOOLDRIDGE, Michael. Intelligent agents. **Multiagent systems**, MIT Press London, v. 35, n. 4, p. 51, 1999.

WOOLDRIDGE, Michael J; JENNINGS, Nicholas R. Intelligent agents: Theory and practice. **The knowledge engineering review**, v. 10, n. 2, p. 115–152, 1995.

YE, P.; WANG, T.; WANG, F. A Survey of Cognitive Architectures in the Past 20 Years. **IEEE Transactions on Cybernetics**, v. 48, n. 12, p. 3280–3290, 2018. DOI: 10.1109/TCYB.2018.2857704.

APÊNDICE A – CÓDIGO DO SIMULADOR

O simulador foi implementado utilizando a versão 3.7 do Python, e o gerenciamento de pacotes foi feito com o Poetry. As dependências estão disponíveis no arquivo `pyproject.toml`.

A.1 __MAIN__.PY

```
1  from simulation import Simulation
2  from analyzer import get_mean
3  from generator.perceptions import PerceptionGenerator
4
5  import strings
6  from time import time
7  import pandas as pd
8  from shutil import copyfile
9  import click
10 import os
11
12 from proggy import BarInfo
13 from proggy.tty import TTYProgressBar
14
15
16 # Click was used in testing and first simulations, but was replaced with
17 # multiple function calls. This way, it is possible to run the program once
18 # for all 16 factors variation, instead of passing the arguments through console
19 # every time one simulation ends. Uncommenting this decorators and the first line
20 # from main enable console usage again.
21
22 # @click.command()
23 # @click.option("--generate", "-G", default=0, nargs=2, help=strings.generate_help)
24 # @click.option(
25 #     "--reload-agent/--not-reload", "-R/", default=True, help=strings.reload_help
26 # )
27 # @click.option("--reasoning-time", default=1.0, help=strings.reasoning_time_help)
28 # @click.option("--planning-time", default=32.0, help=strings.planning_time_help)
29 # @click.option(
30 #     "--perceptions-per-cycle", "-C", default=1,
   ↪  help=strings.perceptions_per_cycle_help
31 # )
32 # @click.option("--iterations", "-I", default=1, help=strings.iterations_help)
33 def run(
34     generate,
35     reload_agent,
36     reasoning_time,
37     planning_time,
38     perceptions_per_cycle,
```

```
39      iterations,
40  ):
41
42      start_time = time()
43      vtimes = []
44      pps = []
45      pcs = []
46      name_string =
        ↪ f"valid{generate[1]}reasoning{reasoning_time}planning{planning_time}" +
        ↪ "perceptions{perceptions_per_cycle}reload{str(reload_agent)}"
47
48      with TTYProgressBar(BarInfo(size=30, total=iterations)) as p:
49
50          for i in range(iterations):
51              # print(f"-------------\nSIMULATION {i}\n-------------")
52
53              if generate:
54                  g = PerceptionGenerator(*generate, perceptions_per_cycle)
55                  g.generate()
56                  if not os.path.exists(f'./data3/perceptions/{name_string}'):
57                      os.makedirs(f'./data3/perceptions/{name_string}')
58                  copyfile("perceptions.txt",
                    ↪ f"./data3/perceptions/{name_string}/iteration{i}.txt")
59
60              s = Simulation(
61                  reasoning_time,
62                  planning_time,
63                  perceptions_per_cycle,
64                  reload_agent=reload_agent,
65              )
66              vtime, perceptions_processed, plans_created = s.start()
67              vtimes.append(vtime)
68              pps.append(perceptions_processed)
69              pcs.append(plans_created)
70
71              if not os.path.exists(f'./data3/agents/{name_string}'):
72                      os.makedirs(f'./data3/agents/{name_string}')
73              copyfile("agent.txt", f"./data3/agents/{name_string}/iteration{i}.txt")
74
75              p.progress += 1
76
77      final_time = time()
78      total_time = final_time - start_time
79      total_time = "{:.2f}".format(total_time)
80      print(
81          f"------------\nFinished Simulation\nTime elapsed:
            ↪ {total_time}s\n------------"
82      )
83
```

```
84      d = {"vtime": vtimes, "perceptions_processed": pps, "plans_created": pcs}
85      df = pd.DataFrame(data=d)
86
87      df.to_csv(f'./data3/results/{name_string}.csv', index=False)
88
89
90  if __name__ == "__main__":
91      # uncomment run() if using click
92      # run()
93      run((5000, 50), True, 16, 32, 8, 1)
94      run((5000, 50), False, 16, 32, 8, 1)
95      run((5000, 50), False, 16, 32, 8, 1)
96      run((5000, 50), False, 16, 32, 8, 1)
97      run((5000, 50), False, 16, 32, 8, 1)
```

A.2 BASE-AGENT.TXT

```
1 square(smooth) -> pick(); wrap_up(red);
2 square(striped) -> pick(); wrap_up(red);
3 circle(smooth) -> pick(); wrap_up(red);
4 circle(striped) -> pick(); wrap_up(blue);
5 triangle(smooth) -> pick(); wrap_up(blue);
```

A.3 GENERATOR/PERCEPTIONS.PY

```python
from .perceptions_options import (
    VALID_POSSIBLE_PERCEPTIONS_BODIES,
    VALID_POSSIBLE_PERCEPTIONS_ARGS,
    INVALID,
)

from random_words import RandomWords
from random import choice, shuffle
from time import time


class PerceptionGenerator:
    def __init__(self, cycles, invalid_perceptions_percentage, perceptions_per_line):
        self.cycles = cycles
        self.invalid_p = invalid_perceptions_percentage
        self.valid_p = 100 - invalid_perceptions_percentage
        self.perceptions_per_line = perceptions_per_line

    def create_valid_perception(self):
        body = choice(VALID_POSSIBLE_PERCEPTIONS_BODIES)
        arg = choice(VALID_POSSIBLE_PERCEPTIONS_ARGS)
        return f"{body}({arg})"

    def generate(self):
        start_time = time()

        valid = []
        invalid = []

        valid_cycles = int(self.valid_p * self.cycles / 100)
        invalid_cycles = int(self.invalid_p * self.cycles / 100)

        random_words = RandomWords()
        random_bodies = []
        random_args = []

        # This module only allows requesting 5450 words at once.
        # In some cases, we need 76000 words, so we need to do it "manually"
        for i in range(self.perceptions_per_line):
            bodies = random_words.random_words(count=invalid_cycles)
            args = random_words.random_words(count=invalid_cycles)
            random_bodies = random_bodies + bodies
            random_args = random_args + args

        for i in range(valid_cycles):
            # Generate perceptions for line
```

```
47              perception_line = []
48              for j in range(self.perceptions_per_line):
49
50                  perception = self.create_valid_perception()
51                  while perception in INVALID:
52                      perception = self.create_valid_perception()
53
54                  perception_line.append(perception)
55
56              # Concatenate perceptions to create line
57              final_perception = ""
58              for perception in perception_line:
59                  final_perception = final_perception + perception + ","
60
61              # Add final perception string without the last char
62              valid.append(final_perception[:-1])
63
64          for i in range(invalid_cycles):
65              # Generate perceptions for line
66              perception_line = []
67              for j in range(self.perceptions_per_line):
68                  body = random_bodies.pop(0).lower()
69                  arg = random_args.pop(0).lower()
70
71                  perception = f"{body}({arg})"
72                  perception_line.append(perception)
73
74              # Concatenate perceptions to create line
75              final_perception = ""
76              for perception in perception_line:
77                  final_perception = final_perception + perception + ","
78
79              # Add final perception string without the last char
80              invalid.append(final_perception[:-1])
81
82          perceptions = valid + invalid
83          shuffle(perceptions)
84
85          file = open("perceptions.txt", "w")
86
87          for line in perceptions:
88              file.write(line)
89              file.write("\n")
90
91          file.close()
92
93          final_time = time() - start_time
94          # print(f"{len(perceptions)} perceptions cycles generated in {final_time}s")
```

A.4 GENERATOR/PERCEPTIONS_OPTIONS.PY

```
1 # TODO: Get this configs directly from the agent file
2 VALID_POSSIBLE_PERCEPTIONS_BODIES = ["circle", "square", "triangle", "star"]
3
4 VALID_POSSIBLE_PERCEPTIONS_ARGS = ["striped", "smooth"]
5
6 # Invalid perception formed by valid bodies and args
7 INVALID = ["triangle(striped)"]
```

A.5 SIMULATION.PY

```
from pr_system import PerceptionRevision
from shutil import copyfile


class Simulation:
    def __init__(
        self,
        reasoning_at,
        autoplanning_at,
        perceptions_per_cycle,
        reload_agent=True,
        debug=False,
    ):
        self.perception_queue = []
        self.perceptions_per_cycle = perceptions_per_cycle

        perceptions = open("perceptions.txt", "r")

        content = perceptions.readlines()
        for c in content:
            splited = c.split(",")
            for s in splited:
                self.perception_queue.append(s.replace("\n", ""))

        perceptions.close()

        # A copy of the base agent is created because the autoplanner
        # will change the file.
        if reload_agent:
            copyfile("base-agent.txt", "agent.txt")

        self.model = PerceptionRevision("agent.txt", reasoning_at, autoplanning_at)
        self.debug = debug

    def start(self):
        vtime = 0
        perceptions_processed = 0

        perceptions_number = self.perceptions_per_cycle

        while self.perception_queue:
            if perceptions_number > len(self.perception_queue):
                perceptions_number = len(self.perception_queue)

            perceptions = [
                self.perception_queue.pop(0) for i in range(perceptions_number)
```

```
47              ]
48
49              (_vtime, pp) = self.model.process_perceptions(perceptions)
50              vtime = vtime + _vtime
51              perceptions_processed = perceptions_processed + pp
52
53          if self.debug:
54              print(f"vtime: {vtime}\nperceptions_processed: {perceptions_processed}")
55              print(f"{self.model.autoplanner.plans_created} new plans created")
56
57          return (vtime, perceptions_processed, self.model.autoplanner.plans_created)
```

A.6 PR_SYSTEM.PY

```
"""Perception Revision System implementation.
This module implements all the formal model described in the
illusion and hallucination paper. It can be attached to any
cognitive architecture to reason about the perceptions coming
from the environment.
"""

from utils import (
    parse_agent_plans,
    get_perceptions_actions,
    get_agent_context,
    parse_perception,
)

from structures import AvaliationBlock, Autoplanner

from typing import List, Tuple
from random import choice


class PerceptionRevision:
    def __init__(self, agent, reasoning_at, autoplanning_at):
        self.agent = agent
        self.plans = parse_agent_plans(self.agent)
        self.actions = get_perceptions_actions(self.plans)

        # Now, the context only includes terms from the left side of the plan.
        # Verify if the model uses this definition!
        self.context_bodies, self.context_args = get_agent_context(self.plans)
        self.context = self.context_bodies + self.context_args

        self.illusion1_AB = AvaliationBlock(reasoning_at, autoplanning_at)
        self.illusion2_AB = AvaliationBlock(reasoning_at, autoplanning_at)
        self.hallucination_AB = AvaliationBlock(reasoning_at, autoplanning_at)
        self.reasoning_at = reasoning_at

        # This could be replaced with a set, but random.choice
        # gives the error "'set' object is not subscriptable"
        self.avaliation_blocks = []

        self.MAP_PERCEPTION_TO_AB = {
            "illusion1": self.illusion1_AB,
            "illusion2": self.illusion2_AB,
            "hallucination": self.hallucination_AB,
        }

```

```
47          self.autoplanner = Autoplanner(self.actions, self.context, agent)
48
49      def __update_context(self):
50          self.plans = parse_agent_plans(self.agent)
51          self.context_bodies, self.context_args = get_agent_context(self.plans)
52          self.context = self.context_bodies + self.context_args
53
54      def __classify_perception(self, perception: str) -> str:
55          """Classify if a perception is an illusion or hallucination.
56          Args:
57              perception: The perception string in the format p(x).
58          Returns:
59              An int, 1 for illusion class 1, 2 for illusion class 2 and 3 for
    ↪  hallucination.
60          """
61          body, arg = parse_perception(perception)
62
63          if body in self.context and arg in self.context:
64              return "valid"
65
66          if body in self.context:
67              return "illusion1"
68
69          if arg in self.context:
70              return "illusion2"
71
72          return "hallucination"
73
74      def process_perceptions(self, perceptions: List[int]) -> Tuple[int, int]:
75          # Return = (vtime, perceptions_processed)
76
77          have_anomaly = False
78          anomalies = 0
79          # Add each perception to it's respective avaliation block
80          for perception in perceptions:
81              perception_type = self.__classify_perception(perception)
82
83              if perception_type in self.MAP_PERCEPTION_TO_AB:
84
85                  ab = self.MAP_PERCEPTION_TO_AB[perception_type]
86                  ab.list.push(perception)
87
88                  if ab not in self.avaliation_blocks:
89                      self.avaliation_blocks.append(ab)
90
91                  have_anomaly = True
92                  anomalies = anomalies + 1
93
94          if not have_anomaly:
```

```
95              return (self.reasoning_at, len(perceptions))
96
97          vtime = 0
98          keep_planning = True
99
100         while keep_planning:
101             if self.avaliation_blocks:
102                 avaliation_block = choice(self.avaliation_blocks)
103                 (vtime, keep_planning) = avaliation_block.evaluate(vtime)
104
105                 if keep_planning:
106                     plan_perception = avaliation_block.list.pop()
107                     self.autoplanner.plan(plan_perception)
108
109                 if avaliation_block.list.is_empty():
110                     self.avaliation_blocks.remove(avaliation_block)
111             else:
112                 keep_planning = False
113
114         self.__update_context()
115
116         return (vtime, len(perceptions) - anomalies)
```

A.7 STRUCTURES.PY

```
1  from typing import Tuple, TypeVar
2  from random import choice, randint
3
4  T = TypeVar("T")
5
6
7  class OrderedList:
8      def __init__(self):
9          self.__list = {}
10
11     def pop(self) -> T:
12         """Get the value that was inserted more times"""
13         first = True
14         top = None
15         for element in self.__list:
16             if first:
17                 top = element
18                 first = False
19
20             elif self.__list[element] > self.__list[element]:
21                 top = element
22
23         self.__list.pop(top, None)
24         return top
25
26     def push(self, element: T):
27         """Insert an element on the ordered list.
28         If it's the first time that the element is inserted,
29         it's value is 1. If it's not, it's value will be increased
30         by 1.
31
32         Args:
33             element: The element to be inserted in the list.
34         """
35         if element in self.__list:
36             old_value = self.__list[element]
37             self.__list[element] = old_value + 1
38         else:
39             self.__list[element] = 1
40
41     def clean(self):
42         """Clean list to free memory.
43
44         Let S1 be the sum of 'weights' (number of times an element was inserted)
45         of the elements that the weight is bigger than 1.
46         Let S2 be the sum of 'weights' of the elements that hava weight 1.
```

```
47          If S1 is bigger than S2, remove all the elements in the list that have
48          'weight' 1.
49          """
50          S1 = 0
51          S2 = 0
52          for element in self.__list:
53              weight = self.__list[element]
54              if weight > 1:
55                  S1 = S1 + weight
56              if weight == 1:
57                  S2 = S2 + weight
58
59          if S1 > S2:
60              new_list = {}
61
62              for element in self.__list:
63                  if self.__list[element] > 1:
64                      new_list[element] = self.__list[element]
65
66              self.__list = new_list
67
68      def is_empty(self):
69          return not bool(self.__list)
70
71
72  class AvaliationBlock:
73      """Trivial implementation of the Avaliation Block.
74      In this case, the Avaliation Block is just the ordered list. In other
75      scenarios, the process of choosing which perception to treat could be
76      more complex, or the information from the Ordered List could be passed
77      to other elements in the model or in the agent architecture.
78      """
79
80      # at = average time
81      def __init__(self, reasoning_at: int, autoplanning_at: int):
82          self.list = OrderedList()
83          self.reasoning_at = reasoning_at
84          self.autoplanning_at = autoplanning_at
85
86      def evaluate(self, vtime: int) -> Tuple[int, bool]:
87          """Define if a perception in the Ordered List can be processed by the
            ↪  autoplanner."""
88          if vtime < self.reasoning_at:
89              vtime = vtime + self.autoplanning_at
90              return (vtime, True)
91          else:
92              return (vtime, False)
93
94
```

```
95  class Autoplanner:
96      """Naïve autoplanning implementation.
97      In this implementation, we just randomly pick agent's action and
98      combine with the perceptions choosed to create a new plan.
99      """
100
101     def __init__(self, actions, context, agent):
102         self.actions = actions
103         self.context = context
104         self.agent = agent
105         self.plans_created = 0
106
107     def plan(self, perception):
108
109         actions_number = randint(1, len(self.actions))
110
111         actions = []
112
113         for i in range(actions_number):
114             action = choice(self.actions)
115             if action not in actions:
116                 actions.append(action)
117
118         new_actions = []
119
120         for action in actions:
121             arg = choice(self.context)
122             new_actions.append(f"{action}({arg})")
123
124         new_plan = f"{perception} ->"
125
126         for action in new_actions:
127             new_plan = new_plan + " " + action + ";"
128
129         agent_file = open(self.agent, "a")
130
131         agent_file.write("\n" + new_plan)
132
133         agent_file.close()
134
135         self.plans_created = self.plans_created + 1
```

A.8 UTILS.PY

```python
from typing import List, Tuple, Union


def parse_agent_plans(agent: str) -> List[str]:
    """Parse agent file to get it's plans.
    Args:
        agent: Agent file name. Must be on the root or contain the path.
    Returns:
        A list of the agent's plans.
    """
    with open(agent) as f:
        plans = f.read().splitlines()

    return plans


def get_perceptions_actions(plans: List[str]) -> List[str]:
    """Get all actions from given plans.
    This implementations parse an specific kind of plans, i.e. different
    agent oriented programming languages need different implementations of this
    function.
    Args:
        plans: List of plans. Can be get from parse_agent_plans().
    Returns:
        A list of all actions used in the given plans.
    """
    actions = []
    for plan in plans:

        plan_body = plan.split("->")[1]
        plan_actions = plan_body.split(";")

        for action in plan_actions:
            if "(" in action:
                action = action.split("(")[0]

            action = action.replace(" ", "")

            if action not in actions:
                actions.append(action)

    actions.remove("")
    return actions


def get_agent_context(plans: List[str]) -> Tuple[List[str], List[str]]:
```

```
47      """Get the agent context (the domain of the perception function)
48
49      Args:
50          plans: List of plans. Can be get from parse_agent_plans().
51      Returns:
52          The agent's context, a 2-tuple of bodies and args.
53      """
54      bodies = []
55      args = []
56      for plan in plans:
57          plan_head = plan.split("->")[0].replace(" ", "")
58
59          body, arg = parse_perception(plan_head)
60
61          if body not in bodies:
62              bodies.append(body)
63
64          if arg not in args:
65              args.append(arg)
66
67      return (bodies, args)
68
69
70  def parse_perception(perception: str) -> Tuple[str, Union[str, None]]:
71      """Get the perception body and arg.
72      Args:
73          perception: Perception or plan head in the format p(x).
74      Returns:
75          A 2-tuple of body and arg.
76      """
77      if "(" in perception:
78          body, arg = perception.split("(")
79          arg = arg.replace(")", "")
80          return (body, arg)
81
82      return (perception, None)
```

A.9 ANALYZER.PY

```
from statistics import mean


def get_mean():
    results = open("results.txt", "r")

    content = results.read().split(";")

    vtimes = []
    perceptions_processed = []
    plans_created = []

    for c in content:
        try:
            (vtime, perceptions, plans) = c.split(",")
            vtimes.append(float(vtime))
            perceptions_processed.append(float(perceptions))
            plans_created.append(int(plans))
        except ValueError:
            pass

    vt_mean = mean(vtimes)
    vt_string = "{:.2f}".format(vt_mean)
    print(f"VTIME: {vt_string}")

    pp_mean = mean(perceptions_processed)
    pp_string = "{:.2f}".format(pp_mean)
    print(f"PERCEPTIONS PROCESSED: {pp_string}")

    print(f"PLANS CREATED: {mean(plans_created)}")
```

A.10 STRINGS.PY

```
1  # Only used by Click
2  generate_help = "Generate perceptions. Require 2 args: total number of perceptions and
   ↪  percentage of invalid perceptions, in this order. Exemple: --generate 500 80"
3  reload_help = "Reload or not current agent file. If true, will overwrite agent.txt
   ↪  with base-agent.txt file, removing plans created by the autoplanner."
4  reasoning_time_help = (
5      "Define agent's avarege reasoning time. Default is 1, but any value can be used."
6  )
7  planning_time_help = (
8      "Define agent's autoplanner avarege time. Default is 32, but any value can be
       ↪  used."
9  )
10 perceptions_per_cycle_help = (
11     "Define the number of perceptions received by the agent each cycle."
12 )
13 iterations_help = "Number of simulations that will be executed."
```

A.11 PYPROJECT.TOML

```
[tool.poetry]
name = "perception-simulation"
version = "0.1.0"
description = ""
authors = ["kundlatsch <gustavo.kundlatsch@gmail.com>"]

[tool.poetry.dependencies]
python = "^3.7"
RandomWords = "^0.2.1"
click = "^7.1.1"
proggy = "^0.4.3"
pandas = "^1.0.3"

[tool.poetry.dev-dependencies]

[build-system]
requires = ["poetry>=0.12"]
build-backend = "poetry.masonry.api"
```

APÊNDICE B – CÓDIGO DO ANALISADOR

O simulador foi implementado utilizando a versão 3.7 do Python, e o gerenciamento de pacotes foi feito com o Poetry. As dependências estão disponíveis no arquivo `pyproject.toml`.

B.1 __MAIN__.PY

```
from factorial2k import *
from data import RESULTS_PLANS_CREATED
import seaborn as sb
import pandas as pd
import matplotlib.pyplot as plt
from statistics import mean


def normalize(values):
    vmin = min(values)
    vmax = max(values)
    return [(v - vmin) / (vmax - vmin) for v in values]


def main():
    FACTORS_2KN = 4
    results = RESULTS_PLANS_CREATED
    EFFECTS_2KN = effects_table_method(FACTORS_2KN, results)

    sb.set(style="darkgrid")

    # 1 - Independent Errors
    x = normalize(get_residuals(results, EFFECTS_2KN, FACTORS_2KN))
    y = normalize(nonlinear_regression(EFFECTS_2KN, FACTORS_2KN)) #y_hat
    plt.plot(x, y, 'bo')
    plt.xlabel('Residuals', fontsize=10)
    plt.ylabel('Predicted Value', fontsize=10)
    plt.show()

    # 2 - Normally Distributed Errors
    r = []
    for re in results:
        r.append(re[0])
    x = normalize(r)
    y = normalize(nonlinear_regression(EFFECTS_2KN, FACTORS_2KN)) #y_hat
    plt.plot([0,1], 'r')
    plt.plot(x, y, 'bo')
    plt.xlabel('Simulation Value', fontsize=10)
    plt.ylabel('Predicted Value', fontsize=10)
```

```
40      plt.show()
41
42      # 3 - Constant Standard Deviation of Errors
43      x = [i for i in range(2 ** FACTORS_2KN)]
44      y = normalize(nonlinear_regression(EFFECTS_2KN, FACTORS_2KN))  # y_hat
45      m = mean(y)
46      y2 = [m for i in range(2 ** FACTORS_2KN)]
47      plt.plot(x, y, "bo")
48      plt.plot(x, y2, "r--")
49      plt.ylabel("Predicted Value", fontsize=10)
50      plt.show()
51
52
53  if __name__ == "__main__":
54      main()
```

B.2 FACTORIAL2K.PY

```
import pyDOE2 as pd
import numpy as np
from itertools import combinations
import string


def create_sign_table(factors):
    # This combination string is used to create the factorial_string,
    # because pyDOE2 input is the comlumns that we want to create in
    # the signal table. Here we are using it to generate the 2k^N table,
    # so we need the combination of N comlumns.

    combination_string = ""
    for _, letter in zip(range(0, factors), string.ascii_lowercase):
        combination_string += letter

    _combinations = [
        "".join(l)
        for i in range(len(combination_string))
        for l in combinations(combination_string, i + 1)
    ]

    factors_string = " ".join(_combinations)
    factorial_columns = pd.fracfact(factors_string)

    image_column = np.ones((2 ** factors, 1))
    sign_table = np.hstack((image_column, factorial_columns))
    return sign_table


def effects_table_method(factors, results):
    results_column = np.array(results)
    sign_table = create_sign_table(factors)

    tpf = 2 ** factors
    coefficients = [
        float(np.dot(sign_table[:, i], results_column) / tpf) for i in range(tpf)
    ]

    return coefficients


def variation_allocation(effects):
    # The first elements represents q0, and is not important here
    effects.pop(0)

```

```
47      allocation = [4 * effect * effect for effect in effects]
48      total_variation = sum(allocation)
49
50      percentages = [(100 * a) / total_variation for a in allocation]
51
52      return percentages
53
54
55  def get_residuals(results, effects, factors):
56      predicted_results = nonlinear_regression(effects, factors)
57      return [float(predicted_results[i] - results[i]) for i in range(len(results))]
58
59
60  def nonlinear_regression(coefficients, factors):
61      sign_table = create_sign_table(factors)
62      return [regression_eq(coefficients, r) for r in sign_table]
63
64
65  def regression_eq(coefficients, independent_values):
66      map(lambda x: np.array(x), (independent_values, coefficients))
67      return sum(np.multiply(coefficients, independent_values))
```

B.3 DATA.PY

```
# Model
# RESULTS = [[1], [9], [5], [13], [3], [11], [7], [15], [2], [10], [6], [14], [4], [12], [8], [16]]
# This are the corresponding indexes in the results table.
# TODO: Get results automatically from the simulation files

# Vtime
RESULTS_VTIME = [
    [4874.60],
    [3226.35],
    [152096.65],
    [48257.85],
    [20800.40],
    [228349.49],
    [168025.60],
    [273475.84],
    [4999.93],
    [5000.00],
    [153436.30],
    [140958.00],
    [20813.00],
    [304313.00],
    [168032.00],
    [312032.00],
]

# Perceptions Processed
RESULTS_PECEPTIONS_PROCESSED = [
    [4749.20],
    [1452.70],
    [4749.10],
    [1452.63],
    [4749.20],
    [1454.77],
    [4749.20],
    [1453.88],
    [4751.26],
    [2606.74],
    [4816.31],
    [4745.69],
    [4750.14],
    [1295.42],
    [4750.03],
    [1311.50],
]

```

```
46 # Plans Created
47 RESULTS_PLANS_CREATED = [
48     [250.80],
49     [3547.3],
50     [250.9],
51     [3547.37],
52     [250.8],
53     [3545.23],
54     [250.8],
55     [3546.12],
56     [502],
57     [9304],
58     [2936.6],
59     [4066.4],
60     [251],
61     [4751],
62     [251],
63     [4751],
64 ]
65
66
67 # Test
68 RESULTS_2K2 = [[15], [45], [25], [75]]
```

B.4 TESTS/TEST_FACTORIAL2K.PY

```python
from pytest import approx
from src.factorial2k import (
    effects_table_method,
    variation_allocation,
    nonlinear_regression,
)


# 2K^2 FACTORIAL DESIGN TESTS #
FACTORS_2K2 = 2
RESULTS_2K2 = [[15], [45], [25], [75]]
EFFECTS_2K2 = effects_table_method(FACTORS_2K2, RESULTS_2K2)
ALLOCATION_2K2 = variation_allocation(EFFECTS_2K2.copy())


def test_effects2k2():
    assert EFFECTS_2K2 == [40.0, 20.0, 10.0, 5.0]


def test_variation2k2():
    assert ALLOCATION_2K2 == approx([76, 19, 5], rel=1e-1, abs=1)


# 2K^n FACTORIAL DESIGN TESTS #
FACTORS_2KN = 3
RESULTS_2KN = [[14], [22], [10], [34], [46], [58], [50], [86]]
EFFECTS_2KN = effects_table_method(FACTORS_2KN, RESULTS_2KN)
ALLOCATION_2KN = variation_allocation(EFFECTS_2KN.copy())


def test_effects2kN():
    assert EFFECTS_2KN == [40.0, 10.0, 5.0, 20.0, 5.0, 2.0, 3.0, 1.0]


def test_variation2kN():
    assert ALLOCATION_2KN == approx([18, 4, 71, 4, 1, 2, 0], rel=1e-1, abs=1)


# VERIFYING THE ASSUMPTIONS


def test_nonlinear_regression():
    assert nonlinear_regression(EFFECTS_2K2, FACTORS_2K2) == [15.0, 45.0, 25.0, 75.0]
```

B.5 PYPROJECT.TOML

```
[tool.poetry]
name = "factorial-design"
version = "0.1.0"
description = ""
authors = ["kundlatsch <gustavo.kundlatsch@gmail.com>"]

[tool.poetry.dependencies]
python = "^3.7"
pyDOE2 = "^1.3.0"
pytest = "^5.4.1"
seaborn = "^0.10.1"
pandas = "^1.0.3"

[tool.poetry.dev-dependencies]

[build-system]
requires = ["poetry>=0.12"]
build-backend = "poetry.masonry.api"
```

APÊNDICE C – ARTIGO CIENTÍFICO

Este apêndice apresenta um artigo sobre o trabalho desenvolvido, formatado seguindo as normas da Sociedade Brasileira de Computação.

HAIL: um modelo de revisão de percepções de agentes inteligentes baseado em alucinação e ilusão

Gustavo E. Kundlatsch[1], **Thiago Ângelo Gelaim**[1], **Elder Santos**[1]

[1]Departamento de Informática e Estatıstica
Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brazil

Abstract. *Perceptions are the main way to an entity to receive information from the environment. Each person has a different manner of percept and interpretate the world. However, it is known that in the human perception there are anomalies, incorrect perceptions that decieve the mind. So, how can we know if our perceptions are real or just tricks from our mind? And the follow up question is: what about computers? Inteligent agents are computational entities capable of making decisions based on the environment, and use sensors to perceive the world. But sensors can fail. In the current work, we present a generic model of perception revision, capable of treating anomalies received by the agent and creating new plans from them to adapt to the environment. This model was implemented and submited to experiments to test it's behavior. The simulations demonstrate that the model is capable of detect and classify anomalies and that in environments with a high amount of anomalies the model can have more inpact creating new plans through the automatic planning process and increasing the amount of valid perceptions received by the agent's cognition.*

Resumo. *Percepções são a principal maneira de uma entidade receber informações do ambiente. Cada pessoa possui uma maneira diferente de perceber e interpretar o mundo. Entretanto, sabe-se que na percepção humana existem anomalias, que são percepções incorretas que enganam a mente. Dito isso, como podemos saber se nossas percepções são reais ou se são apenas fruto de nossa imaginação? E a questão derivada disso é: e computadores? Agentes inteligentes são entidades computacionais autônomas capazes de tomarem decisões baseadas no ambiente no qual estão inseridos, e utilizam sensores para reconhecerem o mundo a sua volta. Mas esses sensores podem falhar. Neste trabalho, apresentamos um modelo genérico de revisão de percepções, capaz de tratar anomalias recebidas pelo agente e criar novos planos a partir delas para se adaptar ao ambiente. Esse modelo foi implementado e submetido a experimentos para testar seu funcionamento. As simulações realizadas demonstraram que o modelo é capaz de detectar e classificar as anomalias e que em ambientes com grande quantidade de anomalias o modelo consegue ter maior impacto criando novos planos através do planejamento automatizado e aumentando a quantidade de percepções válidas recebidas pelo raciocínio do agente.*

1. Introdução

Dentro da Inteligência Artificial (IA), agentes inteligentes são entidades capazes de raciocinar a respeito do ambiente em que estão inseridos e tomar decisões baseadas na situação

em que se encontram [Russell and Norvig 2016]. Dessa maneira, podemos descrever um agente pelo seus processos de percepção, raciocínio e atuação. O agente ocupa um ambiente, do qual recebe informações e no qual atua. O ambiente é o mundo em que o agente está inserido, podendo ser uma construção virtual como uma simulação ou uma parte do mundo real, no caso de um agente físico. Existem diversos tipos de ambientes que podem ser classificados de acordo com o seu fechamento (que determina se agentes de fora do ambiente podem afetar o sistema), dinamismo (a maneira como o ambiente evolui), determinismo (a consistência dos efeitos no ambiente) e cardinalidade (o número de objetos a serem afetados e percebidos) [Moya and Tolk 2007].

Uma das maneiras de um agente atualizar seu conhecimento a respeito do ambiente é a percepção, o processo de utilizar sensores para detectar o ambiente e transformar os dados coletados em informações úteis [Weyns et al. 2004]. O raciocínio, por sua vez, é o processamento das percepções baseado nos objetivos do agente, que resulta em um conjunto ações a serem tomadas através dos atuadores. O processo do raciocínio é comandado pela arquitetura cognitiva do agente, um modelo computacional inspirado na estrutura da mente humana [Dyachenko et al. 2018]. As arquiteturas cognitivas podem ser divididas em três categorias: simbólicas, emergentes e híbridas [Ye et al. 2018]. Arquiteturas simbólicas descrevem o ambiente através de símbolos armazenados em memória em uma base de conhecimentos, e utilizam lógica simbólica para realizar o ciclo de percepção, raciocínio e ação. Arquiteturas emergentes se baseiam na estrutura biológica do cérebro e normalmente utilizam redes neurais em uma estrutura hierárquica para lidar com situações de incerteza. Por fim, arquiteturas híbridas combinam o comportamento emergente e o processamento simbólico para resolver problemas de diversos domínios.

Todavia, sensores podem apresentar problemas para o processo de percepção por razões como campo de visão, distância do objeto observado, resolução dos sensores e leituras não confiáveis [Chrisman et al. 1991]. Tratar deste problema normalmente é responsabilidade da arquitetura cognitiva do agente, pois a arquitetura precisa ser capaz de fazer a ponte entre o ambiente e o conhecimento do agente [Langley et al. 2009].

O objetivo deste trabalho é apresentar um modelo genérico (independente da arquitetura do agente) que pode ser acoplado entre o processo de percepção e raciocínio, capaz de detectar e tratar percepções inválidas para transformá-las em informações úteis através de um processo de criação de novos planos. Esse modelo pressupõe um ambiente aberto (onde agentes externos podem influenciar o ambiente), dinâmico (mudanças no ambiente são causadas por eventos aleatórios) e não determinístico (ações do agente causam resultados diferentes no ambiente, mesmo em situações aparentemente idênticas, pois os resultados variam dependendo da percepção do agente daquele evento).

2. Fundamentação teórica

Alguns conceitos precisam ser bem definidos para que sejam utilizados mais tarde na formalização do modelo proposto. Esse capítulo apresenta essas definições iniciais.

2.1. Agente inteligente

Um agente inteligente é uma entidade autônoma, capaz de tomar as próprias decisões para atingir seus objetivos [Wooldridge 1999]. Apesar da definição intuitiva ser simples, assim como no termo inteligência artificial não existe um consenso da comunidade sobre o que é um agente. Para este trabalho, compomos a seguinte formalização de agente inteligente:

Definição 1. Um agente é uma tripla $Ag = \langle K, P, \gamma \rangle$, onde:

- K é uma base de conhecimentos, tal que $K = K_i \cup K_p$, onde K_i é o conjunto de conhecimentos inicias do agente e K_p os conhecimentos adquiridos através das percepções. K_i é iniciado com valores arbitrários de acordo com a necessidade do agente e K_p é iniciado vazio. Uma base de conhecimentos é uma estrutura que representa fatos a respeito do mundo e apresenta formas de raciocinar a respeito desses fatos para deduzir novos conhecimentos [Hayes-Roth et al. 1983];
- P é o conjunto de planos do agente, sendo um plano definido como $plano = (\Psi, A, \Omega)$, onde Ψ é o conjunto união formado pelas pré-condições das ações que compõem o plano, A o conjunto de ações que compõe o plano e Ω o conjunto união formado pelas pós-condições das ações que compõem o plano. Por sua vez, uma ação é definida como $acao = (\psi, n, \omega)$, sendo ψ um conjunto de pré-condições, n um nome para a ação e ω um conjunto de pós-condições; e
- γ é a função de percepção, definida como $\gamma(p, K) \rightarrow P_i$, onde p é o conjunto de percepções recebidas, K a base de conhecimentos de Ag e P_i o retorno da função, que é um subconjunto próprio do conjunto P de planos do agente.

A partir dessa definição, podemos construir o conceito de contexto, que será importante mais tarde para o modelo de revisão de percepções. O contexto de um agente é o conjunto de todos os símbolos compreendidos pelo agente, e cuja percepção de cada um desses símbolos resulta na execução de um conjunto de ações diretamente mapeadas.

Definição 2. O contexto c de um agente Ag é o domínio de sua função γ.

Para este trabalho, serão utilizados símbolos compostos para representar percepções e ações. Esses símbolos são formados da maneira *predicado(caracteristica)*, onde o predicado é o elemento principal do símbolo e a característica uma propriedade secundária.

2.2. Percepção

Existem diversas definições para o termo "percepção". Podemos entender percepção como um conjunto de sensações que, através da maneira subjetiva que um dado agente o interpreta, representa determinadas entidades do ambiente [Gibson 1950]. Ou seja, a percepção não é simplesmente a representação direta das entidades reais que existem no mundo, mas um processo complexo que varia para cada indivíduo.

Para o modelo que iremos propor, baseado na Definição 1, o conceito de percepção pode ser simplesmente definido como as entradas da função de percepção γ de um agente. Vale destacar a diferença entre percepção e contexto, pois o contexto é constituído pelas percepções que fazem parte do domínio da função γ, ou seja, uma percepção é toda informação produzida pelo ambiente que o agente recebe, e contexto é o subconjunto das percepções que o agente possui planos para tratar.

2.2.1. Refinamento

Como o volume de percepções de um agente pode ser muito grande, e as percepções tomam determinado tempo para serem processadas, o número de percepções que chegam ao ciclo de raciocínio do agente pode ser reduzido para diminuir seu custo computacional. Neste trabalho, esse processo será chamado de refinamento, definido da seguinte maneira:

Definição 3. Refinamento de percepções é uma função θ tal que, dado o conjunto de entradas de percepções p, reduz tais percepções para um subconjunto próprio ρ.

2.2.2. Anomalias

Anomalias são as percepções consideradas inválidas, geradas por alguma falha no processo de percepção. Toda percepção para a qual o agente não possui uma resposta mapeada em sua função de percepção é considerada uma anomalia. O conjunto de anomalias de um agente é o conjunto de todas as percepções possíveis que não fazem parte de seu contexto. A seguinte definição será utilizada:

Definição 4. Uma percepção p de um agente Ag com contexto c, é uma anomalia caso $p \notin c$ de Ag.

Neste trabalho, as anomalias serão classificadas em dois grupos: alucinações e ilusões. Essa classificação foi inspirada no problema da percepção, um campo de estudo da filosofia [Crane and French 2017].

Uma ilusão é uma percepção da forma *predicado(caracteristica)* onde ou o predicado ou a característica não faz parte do contexto do agente, ou seja, é uma percepção parcialmente correta. Uma alucinação é uma uma percepção da forma *predicado(caracteristica)* onde nem o predicado nem a característica faz parte do predicado, ou seja, a percepção pode ser semanticamente correta mas ela não deveria ter sido realizada pelo agente.

2.3. Planejamento automatizado

Planejamento automatizado é um dos problemas fundamentais da Inteligência Artificial. As motivações para usar o planejamento automatizado são a capacidade de utilizar recursos de planejamento acessíveis e eficientes e reproduzir uma parte do processo cognitivo humano com um componente totalmente integrado de comportamento deliberativo [Ghallab et al. 2004].

Na Definição 5 é apresentada a noção abstrata de planejamento automático, descrita como um modelo conceitual simples que contém os elementos principais do problema, tendo sido originalmente apresentada por [Ghallab et al. 2004].

Definição 5. Um modelo conceitual de planejamento automatizado é descrito como a interação entre os seguintes três componentes:

- Um sistema de transição de estados Σ, especificado por uma função de transição de estados y, de acordo com os eventos e ações que ele recebe.
- Um $controlador$, que dado uma entrada de estados s do sistema, fornece como saída uma ação de acordo com algum plano.
- Um $planejador$, que dado uma entrada de uma descrição de sistema Z, uma situação inicial e alguns objetivos, sintetiza um plano para o controlador a fim de alcançar o objetivo.

Um sistema de transição de estados Σ é uma quádrupla $\Sigma = \langle S, A, E, \Gamma \rangle$, onde:

- $S = \{s_1, s_2, ..., s_n\}$ é um conjunto finito ou recursivamente enumerável de estados;

- $A = \{a_1, a_2, ..., a_n\}$ é um conjunto finito ou recursivamente enumerável de ações;
- $E = \{e_1, e_2, ..., e_n\}$ é um conjunto finito ou recursivamente enumerável de eventos; e
- $\Gamma : S \times A \times E \rightarrow 2^S$ é uma função de transição de estados.

3. Trabalhos relacionados

Existem diversas abordagens para otimizar as percepções recebidas por um agente, isto é, garantir que todas as informações coletadas pelos sensores sejam utilizadas da melhor maneira possível. Diversos artigos apresentam processos de detectar percepções inválidas e tratá-las, de maneira similar ao modelo que será proposto neste trabalho.

Por exemplo, no artigo *Scalable perception for BDI-agents embodied in virtual environments* [van Oijen and Dignum 2011] é definido um *framework* onde os objetos do ambiente são definidos por classes e características e se organizam de maneira hierárquica, e a partir dessa organização o *framework* é capaz de decidir que tipo de percepções o agente deseja perceber de acordo com seus interesses, alterando dinamicamente ao longo do tempo.

Outro trabalho similar é o *PMK – a knowledge processing framework for autonomous robotics perception and manipulation* [Diab et al. 2019], onde o objetivo dos autores foi criar um mapa de todas as percepções possíveis no ambiente, de maneira que todas as percepções recebidas pelo agente estivessem em seu contexto, evitando anomalias. Todavia, isso não é viável em cenários abertos ou de alta complexidade, pois o mapeamento necessário é extenso demais. O modelo de programação lógica K-CoPMan (*Knowledge enabled Cognitive Perception for Manipulation* ou Percepção Cognitiva Ativada pelo Conheci-mento para Manipulação) apresenta uma maneira do agente criar sua própria base de conhecimentos a partir da percepção passiva, eliminando as anomalias conforme elas são adicionadas ao contexto [Pangercic et al. 2010].

Esses exemplos são de artigos que buscam otimizar as percepções recebidas pelo agente de maneira simbólica, mas esse problema pode ser abordado de maneira conexionista também. No artigo *Understanding human intention by connecting perception and action learning in artificial agents* [Kim et al. 2017] os autores propõem um modelo, chamado OA-SMTRNN (*Object Augmented Supervised Multiple Timescale Recurrent Neural Network*), para entender a intenção do usuário e responder ativamente da maneira mais adequada, através do uso de redes neurais. Para implementar o reconhecimento de intenção, são focados dois processos cognitivos, a percepção da disponibilidade de objetos e a previsão da ação humana. Com o uso de redes neurais, a ideia é extrair semântica de percepções que seriam inicialmente inválidas para o agente.

O objetivo do modelo proposto neste trabalho é detectar anomalias de maneira simbólica, classificando as percepções recebidas com o uso do contexto do agente como referencial, para então utilizar um processo de planejamento automatizado para gerar novos planos para o agente a partir das percepções inválidas.

4. Modelo proposto

O modelo proposto foi inspirado pelos conceitos de ilusão e alucinação apresentados na Seção 2.2.2, que o nomeiam – o nome HAIL vem da junção das palavras *hallucination*

e *illusion*, alucinação e ilusão em inglês, respectivamente. Seu objetivo é identificar anomalias nas percepções recebidas por um agente qualquer e torná-las informação úteis na forma de novos planos.

O modelo HAIL foi desenvolvido para que seja possível adicioná-lo a qualquer agente, independente de arquitetura cognitiva, como um componente que conecta as percepções vindas do ambiente ao agente. O HAIL pode ser separado em dois módulos, como mostra a Figura 1. De maneira geral, o funcionamento e a comunicação desses módulos se dá da seguinte maneira:

1. As percepções recebidas pelo modelo são refinadas pelo módulo de refinamento;
2. As percepções refinadas passam pelo módulo de alucinação e ilusão onde são categorizadas entre: percepções válidas, alucinações, ilusões classe 1 e ilusões classe 2.
3. As percepções válidas são encaminhadas para o raciocínio do agente, enquanto as anomalias continuam no módulo de alucinação e ilusão armazenadas em estruturas chamadas de bloco avaliador;
4. Quando os requisitos estabelecidos pelo bloco avaliador são cumpridos, as anomalias são selecionadas para passarem pelo processo de planejamento automatizado, alimentando o agente com novos planos.

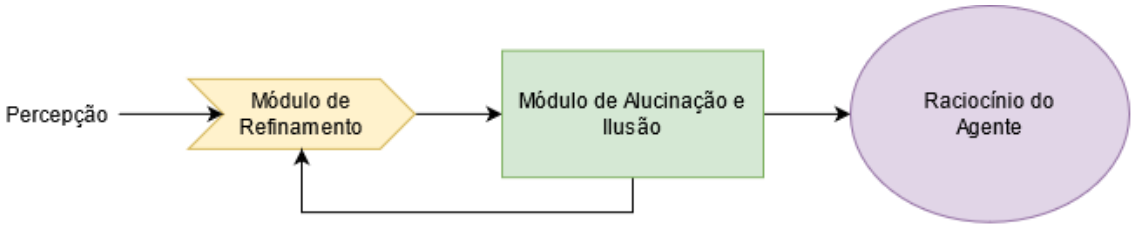


Figura 1. Visão geral do modelo HAIL.

4.1. Módulo de refinamento

O módulo de refinamento funciona como uma primeiro filtro para que percepções indesejadas pelo agente não cheguem até seu ciclo de raciocínio. O processo de refinamento é descrito pela Definição 3.

O processo de refinamento não é obrigatório. Caso não seja de interesse de uma determina implementação do HAIL refinar suas percepções, basta que a função do módulo de refinamento seja a função identidade $f(x) = x$, possuindo assim $\rho = p$.

4.2. Módulo de alucinação e ilusão

A Figura 2 apresenta um diagrama do funcionamento do módulo de alucinação e ilusão. Sua função é receber todas as percepções que passaram pelo processo de refinamento, e detectar quais delas são anomalias. Para isso, esse módulo utiliza uma cadeia de decisores descrita pelo Algoritmo 1. Depois de serem classificadas pelos decisores, as anomalias são guardadas em seus respectivos blocos avaliadores.

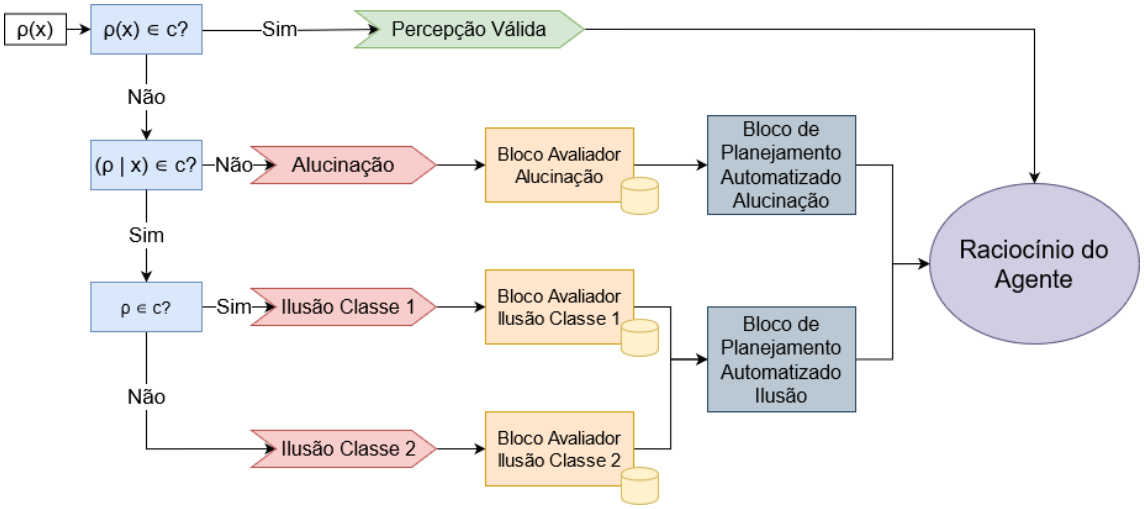


Figura 2. Módulo de alucinação e ilusão.

Entrada: contexto c do agente, percepção $\rho(x)$
início
  se $\rho(x)$ *está em* c **então**
    $\rho(x)$ é uma percepção válida;
  senão se *nem* ρ *nem* x *estão em* c **então**
    $\rho(x)$ é uma alucinação;
  senão se ρ *está em* $\subset$ **então**
    $\rho(x)$ é uma ilusão classe 1;
  senão
    $\rho(x)$ é uma ilusão classe 2
  fim
fim

Algoritmo 1: Funcionamento dos decisores do módulo de alucinação e ilusão.

4.2.1. Bloco avaliador

O objetivo do bloco avaliador é decidir quando alucinações e ilusões que foram recebidas devem ser processadas, evitando que o planejamento automatizado que será realizado em seguida tenha impacto no tempo de execução de um ciclo de raciocínio do agente. Para isso, utilizamos uma lista ordenada por peso como escalonador. O principio do funcionamento da lista ordenada por peso é o mesmo de uma fila, *first in first out* (FIFO), mas atribui um peso a cada entrada que aumenta quando novos elementos iguais são inseridos. Quando um elemento é inserido pela primeira vez na fila, ele recebe o peso 1, e quando uma cópia do mesmo elemento é inserida o elemento tem seu peso aumentado em 1. Quando uma operação de remoção é executada, o elemento de maior peso é removido. Se dois ou mais elementos tiverem o maior peso, aquele que foi inserido primeiro é removido.

O bloco avaliador seleciona quando uma percepção deve ser tratada através de uma função matemática, levando em conta o tempo médio de processamento de uma

percepção válida e de uma anomalia. Além desse funcionamento básico, o bloco avaliador ainda contém um mecanismo para remover anomalias classificadas com irrelevantes para o sistema, através de uma função de limpeza. Caso essa função retorne verdadeiro, todos os elementos de peso 1 da sua respectiva lista são removidos. Essas duas funções são descritas com mais detalhes na Seção 4.3.

4.2.2. Bloco de planejamento automatizado

O bloco de planejamento automatizado é potencialmente a parte mais custosa computacionalmente, o que pode ser um gargalo do sistema, principalmente caso o agente funcione em tempo real e receba um volume muito elevado de percepções por segundo. Um planejamento automatizado implementado de maneira puramente simbólica tende a ser complexo computacionalmente, uma vez que pode considerar milhares de alternativas para o estado de mundo atual, tentando chegar mais perto de seu objetivo. Um processo de planejamento automatizado conexionista é uma alternativa, uma vez que estamos tratando de uma análise incompleta do mundo. Caso seja possível, teorias com maior custo computacional (como criatividade computacional [Colton et al. 2012]) podem ser aplicadas aqui para um resultado ainda mais preciso.

Uma percepção chega ao bloco de planejamento automatizado uma vez que ela seja a primeira na fila ponderada e a função de processamento retorne verdadeiro em sua verificação. De um ciclo para outro, as percepções permanecem na fila, a não ser que sejam descartadas pelo mecanismo de limpeza. Nosso modelo não explicita qual é a ordem que os blocos avaliadores devem processar suas filas para mandar anomalias para o planejamento automatizado (isto é, se deve primeiro ser priorizada as anomalias do bloco avaliador de alucinações, ilusões classe 1 ou ilusões classe 2), ficando a cargo da implementação em questão tomar essa decisão.

4.3. Formalização

O bloco básico do modelo de revisão de percepções proposto, chamado de HAIL, é composto por um módulo para alucinação e ilusão M_{ai} e uma função de refinamento θ, conforme descrito na Definição 6. O módulo de ilusão e alucinação é uma quádrupla, apresentada na definição 7. A função de refinamento é uma função abstrata, cuja entrada é obtida através dos sensores do agentes e a saída é a entrada do módulo de alucinação e ilusão, conforme já foi descrito anteriormente na Seção 4.1.

Definição 6. O modelo de revisão de percepções HAIL é uma dupla $HAIL = \langle M_{ai}, \theta \rangle$, onde:

- M_{ai} é o módulo de ilusão e alucinação; e
- θ é a função de refinamento $\theta(p) = \rho$, onde p é um conjunto de percepções e ρ é um subconjunto próprio de p.

Após ter passado pela função θ, as percepções ρ irão passar pelo Algoritmo 1, e serão encaminhadas de acordo com sua classificação. O bloco de ilusão e alucinação é descrito por uma quádrupla, com conjuntos de decisores, blocos e uma função de transição.

Definição 7. O bloco de ilusão e alucinação é uma quádrupla $M_{ai} = \langle D, Ab, Ap, \Delta \rangle$, onde:

- D é o conjunto de decisores $D = \{d_a, d_h, d_i\}$, onde:
  - d_a é o decisor de anomalias, definido pela função:

$$d_a = \begin{cases} 0 & \text{se } \rho(x) \text{ está em } c^1; \\ 1 & \text{se } \rho(x) \text{ não está em } c. \end{cases}$$

  - d_h é o decisor de alucinação, definido pela função:

$$d_h = \begin{cases} 0 & \text{se nem } \rho \text{ nem } (x) \text{ está em } c; \\ 1 & \text{se } \rho \text{ ou } (x) \text{ está em } c. \end{cases}$$

  - d_i é o decisor de ilusão, definido pela função:

$$d_i = \begin{cases} 0 & \text{se } \rho \text{ está em } c; \\ 1 & \text{se } (x) \text{ está em } c. \end{cases}$$

- Ab é o conjunto de blocos avaliadores $Ab = \{Ab_h, Ab_{i1}, Ab_{i2}\}$, onde Ab_h é o bloco avaliador de alucinações, Ab_{i1} é o bloco avaliador de ilusões classe 1 e Ab_{i2} é o bloco avaliador de ilusões classe 2.
- Ap é o conjunto de blocos de planejamento automatizado $Ap = \{Ap_h, Ap_i\}$, onde Ap_h é o bloco de planejamento automatizado de alucinações e Ap_i é o bloco de planejamento automatizado de ilusões.
- Δ é a função de transição definido pela tabela abaixo, onde out é um estado final, que leva a percepção para fora do modelo de revisão de percepções, ou seja, pode tanto significar o encaminhamento de uma percepção válida para o raciocínio do agente quanto o fim da execução de um ciclo de revisão.

Tabela 1. Função de transição Δ do módulo de ilusão e alucinação.

Estado	0	1
d_a	out	d_h
d_h	Ab_h	d_i
d_i	Ab_{i1}	Ab_{i2}
Ab_h	out	Ap_h
Ab_{i1}	out	Ap_i
Ab_{i2}	out	Ap_i

O módulo de ilusão e alucinação é o artefato principal do modelo. Ele é subdividido em três partes principais: os decisores, os blocos avaliadores e os blocos de planejamento automatizado. Seu comportamento é descrito por uma função de transição que define o estado atual do módulo.

Definição 8. Um bloco avaliador é uma tripla $Ab_x = \langle L, Pf, Cf) \rangle$, $x \in \{h, i1, i2\}$, onde:

[1] c é o contexto do agente, de acordo com a definição 2.

- L é uma lista ordenada pelo número de vezes que uma mesma anomalia é dada como entrada;
- Pf é a função de processamento, definida abaixo:

$$Pf = \begin{cases} 1 & \text{se } T_m(A) \leq T_m(V) * (|A| - |A_{pr}|); \\ 0 & \text{caso contrário.} \end{cases}$$

  Onde:
  - T_m é a função que retorna a média do tempo gasto para processar as percepções de um conjunto;
  - A é o conjunto de anomalias, $A(x)$ é um elemento específico x e $|A|$ o número de anomalias do conjunto;
  - A_{pr} é o conjunto de anomalias que já foram validadas para serem processadas pela função de processamento neste ciclo de raciocínio (A_{pr} é instanciada vazia a cada ciclo de raciocínio), e $|A_{pr}|$ o número de anomalias desse conjunto.
  - V é o conjunto de percepções válidas.
- Cf é a função de limpeza definida abaixo com auxílio da função equação de limpeza Cf, sendo α um coeficiente de limpeza variável que precisa ser definido pela instância implementada do modelo (por padrão, toma-se $\alpha = 1$):

$$Cf = \begin{cases} 1 & \text{se } Ce = Verdadeiro; \\ 0 & \text{caso contrário.} \end{cases}$$

$$Ce = \sum_{i=1}^{|L|} W_n(L_i) > \alpha \sum_{j=1}^{|L|} W_1(L_j)$$

  Onde:
  - L é a lista ordenada do bloco, sendo $|L|$ seu número de anomalias e L_i a anomalia i da lista.
  - W é a função peso da anomalia L_i definida como $W(L_i) = |L_i|$, sendo $|L_i|$ o peso da anomalia especificada (número de entradas recebidas dessa mesma anomalia na lista). A função W é utilizada para especificar as seguintes funções:

    (i) $W_1(L_i) = \begin{cases} 1 & \text{se } W(L_i) = 1; \\ 0 & \text{caso contrário.} \end{cases}$

    (ii) $W_n(L_i) = \begin{cases} W(Li) & \text{se } W(L_i) > 1; \\ 0 & \text{caso contrário.} \end{cases}$

O bloco de planejamento automatizado recebe como entrada uma anomalia de um dos blocos avaliadores quando a função de processamento retorna verdadeiro, e pode ser definido da seguinte maneira:

Definição 9. Um bloco de planejamento automatizado é uma instância do modelo conceitual de planejamento automatizado (Definição 5).

5. Experimentos

Para este trabalho foram realizados três experimentos com o objetivo de averiguar se o modelo HAIL é funcional ou não. Esses experimentos não possuem um domínio específico, ou seja, não há um ambiente real onde o agente está inserido, ele recebe percepções aleatórias geradas pelo simulador e os planos inicias do agente não são relevantes para os testes.

Os dois primeiros experimentos foram implementados utilizando o design 2^k fatorial [Jain 1990]. Esse tipo de design consiste em variar k fatores em 2 níveis diferentes, -1 e 1, que são extremos opostos. Os fatores e as variáveis livres utilizadas estão nas Tabelas 2 e 3, respectivamente. A análise do impacto dos fatores foi realizada utilizando a equação de regressão não linear do design 2^k fatorial.

Tabela 2. Fatores utilizados nos experimentos realizados com o modelo HAIL.

Fatores	Sigla	Nível -1	Nível 1
Porcentagem de Percepções Inválidas	PPI	5%	95%
Tempo Médio gasto pelo planejamento Automatizado	TMA	1/2 T	64 T
Tempo Médio gasto em um Ciclo de raciocínio	TMC	01 T	32 T
Número de Percepções recebidas por Ciclo	NPC	01	16

Tabela 3. Variáveis dependentes analisadas nos experimentos realizados com o modelo HAIL.

Variáveis	Motivação
Tempo Virtual Decorrido	Medir desempenho geral do modelo
Planos Criados	Avaliar potencial do modelo de inserir aprendizado em arquiteturas que não o possuem
Percepções Válidas Processadas	Analisar a capacidade do modelo de ganhar desempenho ao longo do tempo

Os experimentos consistem em diversas simulações, que submetem o modelo proposto ao processo de revisão de um grande número de percepções. Uma simulação possui 5000 ciclos de percepção, sendo que cada ciclo pode possuir uma ou várias percepções. Essas percepções podem ser válidas (pertencentes ao contexto do agente) ou inválidas (não pertencentes ao contexto do agente), sendo que a proporção entre o tipo de percepções é definido pela PPI. As percepções são produzidas aleatoriamente por um gerador de percepções.

O gerador de percepções cria as percepções válidas sorteando símbolos que pertencem ao contexto do agente, e as percepções inválidas são criadas utilizando o pacote *RandomWords* [ŚwiEcicki 2012], que gera palavras em inglês aleatórias. As percepções são símbolos compostos da forma *corpo(argumento)*, e a quantidade de percepções válidas e inválidas geradas é baseado no PPI e TMA da simulação executada.

Para mensurar o tempo gasto pelos processos do agente, será considerada uma unidade de tempo genérica T, e iremos considerar o tempo virtual da execução das simulações com base nessa unidade.

Tabela 4. Análise dos fatores do experimento 1.

Fator	Efeito Tempo Virtual	Efeito Planos Criados
PPI	12.69%	66.10%
TMC	22.20%	0.50%
TMA	31.64%	2.95%
NPC	1.44%	8.67%
PPI + TMC	4.16%	3.76%
PPI + TMA	24.13%	0.05%
PPI + NPC	1.39%	2.13%
TMC + TMA	0.55%	0.50%
TMC + NPC	0.10%	0.50%
TMA + NPC	0.01%	2.99%
PPI + TMC + TMA	0.53%	3.76%
PPI + TMC + NPC	0.09%	3.75%
PPI + TMA + NPC	0.02%	0.06%
TMC + TMA + NPC	0.54%	0.50%
PPI + TMC + TMA + NPC	0.52%	3.76%

5.1. Experimento 1

O objetivo do primeiro experimento é analisar o impacto da mudança dos fatores no desempenho do modelo através da analise da variação no tempo virtual decorrido e da quantidade de planos criados pelo módulo de planejamento automatizado.

Cada simulação foi repetida dez vezes, e a média dos resultados foi obtida. Isso foi feito para garantir que os valores obtidos não foram afetados pela geração aleatória de percepções. Quando repetições são adotadas no experimento normalmente utiliza-se a metodologia $2^k r$ fatorial, que leva em conta o erro obtido entre as diferentes execuções da simulação. Todavia, ao fazer a análise de erros o resíduo obtido foi descartável, com ordem de grandeza e^{-12}. Portanto, a metodologia de análise adotada foi a do 2^k fatorial.

Esse experimento demonstrou que o módulo de ilusão e alucinação funciona como previsto: simulações com mais percepções inválidas e menor tempo médio de planejamento automatizado resultam em menos tempo virtual gasto e mais planos criados. Em situações com um grande volume de percepções, implementações que possuem um bloco de planejamento automatizado que consome muito tempo para criar novos planos podem prejudicar o desempenho do agente caso ele seja muito sensível ao tempo. O resultado da análise de impacto dos fatores está exposto na Tabela 4.

5.2. Experimento 2

O segundo experimento foi realizado para analisar o ganho de performance de um agente ao utilizar os planos criados com o bloco de planejamento automatizado. Esse segundo experimento segue a metodologia do primeiro, porém foi realizado apenas uma iteração para cada configuração de fatores (ao invés das dez repetições). Após ser realizada uma simulação com uma configuração de fatores foi realizada uma nova simulação com a mesma configuração, mas usando o agente resultante da primeira. Assim, os planos criados pelo agente foram reaproveitados. Ao invés de analisar quais são os fatores que mais

Tabela 5. Alteração no valor de tempo virtual no experimento 2.

Simulação	PPI	TMC	TMA	NPC	R1	R2	RP
1.1	5%	1	64	1	20750T	20687T	99.70%
1.2	5%	1	64	16	20813T	20750T	99.70%
1.3	5%	32	64	1	168032T	167968T	99.96%
1.4	5%	32	64	16	168032T	168000T	99.98%
1.5	95%	1	64	1	227579T	10859T	4.77%
1.6	95%	1	64	16	304313T	177998T	58.49%
1.7	95%	32	64	1	273536T	162400T	59.37%
1.8	95%	32	64	16	312032T	250048T	80.14%
2.1	5%	1	1/2	1	4874.5T	4876.5T	100.04%
2.2	5%	1	1/2	16	5000T	5000T	100%
2.3	5%	32	1/2	1	152093.5T	152219.5T	100.08%
2.4	5%	32	1/2	16	153435.5T	152887T	99.64%
2.5	95%	1	1/2	1	3228.5T	4955T	153.48%
2.6	95%	1	1/2	16	5000T	4944T	98.88%
2.7	95%	32	1/2	1	48458.5T	157385.5T	324.78%
2.8	95%	32	1/2	16	160000T	140631T	113.77%

influenciam nas variáveis dependentes, comparamos os resultados dessas variáveis antes e depois da criação de novos planos. As variáveis dependentes analisadas foram novamente o tempo virtual decorrido e o número de planos criados.

O principal conclusão obtida a partir dos resultados do experimento 2 é que o principal fator para que o HAIL possa ter impacto no tempo virtual do agente e na quantidade de planos novos que ele recebe é a quantidade de percepções inválidas recebidas. Cenários com poucas percepções não tiveram um grande ganho nessas variáveis. Além disso, foi constatado que em cenários com grande número de anomalias (PPI 95% e NPC 16) o TMA se torna um gargalo para o funcionamento do HAIL, ou seja, quanto mais percepções inválidas são recebidas mais otimizado o bloco de planejamento automatizado precisa ser.

Os resultados das simulações estão expostos nas tabelas 5 (tempo virtual) e 6 (planos criados). As tabelas contém o índice da simulação, os níveis dos fatores, os resultados da primeira e da segunda iteração (R1 e R2, respectivamente) e a relação percentual (RP) entre os resultados, ou seja, a proporção de R2 em relação a R1, obtido através do cálculo $(R2 * 100)/R1$.

5.3. Experimento 3

O terceiro experimento foi executado para observar a capacidade de aprendizado do agente ao longo do tempo. Nele foram executadas 5 simulações com os mesmos fatores dos experimentos anteriores, mas com valores diferentes (como demonstra a Tabela 7). Entre uma simulação e outra o agente não foi recarregado, ou seja, manteve o aprendizado das simulações anteriores.

Nessa simulação foi possível observar que enquanto os planos criados diminuem a cada simulação, pois menos percepções são inválidas (uma vez que o agente aprende

Tabela 6. Alteração no valor de planos criados no experimento 2.

Simulação	PPI	TMC	TMA	NPC	R1	R2	RP
1.1	5%	1	64	1	250.0	249.0	99.60%
1.2	5%	1	64	16	251.0	250.0	99.60%
1.3	5%	32	64	1	251.0	249.0	99.20%
1.4	5%	32	64	16	251.0	250.0	99.60%
1.5	95%	1	64	1	3533.0	93.0	2.63%
1.6	95%	1	64	16	4751.0	2746.0	57.80%
1.7	95%	32	64	1	3548.0	75.0	2.11%
1.8	95%	32	64	16	4751.0	2814.0	59.23%
2.1	5%	1	1/2	1	251.0	247.0	98.41%
2.2	5%	1	1/2	16	502.0	500.0	99.60%
2.3	5%	32	1/2	1	251.0	247.0	98.41%
2.4	5%	32	1/2	16	2935.0	878.0	29.91%
2.5	95%	1	1/2	1	3543.0	90.0	2.54%
2.6	95%	1	1/2	16	9312.0	260.0	2.79%
2.7	95%	32	1/2	1	3541.0	83.0	2.34%
2.8	95%	32	1/2	16	4078.0	0.0	0.0%

com as simulações anteriores), a quantidade de percepções válidas processadas aumenta. Esse comportamento é demonstrado na Figura 5.3. Além disso, o tempo virtual cai na mesma proporção que os planos criados diminuem, uma vez que a criação de um plano é duas vezes mais custosa em tempo do que o processamento de uma percepção válida nesse experimento.

Tabela 7. Valor dos fatores no experimento 3.

Fator	Valor
PPI	50%
TMC	16
TMA	32
NPC	8

6. Conclusão

Agentes são entidades autônomas que, inseridas em um ambiente, são capazes de tomar decisões de maneira autônoma. Para se comunicar com o ambiente no qual estão inseridos, os agentes realizam o processo de percepção, utilizando sensores para receber diversos tipos de informações do mundo ao seu redor. Todavia, as percepções recebidas podem ser incorretas por diversos motivos.

Neste trabalho, apresentamos um modelo genérico capaz de detectar percepções inválidas, classificá-las de acordo com suas características e criar novos planos a partir delas. Esse modelo foi implementado e experimentos foram feitos para mostrar que seu funcionamento era de acordo com o esperado.

Para os trabalhos futuros, é necessário criar versões alternativas de cada um de

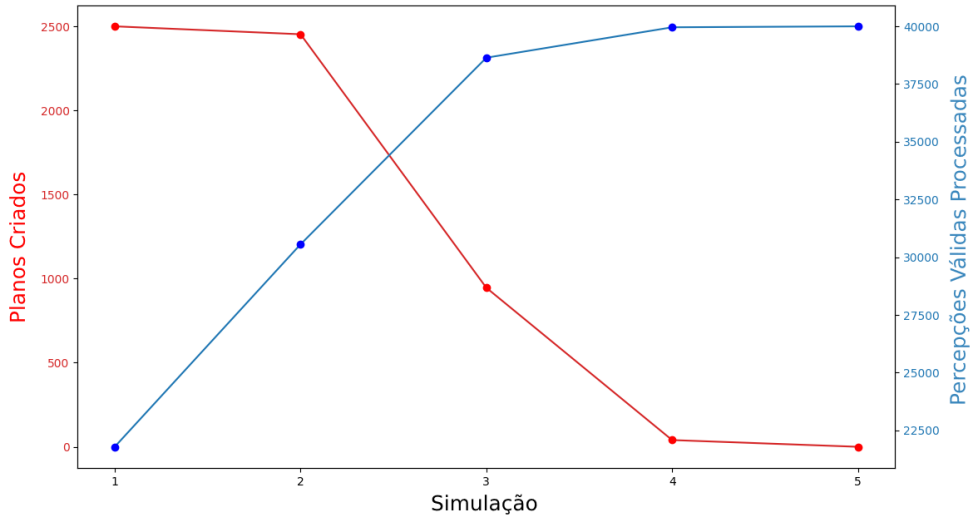


Figura 3. Evolução das percepções válidas processadas e dos planos novos criados ao longo das simulações.

seus componentes para validar se seu comportamento é o melhor possível. Além disso, é necessário testar o modelo em um ambiente real, acoplado a uma arquitetura específica, para testar seu desempenho em diferentes situações.

Referências

Chrisman, L., Caruana, R., and Carriker, W. (1991). Intelligent agent design issues: Internal agent state and incomplete perception. In *Proceedings of the AAAI Fall Symposium on Sensory Aspects of Robotic Intelligence. AAAI Press/MIT Press*. Citeseer.

Colton, S., Wiggins, G. A., et al. (2012). Computational creativity: The final frontier? In *Ecai*, volume 12, pages 21–26. Montpelier.

Crane, T. and French, C. (2017). The problem of perception. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2017 edition.

Diab, M., Akbari, A., Ud Din, M., and Rosell, J. (2019). Pmk—a knowledge processing framework for autonomous robotics perception and manipulation. *Sensors*, 19(5):1166.

Dyachenko, Y., Nenkov, N., Petrova, M., Skarga-Bandurova, I., and Soloviov, O. (2018). Approaches to cognitive architecture of autonomous intelligent agent. *Biologically Inspired Cognitive Architectures*, 26:130 – 135.

Ghallab, M., Nau, D., and Traverso, P. (2004). Chapter 1 - introduction and overview. In Ghallab, M., Nau, D., and Traverso, P., editors, *Automated Planning*, The Morgan Kaufmann Series in Artificial Intelligence, pages 1 – 16. Morgan Kaufmann, Burlington.

Gibson, J. J. (1950). The perception of the visual world.

Hayes-Roth, F., Waterman, D. A., and Lenat, D. B. (1983). Building expert system.

Jain, R. (1990). *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. John Wiley & Sons.

Kim, S., Yu, Z., and Lee, M. (2017). Understanding human intention by connecting perception and action learning in artificial agents. *Neural Networks*, 92:29–38.

Langley, P., Laird, J. E., and Rogers, S. (2009). Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, 10(2):141–160.

Moya, L. J. and Tolk, A. (2007). Towards a taxonomy of agents and multi-agent systems. In *SpringSim (2)*, pages 11–18.

Pangercic, D., Tenorth, M., Jain, D., and Beetz, M. (2010). Combining perception and knowledge processing for everyday manipulation. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1065–1071.

Russell, S. J. and Norvig, P. (2016). *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

van Oijen, J. and Dignum, F. (2011). Scalable perception for bdi-agents embodied in virtual environments. In *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 2, pages 46–53. IEEE.

Weyns, D., Steegmans, E., and Holvoet, T. (2004). Towards active perception in situated multi-agent systems. *Applied Artificial Intelligence*, 18(9-10):867–883.

Wooldridge, M. (1999). Intelligent agents. *Multiagent systems*, 35(4):51.

Ye, P., Wang, T., and Wang, F. (2018). A survey of cognitive architectures in the past 20 years. *IEEE Transactions on Cybernetics*, 48(12):3280–3290.

ŚwiEcicki, T. (2012). Randomwords · pypi. `https://pypi.org/project/RandomWords/`. Acesso 21/07/2020, versão 0.3.0.