INTELIGÊNCIA ARTIFICIAL

Prof. Luciano Frontino de Medeiros

TEMA 1 – AGENTES INTELIGENTES

O conceito de **agente inteligente** é a abstração que permeia o comportamento inteligente, seja por algoritmos com aprendizado de máquina, seja por robôs projetados para alguma atividade. Um agente precisa executar tarefas de forma autônoma, testando as condições do ambiente e executando ações conforme um programa estabelecido. Isso se aplica tanto para um robô quanto para um software que não esteja embutido em um artefato real.

De acordo com Russell e Norvig (2004), um agente é definido como um artefato equipado com **sensores** que têm a capacidade para perceber o ambiente e agir sobre ele por meio de **atuadores**. Comparando-se com o ser humano, os sensores são equivalentes aos olhos, ouvidos, nariz e o tato. Os atuadores podem equivaler às mãos, às pernas e à boca e demais partes ativas de nossa anatomia.

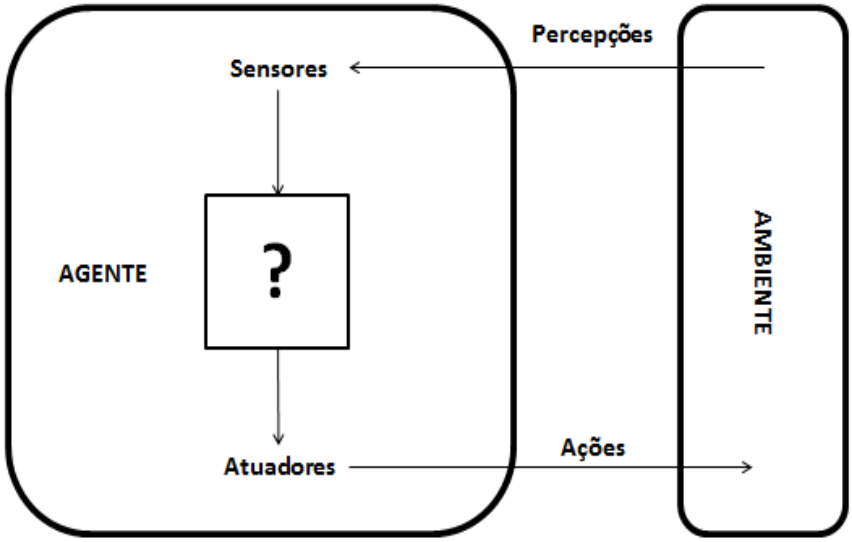
Robôs são construídos com as mais variadas funcionalidades, tendo sensores tais como câmeras, dispositivos de infravermelho, de som e ultrassom, de gás, de temperatura e de umidade, dentre outros. Na qualidade de atuadores, podemos relacionar os servomotores, motores de passo, pistões hidráulicos e relês. Entretanto, é necessário que um agente processo os sinais provenientes dos sensores para efetuar qualquer ação de forma a se caracterizar como “inteligente” em seu ambiente (Medeiros, 2018, p. 35).

Da mesma forma, um algoritmo de aprendizado de máquina, atuando como um agente inteligente, recebe os dados como entrada, que pode ser por meio de sensores que analisam os dados de forma *on-line* em um determinado ambiente. Em seguida, executam um processamento sobre os dados, de acordo com técnicas e algoritmos que aprendem sobre o contexto dos dados de entrada, para depois informar os resultados obtidos ou fornecer os subsídios para tomada de decisões.

Um agente inteligente tende a interagir com o seu ambiente da maneira visualizada na Figura 1. Os sensores podem atuar de forma combinada para bem caracterizar uma percepção de algo acontecendo no ambiente. Entretanto, a percepção do ambiente está sujeita a alterações, manifestando um aspecto temporal; dessa forma, é necessário considerar também a sequência de percepções. Essa sequência é processada internamente pelo agente. Conforme Russel e Norvig (2004, p. 34), “a escolha da ação de um agente em qualquer

instante dado pode depender da sequência inteira de percepções observadas até o momento".

Figura 1 – Ilustração da interação do agente inteligente com o ambiente e seus elementos internos



Fonte: Russel e Norvig, 2004.

O processamento dos sinais pelo agente para a execução de ações é similar ao conceito de função matemática. Assim, denomina-se **função do agente** o mapeamento das ações à disposição do agente com as sequências de percepções que podem acontecer na memória do agente (Russell; Norvig, 2004).

A partir do conceito de agente, pode-se perceber que a tarefa principal do projetista de Inteligência Artificial está na determinação da função do agente, ou seja, o seu comportamento. Para agentes que executarão tarefas simples, a função do agente pode ser bem determinada e seu comportamento pode acondicionar um conjunto de ações bem definidas. Entretanto, para outros agentes em ambientes incertos e com alta complexidade, a função do agente pode ser mesmo impossível de se descrever de maneira bem definida.

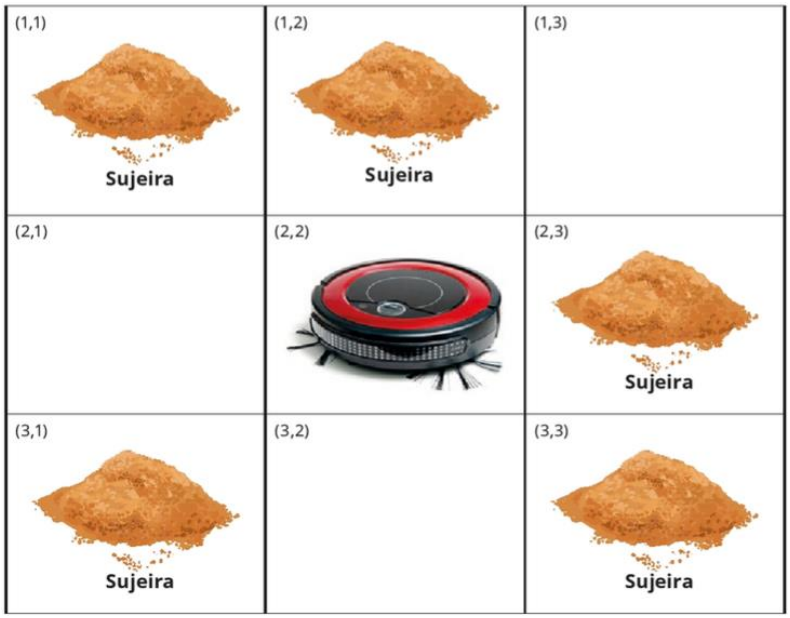
Para os agentes que desenvolvam tarefas simples como, por exemplo, um robô aspirador, a sua função de agente é bem delimitada, de maneira a produzir um comportamento que pode ser aplicado a uma faixa bem específica de ambientes sobre os quais ele pode fazer a limpeza. Por sua vez, um robô projetado para jogar xadrez é um verdadeiro desafio, pois a quantidade de jogadas permitidas pelo jogo chega a valores astronômicos (aproximadamente 10^{120}

possibilidades, estimando-se uma média de 50 jogadas – o número estimado de átomos existentes no Universo chega a 10^{80}!). Em vista disso, um agente inteligente para o jogo de xadrez deve ter uma função de agente restrita, de forma que não leve milhões de anos para fazer uma única jogada, e que possua memória suficiente para armazenar as jogadas possíveis, considerando um horizonte de poucas jogadas à frente (Medeiros, 2018, p.36).

A função do agente se refere a uma descrição abstrata. Essa descrição pode ser representada utilizando-se diagramas ou outras simbologias que demonstrem o que o agente deve fazer. Tendo-se como ponto de partida a função do agente, elabora-se o **programa do agente**, que é implementado de forma concreta e está comprometido com aspectos da arquitetura do agente (Russel; Norvig, 2004, p. 34). O programa é expresso em alguma linguagem de programação utilizada para desenvolver o agente.

Ilustrando-se com um exemplo mais simplificado, supõe-se o “mundo” de um robô aspirador bastante simples, composto de 9 blocos nos quais ele pode se posicionar e identificar se há sujeira ou não em cada bloco (Figura 2). Como representação abstrata, os blocos são referenciados a partir de linhas e colunas, tal como uma matriz. Postulando-se que ele comece suas tarefas a partir do quadrado central, uma sequência de ações permite ao robô uma varredura para verificar se existe sujeira ou não, em cada quadrado que o robô for visitar (Medeiros, 2018, p. 37).

Figura 2 – Representação do ambiente de um robô aspirador



Fonte: Medeiros, 2018, p. 37.

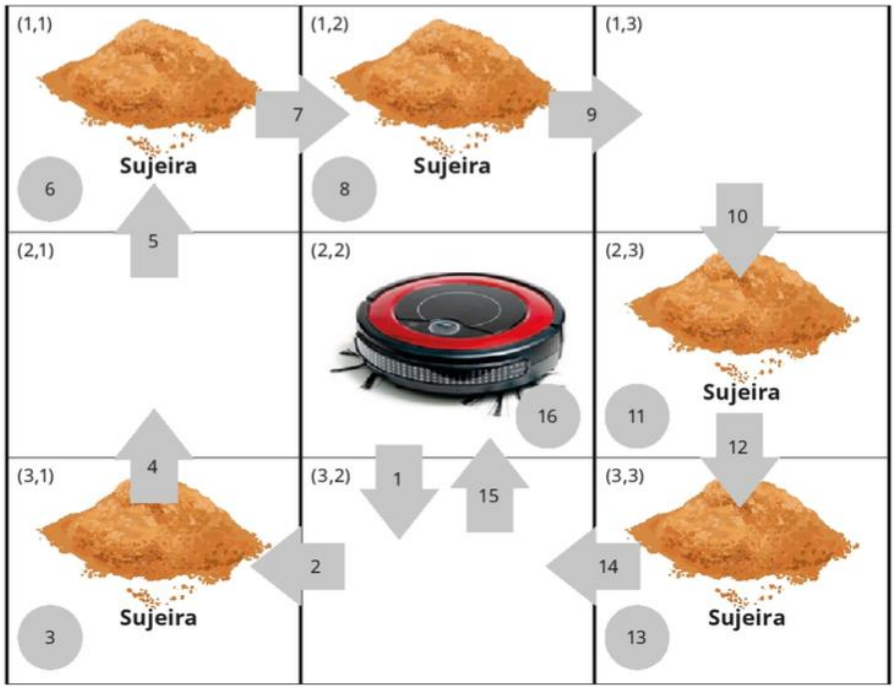
A partir dessa proposição inicial, pode-se imaginar uma série de formas pelas quais o robô poderá visitar cada uma das posições da matriz. Se o robô possui um sensor detector de sujeira e atuadores na forma de motores e rodas para auxiliar na sua movimentação, o robô pode trafegar pelos blocos. Essa configuração permite gerar diversas sequências de percepções. Na Tabela 1, consta uma sequência de percepções possível. Cada percepção é composta de: i) a posição (x,y) na matriz de onde o robô se encontra; e ii) o sinal proveniente do detector de sujeira. Seguindo a definição, o robô deve executar os movimentos feitos nessa sequência para que atinja a sua finalidade geral, que é a limpeza do seu "mundo". Essa sequência pode ser combinada com a representação do ambiente do agente, sendo ilustrado na Figura 3 como acontece essa sequência.

Tabela 1 – Descrição da função do agente para o exemplo do robô aspirador

Sequência	Percepção	Ação
1	[(2,2),Limpo]	Mova-se em frente
2	[(3,2),Limpo]	Mova-se à direita
3	[(3,1),Sujo]	Aspire a sujeira
4	[(3,1),Limpo]	Mova-se à direita
5	[(2,1),Limpo]	Mova-se em frente
6	[(1,1),Sujo]	Aspire a sujeira
7	[(1,1),Limpo]	Mova-se à direita
8	[(1,2),Sujo]	Aspire a sujeira
9	[(1,2),Limpo]	Mova-se em frente
10	[(1,3),Limpo]	Mova-se à direita
11	[(2,3),Sujo]	Aspire a sujeira
12	[(2,3),Limpo]	Mova-se em frente
13	[(3,3),Sujo]	Aspire a sujeira
14	[(3,3),Limpo]	Mova-se à direita
15	[(3,2),Limpo]	Mova-se à direita
16	[(2,2),Limpo]	Pare

Fonte: Medeiros, 2018, p. 38.

Figura 3 – Representação do ambiente do robô aspirador com o mapeamento de ações conforme a sequência de percepções



Fonte: Medeiros, 2018, p. 38.

Conclui-se, portanto, que há diversas formas pelas quais o robô aspirador pode executar a sua tarefa. Apesar de ele conseguir executar a tarefa de deixar o seu ambiente limpo, podemos nos questionar se o robô está mesmo sendo eficiente no desenvolver dessa tarefa. Tem-se, assim, a seguinte pergunta: existe uma maneira correta de o robô executar a tarefa? Existe um caminho pelo qual ele pode se movimentar de forma mínima para que o ambiente todo esteja limpo? Até agora, no projeto do robô, não houve preocupação quanto a isso. Entretanto, à medida que a função do agente se torna mais e mais complexa, isso precisa ser considerado, dado que o robô poderia se movimentar de forma interminável, sem deixar o ambiente limpo (por exemplo, o robô poderia entrar em *loop*, saindo do centro para um bloco lateral, um bloco diagonal, outro lateral adjacente e voltar novamente ao centro, e assim sucessivamente).

Para contornar esse tipo de situação, o agente, além de cumprir a tarefa (ou seja, ser eficiente), ele precisa ser também um **agente racional**: aquele que, a partir do resultado das suas ações, possa obter o maior nível de sucesso. Para se alcançar isso, é necessário considerar alguma metodologia para a quantificação do sucesso de um dado agente inteligente (Russell; Norvig, 2004).

Para cada sequência de percepção possível, um agente racional deve selecionar uma ação que se espera venha a **maximizar** sua medida de desempenho, dada a evidência fornecida pela sequência de percepções e por qualquer conhecimento interno do agente (Russell; Norvig, 2004, p. 36).

Assim, é necessário um indicador, ou um conjunto de medidas de desempenho, para se aferir a racionalidade do agente. A racionalidade vai depender em qualquer momento de quatro fatores:

- A **medida de desempenho** como critério para obtenção do sucesso da tarefa;
- O **conhecimento** prévio do agente com relação ao ambiente;
- As **ações** que o agente pode executar;
- A **sequência de percepções** que o agente tem até o momento (Russell; Norvig, 2004, p. 36, grifo nosso).

No exemplo do robô aspirador, poderia se considerar uma medida de desempenho a quantidade de quadrados limpos em um período de tempo, ou por algum ciclo de movimentação. O ambiente onde o robô executa a sua tarefa é perfeitamente conhecido, mas a sujeira pode aparecer de maneira aleatória nas células. O sensor de posição pode identificar o bloco onde ele se encontra (usando um leitor de posição) e identificando se existe sujeira ou não (por meio de uma câmera ou algum sensor que possa identificar a diferença de um bloco limpo ou sujo). Com essas funcionalidades agregadas, o robô aspirador poderia agora ser classificado como um **agente racional**.

TEMA 2 – RESOLUÇÃO DE PROBLEMAS POR BUSCA

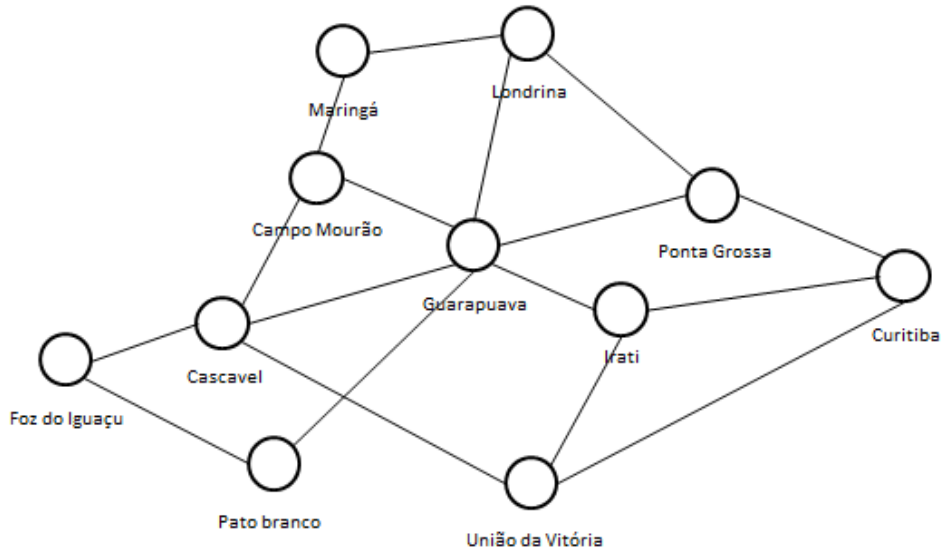
Os **agentes de resolução de problemas** tomam decisões sobre os próximos passos encontrando aquelas as sequências de ações que resultam nos estados desejáveis (Russell; Norvig, 2004, p. 62). Por exemplo, um dos problemas presentes em várias situações é o de roteirização. Esse problema está presente na área de logística e turismo, dentre outras. Quando se tem um conjunto de locais que precisam ser visitados considerando a existência de rotas definidas para esses locais, um dos problemas típicos é como percorrer todos os locais considerando um custo mínimo ótimo. Esse problema foi descrito inicialmente na IA como o **problema do viajante**.

Se considerarmos um mapa rodoviário, tal como o da Figura 4, se o objetivo é sair da cidade de Foz do Iguaçu em direção até a cidade de Curitiba, o agente

deve escolher aquelas rotas ou cursos de ação que permitem chegar ao destino, e aqueles que não permitem chegar em um tempo aceitável podem ser rejeitados, simplificando assim a tarefa do agente. Dessa forma, a **formulação de objetivos** se baseia na situação atual do agente e na medida de desempenho dele, o que configura o primeiro passo para a resolução do problema.

Aliado a esse passo, a **formulação de problemas** é outro passo essencial, pois é o processo de decidir quais estados e ações deverão ser considerados na abordagem do problema, dado um determinado objetivo (Russell; Norvig, 2004, p. 62). Supondo no exemplo anterior que o agente esteja considerando as ações de dirigir de uma cidade até a próxima, os estados da rota adotada corresponderão a estados do problema.

Figura 4 – Mapa rodoviário simplificado do estado do Paraná



Fonte: elaborado pelo autor.

Supondo que o agente esteja em Cascavel, ele considera qual caminho pegar. Ele pode decidir ir por Campo Mourão, Guarapuava ou por União da Vitória. Entretanto, nenhuma dessas cidades é o objetivo, que é o de chegar até Curitiba. Se o agente não tiver familiaridade com o trajeto, ele não saberá qual dos caminhos seguir. Em suma, o agente não poderá escolher um curso de ação porque não terá o conhecimento de qual estado resulta na execução de uma determinada ação. Caso ele não tenha nenhum conhecimento, se o agente não for programado para escolher aleatoriamente um caminho, ele ficará paralisado.

Se o agente possui um mapa, ele pode obter informações sobre os estados em que ele pode entrar e quais ações ele pode perfazer. O agente considera então

as possíveis rotas a partir das cidades consideradas. Dessa forma, um agente que possua uma série de opções imediatas de valor desconhecido pode decidir o que fazer avaliando primeiro as diferentes sequências de ações possíveis que vão levar a estados de valor conhecido, tendo condições de escolher a melhor sequência. O processo de procurar essa sequência é denominado de **busca** (Russell; Norvig, 2004, p. 62).

Um **algoritmo de busca** recebe na sua entrada um **problema** e apresenta na sua saída uma **solução**, descrita sob a forma de uma sequência de ações definidas. A partir da obtenção da solução, as ações recomendadas por ela são colocadas em **execução**. O algoritmo para a resolução de problemas simples pode ser então descrito nos seguintes passos (Russell; Norvig, 2004, p. 63):

- Formulação do objetivo e do problema;
- Busca de uma sequência de ações que resolvem o problema;
- Execução das ações uma por uma.

Quando a sequência se completa, ele formula outro objetivo e então recomeça. É importante observar que na execução da sequência, o agente não leva em conta as percepções, supondo na suposição de que a solução que encontrou na busca sempre vai funcionar.

Um problema pode ser definido de maneira formal em quatro componentes:

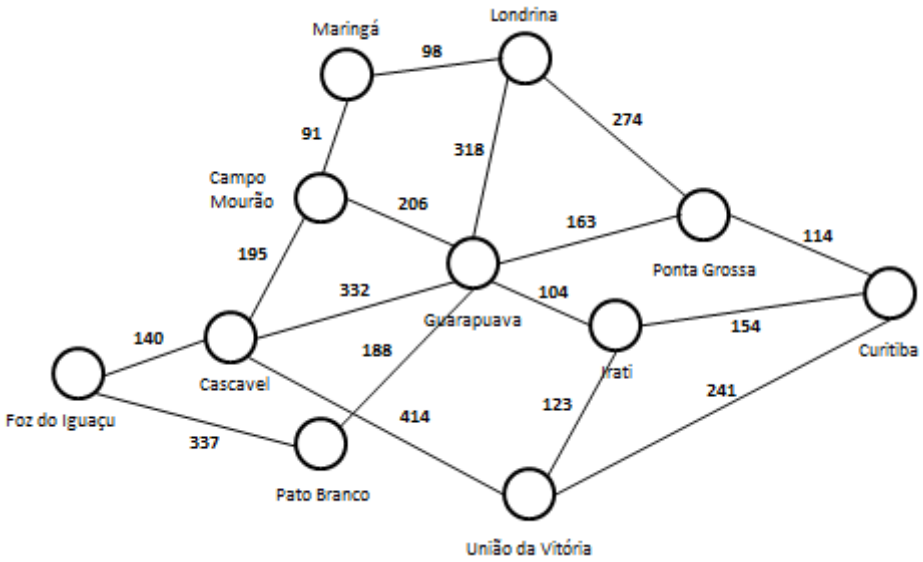
- Um **estado inicial**: o estado no qual o agente começa. Por exemplo, se o agente do exemplo da viagem estiver em Foz do Iguaçu, o estado inicial pode ser descrito como *Origem(Foz do Iguaçu)*;
- Uma **função sucessor**: dado um estado particular *x*, *SUCESSOR(x)* retorna um conjunto de pares ordenados <ação,sucessor>, em que cada ação é uma das ações válidas no estado *x* e cada sucessor pode ser alcançado partindo-se de *x*. Como exemplo, no caso do estado inicial ser *Origem(Foz do Iguaçu)*, a função sucessor para o problema retornaria
  {<Destino(Cascavel), Origem(Foz do Iguaçu)>,
  <Destino(Pato Branco), Origem(Foz do Iguaçu)>}

O estado inicial e a função sucessor definem o espaço de estados do problema, ou seja, o conjunto de todos os estados acessíveis tendo como partida o estado inicial. O espaço de estados representa um **grafo** onde os **nós** são os **estados** e os **arcos** entre os nós são as **ações**.

- O **teste de objetivo**: determina se um estado é um estado objetivo. O objetivo do exemplo é o estado final *Origem(Curitiba)*. Em alguns casos, um objetivo pode ser definido como uma propriedade abstrata e não por um estado ou conjunto de estados específicos. Por exemplo, num jogo de xadrez, o objetivo é alcançar o xeque-mate;
- A **função de custo**: também chamada de função de custo de caminho, que atribui um custo numérico a cada caminho. O agente vai escolher, portanto, uma função de custo que irá significar a sua própria medida de desempenho. No caso do agente do exemplo, a função de custo seria a distância em quilômetros (Figura 5). Há então um **custo de passo** para ir de um estado a outro.

Definidos os quatro elementos da formulação do problema, uma **solução** para um problema é um caminho que leva o agente desde o estado inicial até o estado objetivo. Considerando a qualidade da solução, a qual é medida pela função custo, uma **solução ótima** é aquela que apresenta o menor custo dentre todas as soluções possíveis (Russell; Norvig, 2004, p. 64).

Figura 5 – Mapa com as distâncias em quilômetros do Estado do Paraná



Fonte: elaborado pelo autor.

2.1 Exemplos de Problemas

Para efeito de estudo, os problemas podem ser classificados em:

- **Miniproblema**: destina-se a ilustrar ou praticar métodos de resolução de problemas, podendo ter descrições concisas e exatas. Pode ser utilizado por diversos métodos para comparação de desempenho;
- **Problemas do Mundo Real**: são aqueles que não tendem a ter uma descrição concisa e exata, abstraída de situações da realidade. Em geral são problemas complexos, que podem ser divididos em problemas simples que possuem um método de solução conhecida. Essa estratégia é denominada de “dividir-para-conquistar”.

2.1.1 Miniproblemas

O primeiro miniproblema é o exemplo do robô aspirador, descrito em detalhes na Tabela 2, tratado no tópico sobre agentes inteligentes.

Tabela 2 – Formulação do miniproblema do robô aspirador

Robô Aspirador	
Formulação	***Descrição***
Estados	O agente se posiciona em uma de 9 posições diferentes que podem conter sujeira ou não. A quantidade de espaços possíveis (contando que todos contenham sujeira) é de 9 x 2 estados mais 2 do estado inicial resultando em 20 estados.
Estado inicial	Robô na posição (2,2)
Função sucessor	Gera os estados válidos resultantes da tentativa de ação (Para-frente, Para-direita, Aspirar). Ver na figura 6.
Teste de objetivo	Verificar se todos os quadrados estão limpos.
Custo de caminho	Cada passo custa 1 unidade; o custo do caminho é o número de passos.

Fonte: elaborada pelo autor.

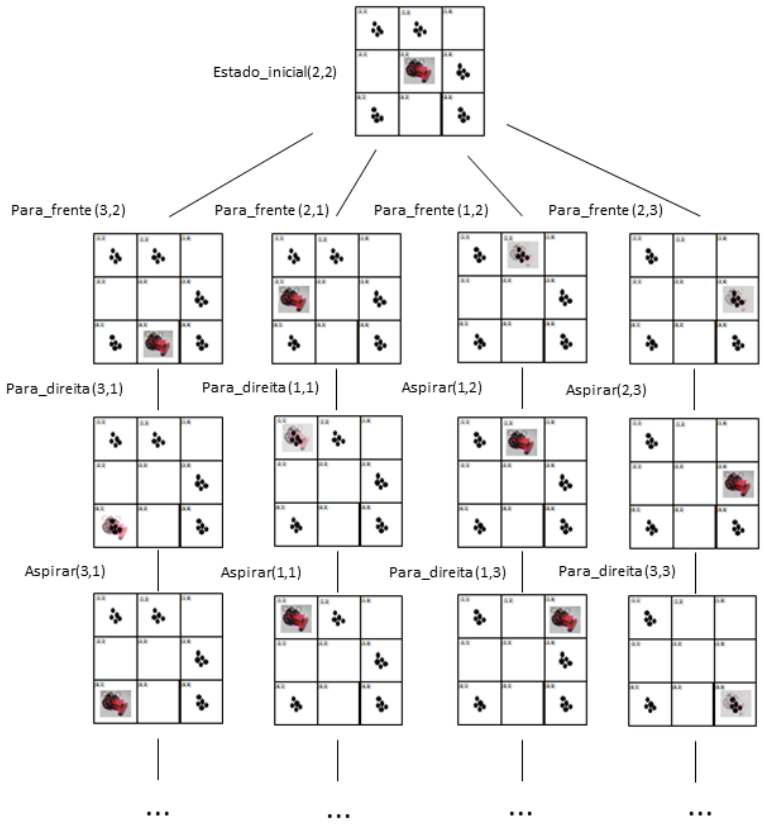
Esse miniproblema tem as posições discretas, sujeira discreta, limpeza confiável e não é desorganizado após o término da tarefa.

O *puzzle* (quebra-cabeças) de oito peças consiste em um pequeno tabuleiro de tamanho 3 x 3 casas, com peças numeradas de 1 a 8 e um espaço vazio para permitir um deslocamento de uma peça. O objetivo do puzzle de 8 peças é atingir um determinado estado (Tabela 3).

Tabela 3 – Formulação do miniproblema do puzzle de oito peças

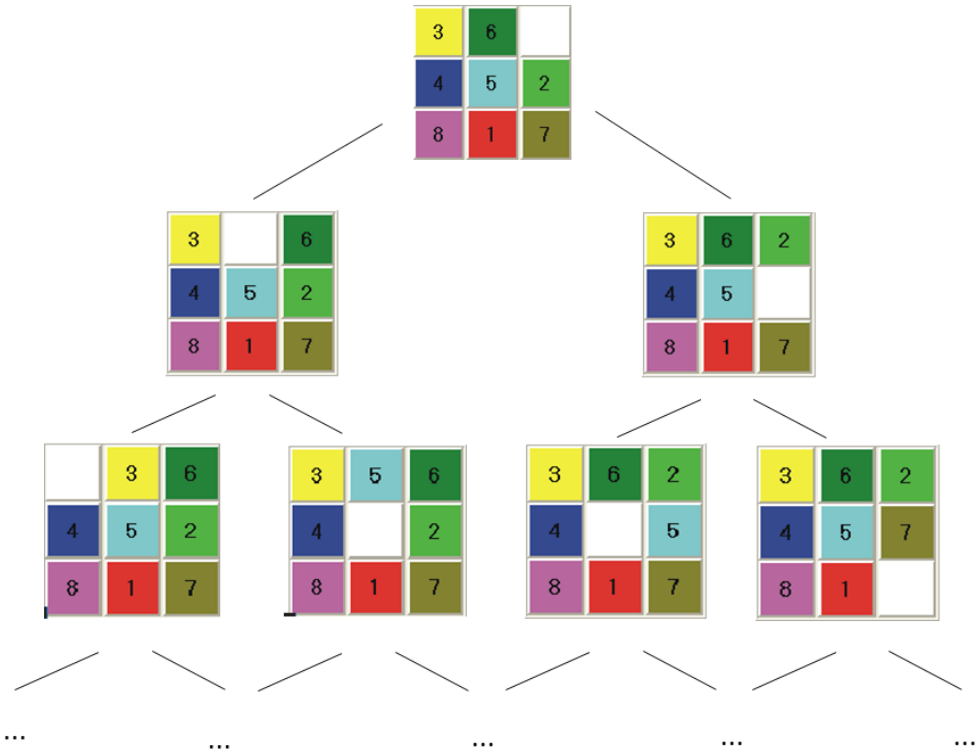
Puzzle de oito peças		
Formulação	***Descrição***	***Exemplo***
Estados	Uma disposição específica das 8 peças sobre o tabuleiro 3x3.	
Estado inicial	Qualquer estado pode ser definido como o estado inicial (Ver figura 7).	
Função sucessor	Gera os estados válidos para execução das quatro ações possíveis: Esquerda, Cima, Baixo, Direita.	
Teste de objetivo	Verifica se o estado alcançado é o mesmo que foi definido como objetivo.	
Custo de caminho	Cada passo custa 1 unidade; o custo do caminho é o número de passos.	

Figura 6 – Expansão da árvore de estados do problema do robô aspirador



Fonte: elaborado pelo autor.

Figura 7 – Expansão da árvore de estados do problema do puzzle de oito peças



Fonte: elaborado pelo autor.

O *puzzle* de oito peças pertence à família de quebra-cabeças deslizantes, utilizados com frequência em testes de algoritmos de IA. O quebra-cabeça de oito peças possui 181.440 estados acessíveis e pode ser resolvido com facilidade. O puzzle de 15 peças (um tabuleiro 4x4) possui cerca de 1,3 trilhão de estados. Já o puzzle de 24 peças (tabuleiro 5x5), possui aproximadamente 10^{25} estados, sendo ainda bastante difícil de resolver de forma ótima com computadores e algoritmos atuais (Russell; Norvig, 2004, p. 67).

TEMA 3 – ESTRUTURA DE AGENTES E PROPRIEDADES DOS AMBIENTES

Russell e Norvig (2004, p. 39) assinalam que uma das primeiras atividades relativas à modelagem de um agente se refere à definição do **ambiente de tarefa**. O ambiente de tarefa é composto de quatro elementos: desempenho, ambiente, atuadores e sensores. No inglês, assume o acrônimo **PEAS** (*Performance, Environment, Actuators and Sensors*). No Quadro 1, estão alguns exemplos de caracterização do PEAS para diferentes situações: sistema de diagnóstico médico, jogo de xadrez, tutor inteligente e robô de solda.

Quadro 1 – Exemplos de agentes racionais e as descrições do PEAS

Tipo de Agente	Medida de desempenho	Ambiente	Atuadores	Sensores
Sistema de diagnóstico médico	Paciente saudável, minimizar custos.	Quarto de hospital, paciente, equipe.	Mostrar perguntas, testes, diagnósticos, tratamentos.	Entrada por teclado, voz, descoberta de informação.
Sistema de xadrez	Calcular a jogada ótima, ganhar a partida.	Tabuleiro 8x8, peças de xadrez, posições iniciais, jogadas possíveis.	Braço articulado para mover as peças.	Câmera para ver as posições das peças.
Tutor inteligente	Maximizar as notas dos alunos nos testes.	Área de interação do software, chat de mensagens.	Mostrar conteúdos, exibir exercícios.	Entrada pelo teclado.
Robô de solda	Efetuar os pontos de solda no tempo previsto, qualidade da solda.	Linha de montagem.	Braço articulado para movimentos no espaço, ponteira de solda.	Identificação de novo objeto a ser soldado, quantidade de solda, temperatura de solda.

Fonte: adaptado de Russell; Norvig, 2004, p. 40.

Dentro do PEAS, os ambientes ainda podem receber diferentes classificações conforme uma série de critérios (Russel; Norvig, 2004, p. 41-42):

- **Completamente observável *versus* Parcialmente observável**: No caso de os sensores acessarem de forma completa os estados do ambiente em cada instante, o ambiente é completamente observável. Mas se houver ruído, sensoriamento impreciso ou lacunas, será parcialmente observável. Ex.: Um jogo de xadrez á completamente observável. Já um sistema de busca na internet é parcialmente observável;
- **Determinístico *versus* Estocástico:** Caso o estado seguinte seja completamente determinado a partir do estado atual e ainda pela ação executada pelo agente, o ambiente é determinístico, senão, estocástico. No caso de o sistema ser determinístico, porém apresentando elementos estocásticos, o ambiente pode ser **estratégico**. Ex.: O robô aspirador de pó é determinístico; um carro com direção autônoma é estocástico;
- **Episódico *versus* Sequencial**: Num ambiente episódico o agente experimenta os eventos de forma atômica, com os episódios começando a partir da percepção do agente e pela execução **de** uma única ação. No caso

de um ambiente sequencial, existe a dependência dos estados atuais com os estados anteriores. Ex.: Um jogo de xadrez é sequencial. Um robô aspirador de movimento aleatório é episódico;

- **Estático *versus* Dinâmico**: Um ambiente que se altere enquanto o agente executa a tarefa é dinâmico. Caso o **ambiente** não se altere, será estático. No entanto, pode haver situações em que os ambientes sejam **semidinâmicos**. Ex.: um jogo de xadrez é estático. Já um carro com direção autônoma é dinâmico (ou **semidinâmico**, considerando as rotas predefinidas);
- **Discreto *versus* Contínuo**: Depende da forma com que o tempo decorre, do estado do ambiente e das percepções e ações. Uma sequência de estados discretos muda aos “saltos” de um estado para outro. Uma sequência de estados contínua muda de maneira suave. Ex.: Um jogo de xadrez é considerado discreto. Um sistema de refrigeração de temperatura pode ser contínuo;
- **Indivídual *versus* Multiagente**: Refere-se à quantidade de **agentes** utilizados para a resolução de um problema. Caso exista somente um agente é individual, senão o sistema será considerado multiagente. Ex.: Um jogo de xadrez pode ser um agente individual; no caso de um agente jogar contra outro, é multiagente. Já um sistema de busca na internet pode ser multiagente por utilizar vários robôs que encontram e organizam a informação.

No quadro 2 da próxima página estão alguns exemplos a respeito da caracterização da natureza do ambiente.

Quadro 2 – Alguns exemplos de caracterização do ambiente de tarefa

Ambiente de tarefa	Observável	Determinístico	Episódico	Estático	Discreto	Agente
Jogo de palavras cruzadas	Completamente	Determinístico	Sequencial	Estático	Discreto	Único
Xadrez com um relógio	Completamente	Estratégico	Sequencial	Semi	Discreto	Multi
Pôquer	Parcialmente	Estratégico	Sequencial	Estático	Discreto	Multi
Gamão	Completamente	Estocástico	Sequencial	Estático	Discreto	Multi
Direção de táxi	Parcialmente	Estocástico	Sequencial	Dinâmico	Contínuo	Multi
Diagnóstico médico	Parcialmente	Estocástico	Sequencial	Dinâmico	Contínuo	Único
Analista de imagens	Completamente	Determinístico	Episódico	Semi	Contínuo	Único
Robô de seleção de peças	Parcialmente	Estocástico	Episódico	Dinâmico	Contínuo	Único
Controlador de refinaria	Parcialmente	Estocástico	Sequencial	Dinâmico	Contínuo	Único
Instrutor interativo de inglês	Parcialmente	Estocástico	Sequencial	Dinâmico	Discreto	Multi

Fonte: elaborado pelo autor.

TEMA 4 – TIPOS DE AGENTE INTELIGENTE

O estudo do tipo de agente, ao invés da caracterização do ambiente externo ao agente, preocupa-se com seus aspectos internos. O objetivo agora é definir como a função do agente deverá trabalhar, e por consequência o programa do agente. O agente precisa, portanto, conter uma arquitetura que permita implementar o programa do agente, a partir de um conjunto de sensores e atuadores, conectados a um processador que irá executar o programa do agente (Russell; Norvig, 2004, p. 44).

De acordo com a sua arquitetura, um agente poderá ser classificado em cinco tipos (Russell; Norvig, 2004, p. 48):

1. Agentes reativos simples;
2. Agentes reativos baseados em modelo;
3. Agentes baseados em objetivos;
4. Agentes baseados em utilidade;
5. Agentes com aprendizagem.

4.1 Agentes reativos simples

É o tipo mais simples de agente. O programa do agente sempre vai mapear as ações com base na percepção atual, descartando a sequência de percepções anteriores.

A implementação de programas para esse tipo de agente pode ser feita utilizando-se **regras de produção** ou **regras se-então**. A execução dessas regras acontece de forma determinística e, por meio de teste de condições, serão executadas as ações, seguindo o conjunto de regras que estão implementadas.

SE <condição> **ENTÃO** <ação>

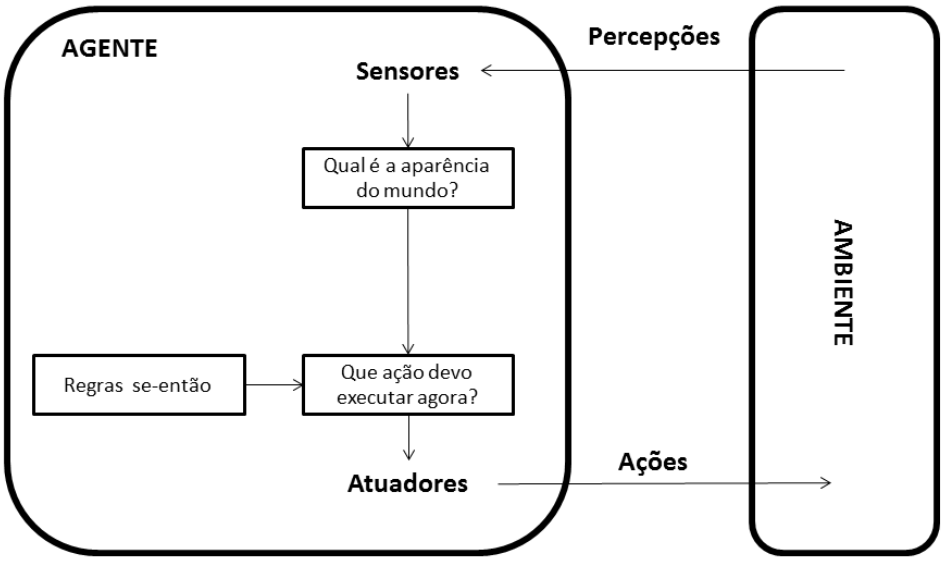
O robô aspirador do exemplo mencionado anteriormente poderia conter quatro regras que gerariam o comportamento desejado (Medeiros, 2018, p. 44):

1. **SE** *estado = Sujo* **ENTÃO** *Aspirar*
2. **SE** *estado = Limpo* **ENTÃO** *faça movimento = SIM* **SENÃO** *faça movimento = NÃO*
3. **SE** *faça movimento = SIM* **E** *quadrado na frente = SIM* **ENTÃO** *direção do movimento* = FRENTE
4. **SE** *faça movimento = SIM* **E** *quadrado na frente* = NÃO **ENTÃO** *direção do movimento = DIREITA*

Nesse sistema de quatro regras, a regra 1 se encarrega da limpeza do quadrado por meio da variável estado. No caso de o quadrado estar limpo então, então a variável faça movimento recebe o valor SIM, o que é feito pela regra 2. Com isso, as regras 3 e 4 ficam habilitadas para operarem. É testada então a variável “quadrado na frente” (ou seja, armazena o valor do sensor que identifica o quadrado). Se houver um quadrado à frente, então a variável de ação direção do movimento recebe o valor FRENTE. Caso não houver (ou seja, o quadrado estará à direita), então a variável direção de movimento recebe o valor DIREITA.

De maneira geral, a arquitetura de um agente reativo simples é mostrada na Figura 8. A partir das percepções do ambiente, o agente se pergunta: “qual é a aparência do mundo?”; esta aparência irá consistir em um conjunto de condições que irão “disparar” as regras que conseguirem atender e assim a ação a ser executada será selecionada e levada aos atuadores.

Figura 8 – Arquitetura para um agente reativo simples

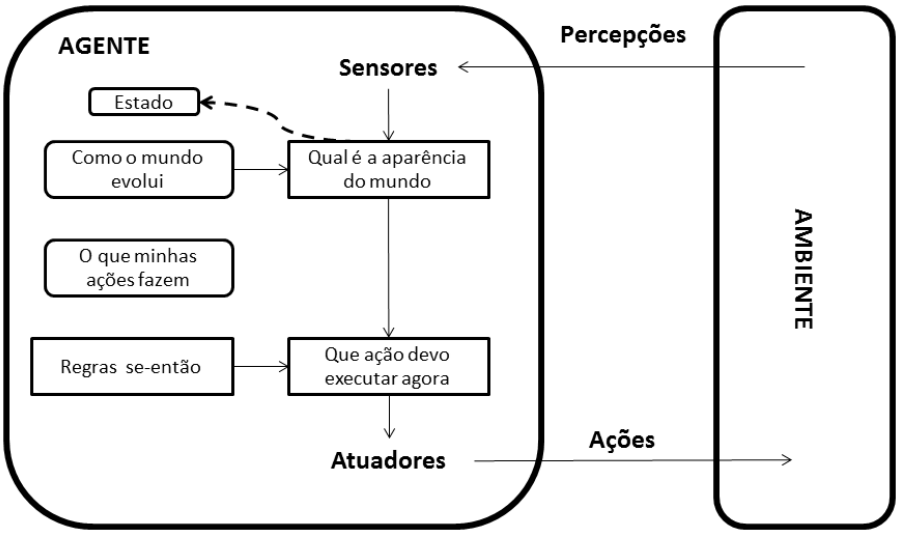


Fonte: Baseado em Russell; Norvig, 2004, p. 47.

4.2 Agentes reativos baseados em modelo

No caso de um ambiente ser caracterizado como observável parcialmente, é necessário que o robô mantenha estados internos que dependam da sequência de percepções para atuar de maneira mais efetiva. Por exemplo, se o robô aspirador tiver uma memória onde possa armazenar um histórico de percepções, o robô terá informação em quais blocos apareceram mais sujeira que outros, podendo começar os movimentos por esses blocos. Um agente que tem o conhecimento de "como o mundo funciona" possui um **modelo** de mundo. Um agente que tenha um modelo de mundo é denominado de agente reativo baseado em modelo (Figura 9).

Figura 9 – Arquitetura para um agente reativo baseado em modelo



Fonte: baseado em Russell; Norvig, 2004, p. 49.

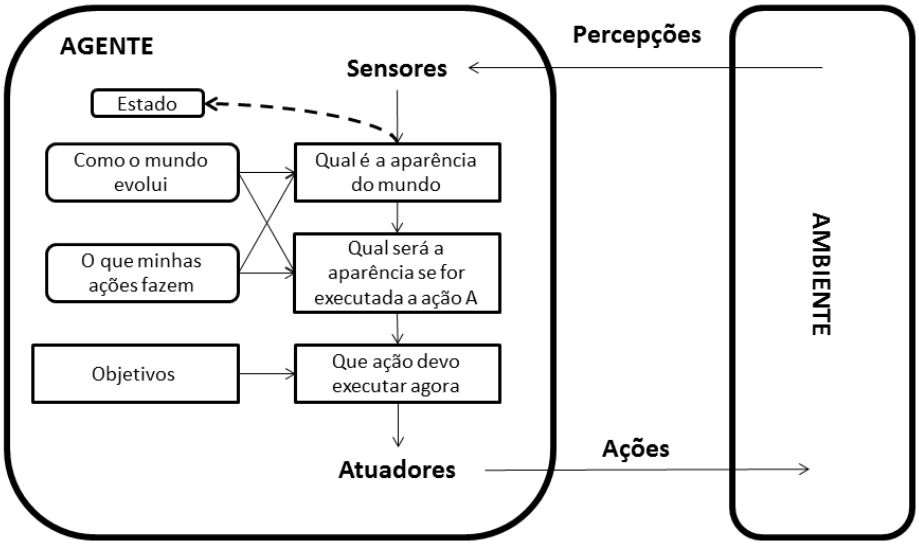
4.3 Agentes baseados em objetivos

Apenas o conhecimento do estado atual do ambiente pode não ser o suficiente para que um robô decida o que fazer. Além de saber a descrição do estado atual e da sequência de percepções obtidas até o momento, pode ser necessária a definição de algum objetivo que estejam relacionados a situações ou cenários desejados (Russell; Norvig, 2004, p. 49).

A tomada de decisão baseada em objetivos é diferente do uso de regras de produção, pois está envolvida uma situação a ser alcançada, uma situação de futuro. O agente então se pergunta: “o que vai acontecer se eu fizer esta ação ou aquela outra?” ou “isso vai me fazer feliz?”. Tais informações nos agentes reativos não são representadas de maneira explícita, devido ao mapeamento direto das percepções para as ações (Figura 10). Ainda que o agente baseado em objetivos possa parecer menos eficiente, ele tende a ser mais flexível (Russell; Norvig, 2004, p. 49).

Voltando ao exemplo do robô aspirador, pode ser colocado em seu projeto um objetivo que leve em conta o custo de operação ou consumo de energia. Com a execução contínua, o robô pode antecipar a frequência com que as limpezas são feitas, e nos intervalos ele pode entrar em modo de economia de energia para alcançar este objetivo (Medeiros, 2018, p. 46).

Figura 10 – Esquema de um agente baseado em objetivo



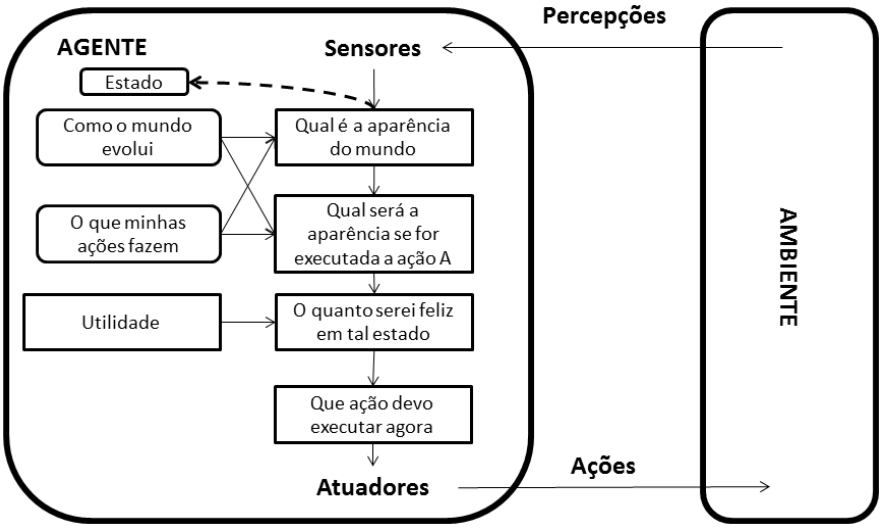
Fonte: baseado em Russell; Norvig, 2004, p. 50.

4.4 Agentes baseados em utilidade

Como saber se um agente está sendo útil a partir de uma medida? Até agora, os agentes anteriores não estavam preocupados com este aspecto. Pode-se especificar uma espécie de medida de utilidade: conforme a avaliação de uma função de utilidade, o agente ou robô poderá tomar decisões mais adequadas (Figura 11). Pode ser que o robô tenha objetivos contraditórios e apenas alguns deles podem ser alcançados em dado momento. Ou pode haver vários objetivos, mas nenhum deles pode ser alcançado com alto grau de certeza (Russell; Norvig, 2004, p. 50).

Por exemplo, um robô seguidor de linha pode conter um sensor que permita identificar se existe um bloqueio intermitente na trilha para a execução do movimento. Portanto, além da avaliação das regras conforme o sensoriamento da trilha, deverá também monitorar a todo momento de há algum objeto bloqueando a trilha. No caso de haver mais de uma trilha a ser percorrida, o robô poderá executar aquela em que uma função de avaliação minimize o percurso, ou maximize o alcance do objetivo.

Figura 11 – Esquema de um agente baseado em utilidade



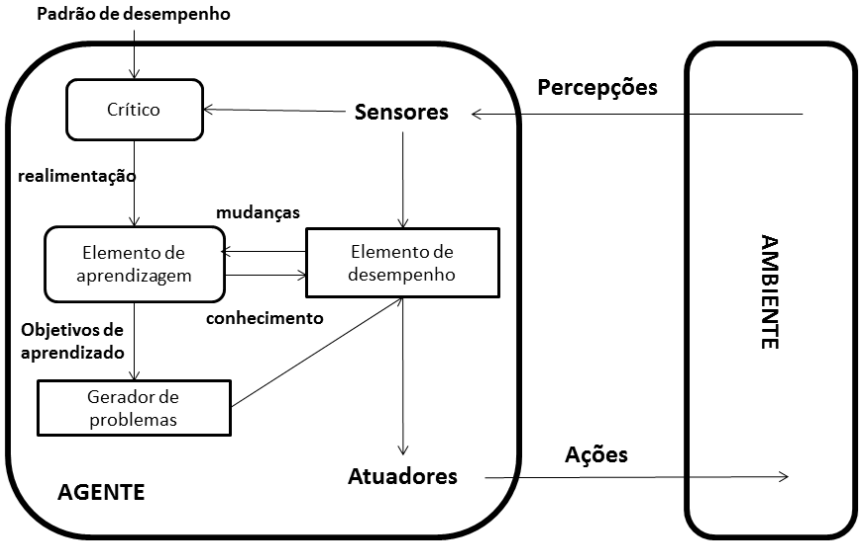
Fonte: baseado em Russell; Norvig, 2004, p. 51.

4.5 Agentes com aprendizagem

O agente com aprendizagem permite ir além do conhecimento prévio na construção do agente, dotando-o de mecanismos para que possa aprender na experiência com o ambiente, tornando-se mais competente ao longo da sua operação (Figura 12). Um agente de aprendizado pode ser dividido em quatro elementos (Russell; Norvig, 2004, p. 51-52):

1. **Elemento de desempenho**: é a parte considerada até agora sobre agentes, que recebe as percepções e decide qual ação executar;
2. **Crítico**: elemento do agente que informa como o agente está se comportando em relação ao seu padrão constante de desempenho;
3. **Elemento de aprendizado**: utiliza a informação proveniente do crítico para modificar o elemento de desempenho em direção a um melhor funcionamento no futuro;
4. **Gerador de problemas**: elemento responsável para sugestão de novas regras e ações que podem levar a novas experiências.

Figura 12 – Esquema de um agente com aprendizagem



Fonte: baseado em Russell; Norvig, 2004, p. 52.

TEMA 5 – ALGORITMOS GENÉTICOS

As pesquisas de IA dentro da linha evolucionária se baseiam na observação de mecanismos evolutivos da natureza, incluindo auto-organização e comportamento adaptativo. Os modelos mais conhecidos de algoritmos evolucionários são os algoritmos genéticos, a programação genética e as estratégias evolucionárias.

Um algoritmo genético (AG) faz parte da classe de algoritmos de busca. O algoritmo procura uma solução dentro de um espaço para um problema de **otimização**. Assim, os algoritmos genéticos podem ser uma boa opção para efetuar a busca em problemas considerados intratáveis. A aplicação de algoritmos genéticos se faz em várias áreas tais como a arquitetura de circuitos eletrônicos, planejamento e roteirização, programação de games, previsão do tempo e ainda descoberta de identidades matemáticas (Russell; Norvig, 2004).

Podemos afirmar que um AG é considerado um algoritmo de busca em feixe estocástica, onde os estados sucessores são criados a partir da combinação de dois (ou mais) estados “pais”, ao invés de serem criados a partir de variação de um único estado. Temos assim uma analogia com a seleção natural proveniente da biologia. A busca em feixe é devido ao uso de uma população inicial gerada aleatoriamente, que evolui ao longo das “gerações” (iterações do algoritmo). A produção de uma nova população é avaliada por uma **função**

objetivo, ou **função de fitness**. Essa função retorna a nova população priorizando os estados melhores (Russell; Norvig, 2004, p. 115).

Dos principais fatores que têm feito dos AG uma técnica de busca bem-sucedida e utilizada, estão (Azevedo; Brasil; Oliveira, 2000, p. 35):

- Simplicidade da operação;
- Facilidade de implementação;
- Eficácia na busca da região onde provavelmente está o máximo global;
- Aplicável na situação em que se tem pouco ou nenhum conhecimento do modelo, ou este é impreciso.

Pelas suas características, os AG fazem parte dos métodos probabilísticos de busca e otimização. Os AG utilizam o conceito de probabilidade, mas não são considerados simples buscas aleatórias. Ao contrário, os AG direcionam a busca para regiões onde é mais provável encontrar uma solução ótima. Diferente das técnicas de busca convencionais, as principais diferenças residem em (Azevedo; Brasil; Oliveira, 2000, p. 37):

- A busca da melhor solução é feita sobre uma população de pontos e não sobre um único ponto (busca em feixe);
- Os AG perfazem uma busca cega, sendo a única exigência o conhecimento da função objetivo de cada indivíduo, não sendo necessária nenhuma informação adicional;
- Os AG usam operadores estocásticos ou probabilísticos e não regras determinísticas na direção de uma busca altamente exploratória e estruturada; as informações das gerações anteriores são acumuladas, auxiliando a direcionar estas buscas.

De forma geral, os passos para a execução de um AG são:

1. Cria-se uma população aleatória de candidatos a solução (*pop*);
2. Enquanto as condições de terminação não são satisfeitas: (cada iteração e geração):
   (a) Cria uma nova população vazia (*new-pop*).
   (b) Enquanto *new-pop* não estiver completa:
      i. Faz a **seleção** de dois indivíduos aleatoriamente de pop (dando preferência na seleção aos indivíduos de maior **fitness**);
      ii. Faz o **crossover** os dois indivíduos para obter dois novos indivíduos;

(c) Dá a cada membro de new-pop a chance de mutação;

(d) Substitui pop com new-pop.

3. Seleciona o indivíduo da população com a melhor **fitness** como a solução do problema.

A **população** é a coleção de candidatos a soluções que se estão considerando durante o uso do algoritmo. Pelas gerações do algoritmo, novos membros “nascem” na população enquanto outros “morrem” (saem da população). Uma simples solução na população é referenciada como um **indivíduo**. A adaptação ou adequação (**fitness**) de um indivíduo é uma medida de quão “boa” é a solução representada pelo indivíduo. Quanto melhor a solução, maior será a sua adaptação. É claro que isso depende das características do problema a ser resolvido.

Os indivíduos que fazem parte de um AG precisam de uma codificação que o simbolize no espaço possível de indivíduos que caracterizam o problema. Essa codificação é feita utilizando-se **cromossomos**. Um cromossomo é a tradução das características do indivíduo no **alfabeto** utilizado pelo AG. Podemos concluir que um cromossomo é uma sequência de bits, sendo constituídos por **genes** que se referem a cada bit (Linden, 2012, p. 65). Os algoritmos geralmente utilizam como alfabeto a codificação binária (0’s e 1’s) para sua representação. Um cromossomo terá também um comprimento fixo de bits ou genes.

Os operadores mais comuns utilizados nos AG para criar os novos indivíduos por meio das gerações são:

- Seleção;
- Cruzamento ou *crossover;*
- Mutação.

O processo de **seleção** é análogo à sobrevivência dos mais adaptados no mundo natural. Indivíduos mais aptos (aqueles que atendem ao melhor *fitness*) são selecionados para “procriação”. Utiliza um critério de maior probabilidade de cruzamento priorizando os que possuem maior fitness.

Um dos métodos mais utilizados para perfazer a seleção é o **método da roleta viciada** ou **roleta ponderada**. Esse método dá mais probabilidade para que um indivíduo de maior fitness seja escolhido em contrapartida a outro elemento que tenha menor fitness. Isso não quer dizer que os indivíduos que

possuam menor fitness não passem para a próxima geração, mas que a probabilidade de fazerem parte da nova geração é reduzida.

O **crossover** ou cruzamento ocorre pela mistura de duas soluções ou **indivíduos**, com o objetivo de criar dois novos indivíduos. Esse cruzamento tende a formar novos indivíduos que possuem características dos “pais”, e que tem a possibilidade de atender melhor o *fitness*. O **crossover** atua considerando dois indivíduos pais, assumindo um ponto de corte em seus cromossomos e intercalando as partes resultando em novos indivíduos. AG podem ter pontos de corte fixos ou aleatórios ao longo das gerações.

Durante cada geração, existe uma pequena chance de que um indivíduo dentro da população sofra uma **mutação**, que vai mudar a característica do indivíduo levemente. A mutação embute no AG uma característica totalmente aleatória. Conforme uma taxa especificada no algoritmo, alguns genes dentro dos cromossomos são escolhidos de forma randômica e alterados para outros valores permitidos pelo alfabeto do cromossomo.

O **tamanho da população** é outra variável a se considerar. Quanto maior a população, maior a quantidade de soluções possíveis, o que significa maior variação da população. A diversidade de variações permite que melhores soluções, mais próximas da solução ótima, sejam criadas. Assim, a população deve ser a maior possível. O que limita o tamanho da população é o **tempo** que vai demorar o AG na sua execução, e a quantidade de **memória** para guardar a população.

Qual a melhor forma de terminar a execução do AG? A abordagem mais simples é rodar o algoritmo de busca por um **número fixo de gerações** – quanto mais gerações, melhor. Outra abordagem é encerrar o algoritmo se, depois de passadas algumas gerações, não se obtém uma melhora na adaptação do melhor indivíduo da população. Em outras palavras, a variação dos melhores indivíduos obtidos para os que são criados novamente é mínima. Dessa forma, adotar um **limiar de variação** pode reduzir o número de gerações na execução de um AG.

REFERÊNCIAS

AZEVEDO, F. M.; BRASIL, L. M.; OLIVEIRA, R. C. L. **Redes neurais com aplicações em controle e em sistemas especialistas**. Florianópolis: Visualbooks, 2000.

LINDEN, R. **Algoritmos genéticos**. Rio de Janeiro: Ciência Moderna, 2012.

MEDEIROS, L. F. de. **Inteligência artificial aplicada**: uma abordagem introdutória. Curitiba: Intersaberes, 2018.

RUSSELL, S.; NORVIG, P. **Inteligência Artificial**. 2. ed. Rio de Janeiro: Campus, 2004.