UNIVERSIDADE REGIONAL DE BLUMENAU
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE CIÊNCIAS DA COMPUTAÇÃO
(Bacharelado)

PROTÓTIPO DE UM SIMULADOR DE UM ASPIRADOR DE PÓ, UTILIZANDO ALGORITMO DE BUSCA E AGENTES INTELIGENTES, EM AMBIENTES COM BARREIRAS

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS DA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO - BACHARELADO

JUSSARA VIEIRA RAMOS

BLUMENAU, NOVEMBRO/1999

PROTÓTIPO DE UM SIMULADOR DE UM ASPIRADOR DE PÓ, UTILIZANDO ALGORITMO DE BUSCA E AGENTES INTELIGENTES, EM AMBIENTES COM BARREIRAS

JUSSARA VIEIRA RAMOS

ESTE TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS DA DISCIPLINA DO TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIO PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Roberto Heinzle - Orientador na FURB

_____

Prof. José Roque Voltolini da Silva - Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Roberto Heinzle

_____

Prof. Antônio Carlos Tavares

_____

Prof. Maurício Capobiano Lopes

À NI1LTON e ORENICE,
meus pais. Amo-os de mais.

À quem amo tanto,

RENE.

AGRADECIMENTOS

Ao meu orientador, Prof. Roberto Heinzle principalmente pela eficiente orientação ao longo do desenvolvimento deste trabalho.

A todos os professores e funcionários do Departamento de Sistemas e Computação, em especial ao seu corpo docente pelos ensinamentos oferecidos.

Ao Prof. Antônio Carlos Tavares, com muito carinho e admiração, pelas conversas e trocas de idéias ao longo do curso.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS

BNF - *Bachus-Naur Form*

IA - Inteligência Artificial

IC - Inteligência Computacional

PDAs - Ambientes Digitais Pessoais

RESUMO

Este trabalho descreve um protótipo de um simulador de um robô aspirador de pó autônomo para ambientes diversos com barreiras utilizando técnicas da inteligência artificial para a solução do problema. O simulador é constituido de um agente inteligente composto de duas camadas, uma ativa e outra reativa. A camada ativa, encapsula um sistema inteligente rotineiro e cognitivo, responsável pelo cumprimento de sua diretiva básica: aspirar o pó de todo o ambiente possível. A camada reativa, possui a habilidade de prever e evitar possíveis acidentes, exatamente por ser o ambiente dinâmico e adverso. A camada reativa, por sua vez, é composta de quatro sensores, colocados no aspirador de pó, responsáveis por obter os dados do ambiente, repassando para o simulador que define a direção do aspirador de pó e, quando necessário, utiliza a técnica de algoritmo de busca para escolha do melhor percurso para uma nova posição a ser aspirada, tal responsabilidade faz parte da camada ativa.

ABSTRACT

This work describes a prototype of an auto simulator of vacuum clear robot for divers emvironment with barriers utilizing technics artificial intelligence for the problem solution. The simulator is consistuted of an inteligent agent compound of two layers, one active and other reactive. The active layer, consist of an intelligent system rotiner and congnitive, responsible for the execution of its basic procedure: to aspirate dust of all environment possible. The reactive layer, posses the hability to foresee and to avoid possible accidents, exactly for been the environment dinamic and adverse. The reactive layer, is composed of four sensors, placed in the vaccum clear, responsible for obtain datum of the environment, repassing to the simulater than defines the direction of the vaccum clear and, when necessary, utilizes the technic of algoritm search to choose the best route for a new position to aspire, this responsibility is part of the active layer.

1 INTRODUÇÃO

1.1 INTELIGÊNCIA ARTIFICIAL X ROBÓTICA

O "sonho" de construir máquinas inteligentes não é novo. O homem, desde a antigüidade, procurou desenvolver dispositivos artificiais dotados de movimento e inteligência [TAF96]. O campo de pesquisa é praticamente ilimitado. O conhecimento e a inteligência se estendem a, praticamente, todos os campos de atuação da humanidade, inclusive a domínios até então desconhecidos que ganham perspectivas através de novas ferramentas disponíveis [TAF96].

O número de trabalhos e de produtos comerciais, que utilizam essa tecnologia, têm crescido e influenciado novos comportamentos tecnológicos [TAF96]. Há robôs meramente mecânicos, capazes apenas de reproduzir tarefas e movimentos implícitos em sua construção. Há, contudo, robôs que complementam a parte mecânica com dispositivos de suporte (eletrônicos), constituindo uma espécie de cérebro, onde são armazenados conhecimentos, os quais podem dar certo grau de autonomia a estes engenhos. Este último tipo de rôbo tem sido usado, em geral, nas tarefas executadas em ambientes hostis ao ser humano, como é o caso de viagens espaciais, atividades de prospecção de petróleo no fundo dos oceanos, etc [RAB95].

Enquanto humanos realizam uma grande variedade de coisas usando mais ou menos o mesmo corpo, muitos projetos de rôbos dependem da tarefa para a qual foram pretendidos [RAB95].

Rôbos móveis estão também se tornando largamente proveitosos. Duas aplicações iniciais, como exemplo, são mensageiros em edifícios, especialmente em hospitais, e como guardas de segurança [RAB95]. Robôs móveis são uma importante tecnologia para reduzir o risco de vida humano em ambientes arriscados [RAB95].

De fato, rôbos autônomos tem crescido e se aplicado a cada dia. Mas ainda assim, depara-se com o problema de reconhecimento de ambientes para o caminhamento de rôbos autônomos. Este trabalho vem justamente propor uma solução nesse sentido.

1.2 O PROBLEMA PROPOSTO / ORIGEM DO TRABALHO

Com a crescente necessidade de tempo humano sendo imposta pelo mercado profissional, a necessidade de automação de tarefas simples e ou perigosas tornou-se uma máxima. A tarefa de aspirar pó aparentemente simples é um dos muito exemplos que ainda necessita de uma inteligência, para operação e tomadas de decisões, de forma a desviar o aspirador de pó de obstáculos, reagindo a barreiras e a localização de lugares a serem ainda aspirados. De fato, existem diversos modelos desta ferramenta, mas os mesmos ainda necessitam de um operador humano.

É conhecida uma implementação de fato, pela empresa Electrolux [ELE98], de um aspirador de pó, totalmente automático e funciona com um radar. Mas ele é ineficiente pelo fato de passar várias vezes no mesmo lugar e se perder no ambiente caso mude de lugar algum objeto do ambiente.

A Inteligência Artificial (IA) constitui-se em um conjunto de técnicas de programação para resolver determinados tipos de problemas em informática. Ela procura imitar, através dos programas que comandam estas máquinas, as formas de resolução de problemas do mesmo modo que o homem o faz [TAF96].

Segundo [LUG92], um robô cego executará uma seqüência de ações que responderá à mudanças em seu ambiente ou ser capaz de detectar e corrigir erros no seu plano para poder ser considerado inteligente. Muitas vezes, um robô formulará um plano baseado em informações incompletas e corrigirá sua conduta de como ele deverá executar no plano. Um

robô pode não ter sensores adequados para localizar todos os obstáculos na direção do caminho projetado. Dessa maneira um robô se moverá através da sala baseado no que ele "percebeu" e corrigirá seu caminho quando outros obstáculos forem detectados.

O planejamento, segundo [LUG92] é um aspecto importante do esforço de projetos de robôs para executar suas tarefas com um grau de flexibilidade e responsabilidade no mundo externo. O planejamento é um problema difícil por um número de razões, não menos de que é o tamanho do espaço de possibilidades de seqüências de movimento. Por exemplo, um robô que se move para trás, para frente, para os lados direito e esquerdo, considerando os muitos diferentes modos que o robô poderá possivelmente se movimentar através da sala. Assumindo também que existam obstáculos na sala e que o robô deverá escolher um caminho para se mover ao redor dela de alguma forma eficiente. Escrever um programa que possa ser inteligente para descobrir o melhor caminho de acordo com estas circunstâncias sem sobrecarregar em relação ao enorme número de possibilidades, requer técnicas sofisticadas para representar o espaço conhecido e o controle da pesquisa [LUG92].

Devido a ausência de soluções ou soluções insatisfatórias para esse tipo de tarefa/problema, este trabalho tem como objetivo definir uma nova proposta de solução para o caminhamento de rôbos, através do reconhecimento do ambiente, baseado nas observações feitas por [LUG92] e em vários outros autores. A proposta de definição de um simulador de aspiradro de pó autônomo para ambientes com barreiras tem como objetivo principal, além de assegurar o cumprimento da diretiva básica que é o de aspirar o pó de todo um ambiente, o objetivo de determinar as ações do robô, utilizando para isto o algoritmo de busca em largura. Esta técnica da IA permite determinar a escolha de "caminhos ótimos" ou de "menor custo" em situações de *deadlock* ou a garantia de passar uma única vez em cada local, sem repeti-lo, tornando a execução da tarefa muito mais inteligente e eficiente.

A solução proposta para esta tarefa será a utilização de um agente inteligente composto de duas camadas: uma reativa e outra ativa. A primeira camada, ativa, é composta de quatro sensores colocados no aspirador de pó e que são responsáveis por obter os dados do ambiente a ser aspirado e com o objetivo de evitar possíveis acidentes, exatamente por ser o ambiente dinâmico e possuir obstáculos. A segunda camada, reativa, repassa para a camada

ativa, as informações obtidas pelos sensores. A camada ativa, encapsula um sistema inteligente rotineiro que define a direção a ser tomada pelo aspirador e, quando necessário, utiliza a técnica do algoritmo de busca para a escolha do melhor percurso para uma nova posição que ainda não foi aspirada.

1.3 OBJETIVOS

1.3.1 OBJETIVOS PRINCIPAIS:

O objetivo principal deste trabalho é estudar os problemas relacionados com caminhamento de rôbos autônomos em ambientes dinâmicos com barreiras, e apresentar uma solução à nível de protótipo para o devido problema.

1.3.2 OBJETIVOS SECUNDÁRIOS

a) indicar novas possíveis tarefas envolvendo:

- técnicas de busca;
- agentes reagentes;

b) indicar outras técnicas e possíveis melhorias para o trabalho proposto.

1.4 IMPORTÂNCIA DO TRABALHO

A importância do presente trabalho está intimamente ligada à utilização de técnicas da IA, mais especificadamente agentes e algoritmo de busca, para a especificação do caminhamento de rôbos autônomos e inteligentes, na realização de tarefas corriqueiras realizadas pelo homem, em ambientes diversos com barreiras.

1.5 METODOLOGIA

Visando atingir os objetivos citados anteriormente foi realizada uma pesquisa teórica, a especificação e a implementação de um protótipo e, finalmente a elaboração de ambientes para testes.

A pesquisa teórica abrangeu uma revisão da literatura especializada, tanto nacional quanto estrangeira. Foram pesquisados livros, revistas, artigos científicos, dissertações de mestrado e teses de doutorado.

Na especificação e na implementação do protótipo foram observadas as recomendações levantadas ao longo da pesquisa teórica.

A elaboração de ambientes para testes foram feitos com o intuito de apresentar justamente ambientes que poderiam gerar algum problema ou dificuldade para que o simulador possa mostrar sua performance e eficiência na simulação dos mesmos.

1.6 ORGANIZAÇÃO DO TEXTO

O capítulo1 apresenta um histórico juntamente com uma introdução, os objetivos, a importância, e a estrutura do trabalho.

O capítulo 2 apresenta uma revisão bibliográfica sobre algumas das técnicas da IA, que serão utilizadas no trabalho. Portanto apresenta definições, conceitos, aplicações e métodos sobre:

- agentes inteligentes;
- agentes reativos;
- reconhecimento de ambientes;
- métodos de busca.

O capítulo 3 apresenta a solução proposta para o problema de um robô aspirador em ambientes diversos com barreiras. Apresenta em detalhes:

- como o robô aspirador realiza o reconhecimento do ambiente através de sensores;
- como o robô aspirador realiza o caminhamento, como escolhe a direção a ser tomada e como sabe se já passou em um determinado lugar do ambiente;
- sobre situações *deadlock*;
- a técnica da IA, algoritmo de busca em largura, para encontrar o menor percurso, em um ambiente, para tanto, apresenta definições e situações de uso da referida técnica.

O capítulo 4 apresenta a especificação e implementação do simulador do robô aspirador de pó, o protótipo.

O capítulo 5 apresenta as conclusões e as recomendações.

Nos anexos são apresentados os fontes do simulador juntamente com o acompanhamento passo a passo de um ambiente, para simulação.

2 REVISÃO BIBLIOGRÁFICA

Para melhor compreensão sobre os assuntos e técnicas da IA, abordadas no presente trabalho, são apresentados nos tópicos seguintes as definições, conceitos, aplicações e modelos das técnicas utilizadas.

2.1 INTELIGÊNCIA COMPUTACIONAL

Inteligência Computacional (IC) é o estudo de projetos de agentes inteligentes. Um agente é algo que age em um ambiente. Agentes inclui vermes, cachorros, aeroplanos, humanos, organizações e sociedades. Um agente inteligente é um sistema que age inteligentemente: o que ele faz é apropriado para suas circunstâncias e suas metas, e é flexível a mudanças de ambientes e mudanças de metas. Ele aprende pela experiência e faz escolhas apropriadas aplicando percepção de limitações e computação finita [POO98].

O objetivo científico central da IC é a compreensão de princípios que tornam a conduta da inteligência possível em sistemas naturais ou artificiais. A principal hipótese é que o raciocínio é computável possibilitando especificar métodos para projetos de artefatos inteligentes [POO98].

Segundo [POO98], IA é o nome que determina o campo que se define CI, mas o termo “Inteligência Artificial” é um objeto de grande confusão. [POO98] questiona se a IA é realmente inteligente e define “Inteligência Sintética” como o nome mais apropriado. A idéia científica central está em compreender sistemas naturais e artificiais. A confusão sobre o nome do campo, em parte, é atribuído pela confusão do propósito do campo com sua metodologia. O propósito é a compreenção de como o procedimento da inteligência é

possível. A metodologia é o projeto, construção e o experimento com sistemas computacionais que executam tarefas comumente vistas como inteligentes. A construção desses artefatos é uma atividade essencial uma vez que IC é, sobre tudo, uma ciência empírica; mas que não deve ser confundida com o propósito científico.

2.2 AGENTES

Ação racional significa ação de uma meta realizada com êxito partindo de um ponto. Um agente é justamente alguma coisa que percebe e age, porque o modo de agir racionalmente é a razão lógica da conclusão de uma determinada ação sobre uma meta realizada e portanto a ação da conclusão [RUS95]. A junção da percepção, raciocínio e ação constituem um agente [POO98].

Segundo [JEN98], agentes inteligentes é um novo paradigma de desenvolvimento na aplicações de softwares e é referenciado como uma nova revolução em software que esta começando a ser utilizado em uma grande variedade de aplicações. Agentes são um foco de interesse intenso em muitos sub-campos da CI e IA.

Nos tópicos seguintes é apresentado de modo bem resumido os tipos de agentes em termos de essência teórica, hipóteses/objetivos, motivações, funções, exemplos de protótipos e benefícios potênciais.

2.2.1 DEFINIÇÃO DE AGENTES

Um agente pode ser definido como uma entidade real ou virtual que emerge num ambiente onde pode agir intencionalmente, hábil a perceber e representar o ambiente, hábil a comunicar-se com os outros agentes e possuir autonomia [HUB95].

Segundo [POO98], um agente é algo que age em um ambiente. Agentes incluem vermes, cachorros, aeroplanos, humanos, organizações e sociedades. Um agente inteligente que um sistema que age inteligentemente: o que ele faz é apropriado para suas circunstâncias e seus objetivos, é flexível à mudanças de ambientes e mudanças de objetivos, aprende pela experiência e realiza tarefas apropriadas através de escolhas dadas por limitações percentuals e computação finita.

Segundo [JEN98], um agente inteligente é um sistema de computador que é capaz de ações autônomas flexíveis em ordem para encontrar seus objetivos designados. Por flexível, quer-se dizer que o sistema pode ser:

- **responsivo** : agentes deverão perceber seu ambiente (que pode ser o mundo físico, uma coleção de agentes, a Internet, etc.) e responder em tempo às mudanças que ocorrem nele;
- **proativo**: agentes deverão responder não somente em resposta ao ambiente mas, também serem capazes de exibir oportunamente conduta de direção-objetivo e fazer a iniciativa quando necessário;
- **social**: agentes deverão ser capazes de interagir, quando acharem apropriado, com outros agentes artificiais e humanos para resolver seus próprios problemas e ajudar outros com suas atividades.

Segundo [JEN98] existem algumas dimensões na classificação de softwares agentes, que são:

- **agentes colaborativos**: enfatiza autonomia e cooperação com outros agentes. Eles podem aprender, mas este aspecto não é a maior ênfase da sua operação. Para coordenar suas atividades, eles podem ter que negociar a order de prioridades para alcançar consentimento aceitos mutuamente. Agentes colaborativos tendem a ser estáticos, grandes.

- **agentes de interface**: enfatiza autonomia e aprendizagem. Suporta e fornece assistência proativa, tipicamente como um usuário aprende a usar uma aplicação particular tal como uma planilha eletrônica ou um sistema operacional. Os agentes usuários observam e monitoram as ações feitas pelo usuário da interface, aprende novos atalhos e sugere o melhor modo de fazer a tarefa.

- **agentes móveis**: são softwares capazes de processar viajando ao longo da àrea da rede de trabalho, tal como a WWW, interagindo com *hosts* externos, executando tarefas do interesse do seu proprietário e retornado para *'home'* tendo executado os deveres delimitado à ele. São autônomos e cooperativos, embora diferentemente dos agentes colaborativos. Por exemplo, eles podem cooperar com um agente fazendo a locação de objetos internos e métodos conhecidos para outros agentes.

- **agentes de informação/Internet**: executam o papel de administrador, manipulador e intercalação da informação para muitas fontes distribuídas.

- **agentes reativos**: representam uma categoria especial de agentes que não possui natureza interna, modelos simbólicos de seus ambientes; em vez disso ele responde de maneira resposta-estímulo para representar o estado do ambiente em que ele está embutido.

- **agentes híbridos**: combinam duas ou mais filosofias de agentes em um único agente.

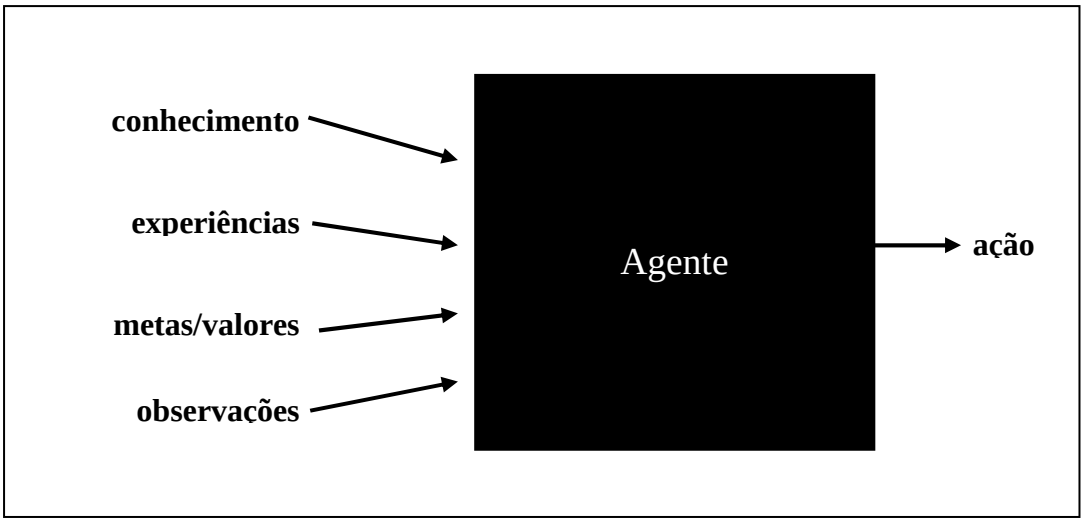
- **agentes heterogêneos**: refere-se a um integrado conjunto de dois ou mais agentes. Um agente heterogêneo também pode conter agentes híbridos.

Um agente pode ser, por exemplo, uma junção de uma máquina computacional com atuadores físicos e sensores, chamado robô. Pode ser a junção de um computador que dá conselhos - um *expert system* (sistema perito) - como um humano que fornece a informação percentual e que carregará para fora a tarefa. Um agente pode ser um programa que age em ambientes puramentes computacionais - um *infobot* [POO98].

A Figura-1 mostra as entradas e saídas que um agente possui, em qualquer tempo [POO98]:

- conhecimento prévio sobre o mundo;
- experiências passadas que aprendeu;
- metas que ele deverá tentar concluir com êxito ou valores sobre o que é importante; e
- observações sobre o ambiente corrente e ele próprio.

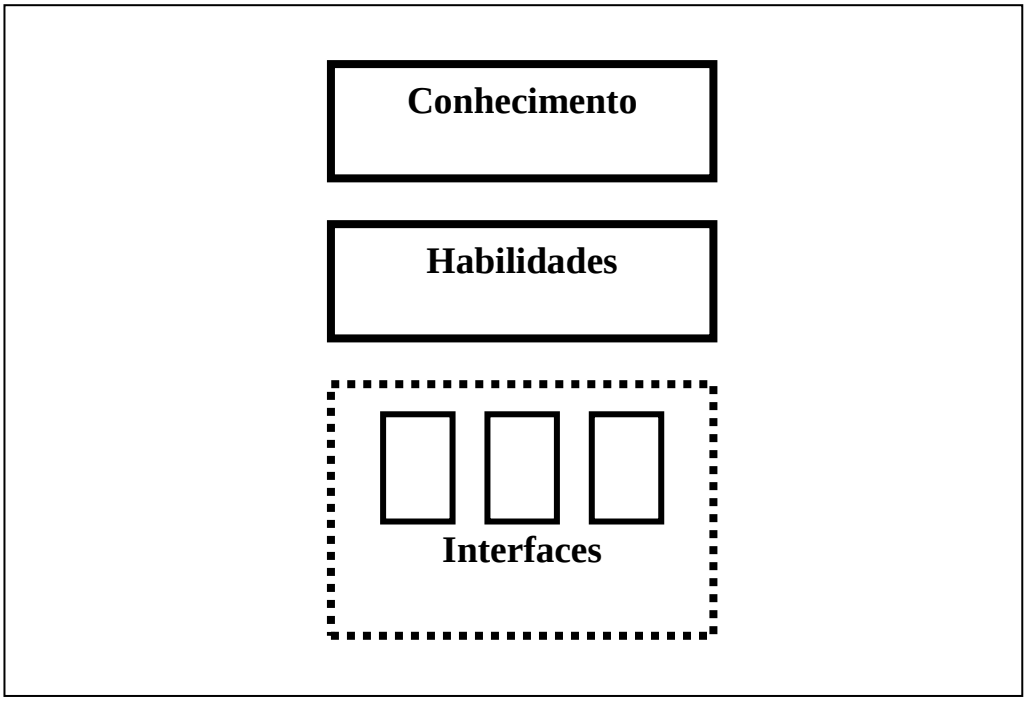
FIGURA-1. ENTRADAS E SAÍDAS DE UM AGENTE



2.2.2 PROJETO DE MODELO DE UM AGENTE

O modelo apresentado por [JEN98], para o desenvolvimento de projetos de agentes, é composto de três blocos, onde estes são tratados separadamente, mas interconectados, Figura-2.

FIGURA-2. PROJETO DE MODELO DE UM AGENTE



- **Habilidades**: descreve o que o agente pode fazer, o que o usuário pode esperar do agente. Usando a aproximação descrita no modelo, o usuário deverá ser capaz de selecionar o "melhor" conjunto de habilidades para especificar a aplicação, basedo no custo, na performance e em outros aspectos. "Inteligência" é outro modo de descrever o atributo. Um agente deverá ter limites de habilidades. Dada uma suficiente instrução ou aprendizagem, um agente deverá ser muito mais inteligente.

- **Conhecimento**: o usuário deverá também ser capaz de selecionar a "melhor" representação do conhecimento. Isto é muitas vezes descrito como "regras". Um conjunto de regras pode ser mais complexo e deverá ter condições inferidas.

Em geral, o conhecimento deverá ser através de informações sobre as preferências do usuário.

- **Interfaces**: é aquilo que o usuário verá na tela. A interface de usuário classifica-se em: interface de usuário e interface de aplicação.

2.2.3 APLICAÇÕES

O domínio de aplicação em que soluções de agentes está existindo é aplicado ou pesquisado em administração de fluxo de trabalho, administração de redes de telecomunicações, controle de tráfego aéreo, processos de serviços de reengenharia, mineração de dados, informações administrativas, comércio eletrônico, educação, assistentes digitais pessoais (PDAs), filtro para e-mail, livrarias digitais, comando e controle e bancos de dados inteligentes [JEN98].

Agentes inteligentes tem sido utilizados nas áreas [POO98]:

- **Industrial**: controle de processo, manufatura , controle de tráfego áereo;
- **Comercial**: informações administrativas, comércio eletrônico, administração de processos de negócios;
- **Medicina**: monitoração de paciente, cuidar da saúde;
- **Entretenimento**: jogos, cinema e teatro iterativo;

2.2.4 AGENTES REATIVOS

Como já dito anteriormente, agentes reativos representam uma categoria especial de agentes que não possui natureza/inteligência interna, modelos simbólicos de seus ambientes; em vez disso ele responde de maneira resposta-estímulo para representar o estado do ambiente em que ele está embutido [JEN98].

Um agente reativo, segundo [POO98], cujas ações realizadas através de simples funções a partir de suas entradas. Por exemplo, uma alarme de seguranças contra ladrões pode detectar movimentos em uma determinada área. Um termostato é outro exemplo de agente reativo. Enquanto sistemas reativos são muito usados, eles tipicamente necessitam de um *hardware* de ligação para perceber o ambiente. Eles não podem adaptar mudanças de objetivos ou mudanças de circunstâncias.

Segundo [JEN98], as três idéias chaves envolvendo agentes reativos são:

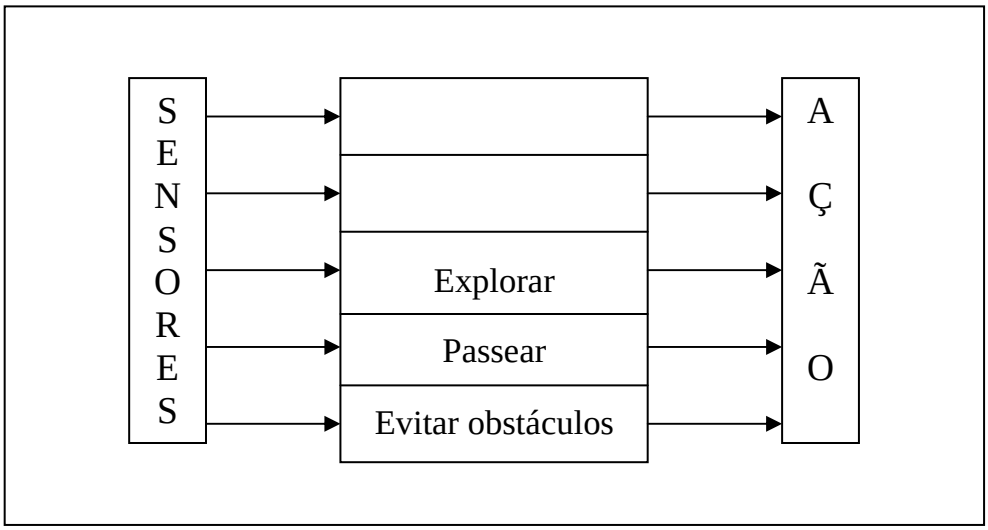
1º - funcionalidade emergente: agentes reativos são relativamente simples e eles interagem com outros agentes do mesmo modo.

2º - decomposição da tarefa: um agente reativo é visto como uma coleção de módulos que operam autonomamente e que são responsáveis pela especificação das tarefas. Comunicação entre os módulos é pequena mas com um alto-nível natural.

3º - sensores: agentes reagentes tendem a operar em representações que são fechadas em sensores de dados rudes em contraste a representações simbólicas de alto-nível que existem em outros tipos de agentes.

A arquitetura representada na Figura-3 pode ser usada para construir vários tipos de rôbos móveis [JEN98]. A mais básica arquitetura é aquela baseada em ação-situação de regras.

FIGURA-3. ARQUITETURA BÁSICA PARA ROBÔS MÓVEIS



A arquitetura acima representada server perfeitamente para ser utilizada por agentes reagentes em reconhecimento de ambientes devido a utilização de sensores na tomada ações, para explorar e caminhar no ambiente que estiver inserido evitando os obstáculos.

Conforme demonstra a figura acima, para que um rôbo móvel possa explorar/passear em um ambiente evitando obstáculos ele deve se utilizar de sensores para obter informações sobre o ambiente em que estiver inserido e a partir dessas informações realizar ações.

2.3 RECONHECIMENTO DE AMBIENTES

Rôbos móveis autônomos são caracterizados por seus veículos que deslocam a si mesmos em um ambiente. Estes rôbos iteragem com seus ambiente onde eles são inseridos através de mecanismo sensoriais, sendo capazes a si próprio operarem em um ambiente desconhecido e que não são necessáriamente estruturados para aquele devido lugar [WRI97].

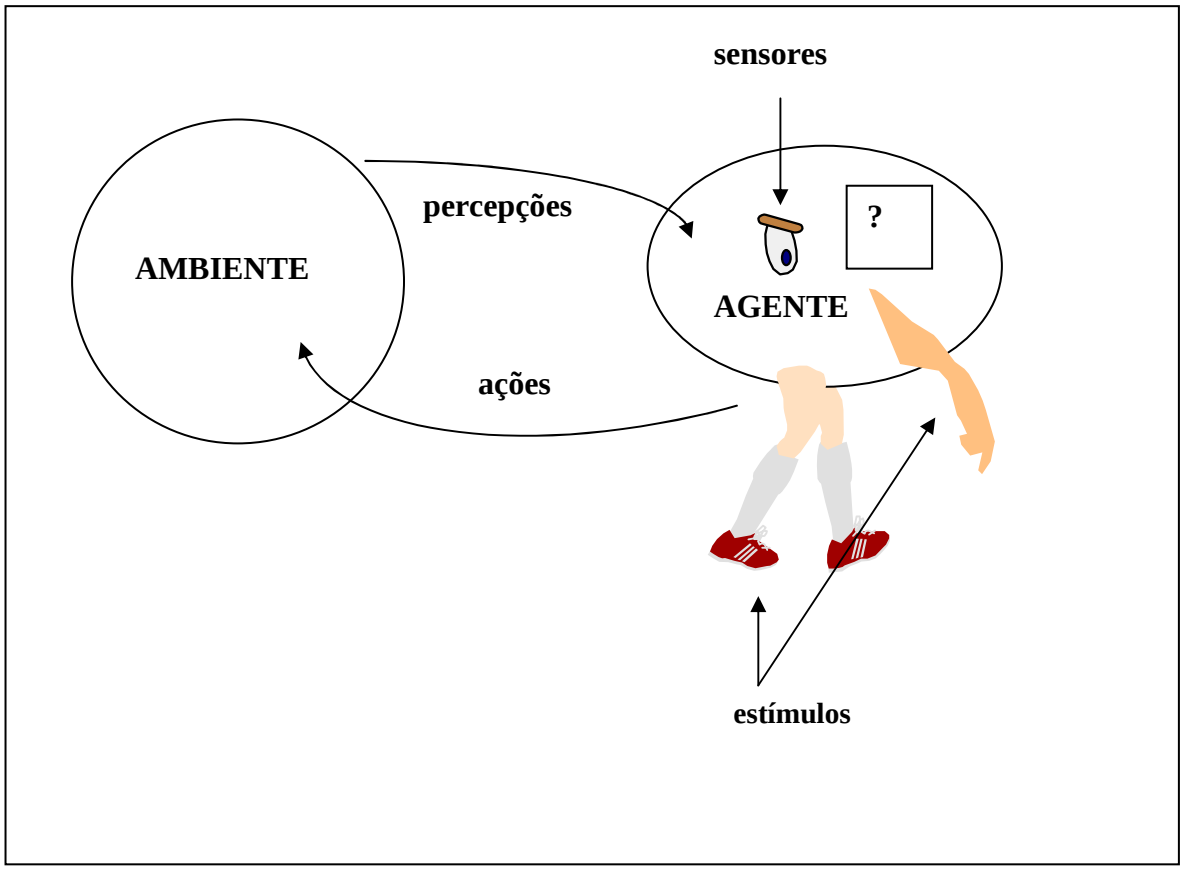
Segundo [WRI97], através de técnicas atuais presentes para tratar com êxito o reconhecimento de ambientes, percebendo mudanças ao redor dele em tempo-real. Reconhecimento de mudanças em tempo real do ambiente requer um alto grau de complexidade, mas novas técnicas tem surgido justamente para sanar este propósito/problema.

Reconhecimento de ambientes é utilizado em máquinas que possuem um estado interno e que simplemente reagem ao estímulo imediato no seu ambiente [NIL98].

A percepção de ambientes de um agente qualquer pode ser visto através de seus sensores e ações sobre o ambiente através de estímulos. Um agente humano tem olhos, ouvidos e outros órgãos de sensores, como mãos, pés, boca e outras partes do corpo para estímulo.

Um agente robótico se utilizará de câmeras e alcance infravermelho de percepção para os sensores e vários motores para estímulo [RUS95]. Na Figura-4, [RUS95], apresenta a diagramação de um agente genérico.

FIGURA-4 - ITERAÇÃO DO AGENTE NO AMBIENTE ATRAVÉS DE SENSORES E PERCEPÇÕES.



Um robô deve se mover baseado no que ele "percebeu" no ambiente e corrigir seu caminho conforme os obstáculos forem detectados [LUG92]. Essa "percepção" sobre o ambiente é obtida através dos sensores laterais (leste, sul, oeste, norte). A idéia de mapeamento para seqüências percebidas de ações é o mesmo que uma lista [RUS95]. O mapeamento portanto é o registro de suas ações e serve para que o agente tome decisões baseados no que ele registrou quando necessário. Resumindo, é sua base de conhecimento.

O reconhecimento do ambiente, no trabalho proposto, é importante para que o robô aspirador de pó saiba por onde percorreu e o que falta percorrer, ou seja, a área que ele aspirou e a área que falta aspirar. Dessa forma o aspirador define o mapa do ambiente. Esse reconhecimento é importante devido ao ambiente ser dinâmico.

2.4 MÉTODOS DE BUSCA

Métodos de busca são técnicas utilizadas para solução de problemas. A solução de um problema pode ser definida como a busca no espaço de estados deste problema. O espaço de estados de um problema é o conjunto de configurações possíveis deste problema [RAB95]. Diferentes métodos produzem diferentes garantias sobre o caminhamento encontrado [POO98].

Muitos problemas de solução de tarefas podem ser transformados no problema de encontrar um caminho no grafo. Busca em grafos produzirá um nível apropriado de abstração dentro dos limites do estudo da solução do problema, independentemente de um domínio particular [POO98]. Informalmente, um grafo consiste de um conjunto de nós juntamente com o conjunto de linhas entre os nós [POO98].

2.4.1 CONCEITOS E DEFINIÇÕES

Um método de busca define os critérios da ordem em que as regras devem ser avaliadas no processo de raciocício do sistema [HEI95]. Nas técnicas de busca, também denominadas por alguns autores de cegas, a ordem em que os nós são expandidos depende somente da informação coletada durante a busca, não sendo afetada pela parte do problema ainda não explorada, nem mesmo pelas metas que estão sendo atingidas [RAB95].

Para facilitar o entendimento é apresentado, a seguir, por [RAB95], o significado de alguns termos básicos:

- **Nós abertos**: são os **nós** que foram gerados e estão aguardando expansão;
- **Nós fechados**: são os **nós** que foram expandidos, isto é, seus sucessores estão disponíveis aos procedimento de busca;
- **ABERTO**: é uma lista que aguarda os **nós abertos**;

- **FECHADO**: é uma lista que aguarda os **nós fechados**.

As técnicas de busca mais comuns são:

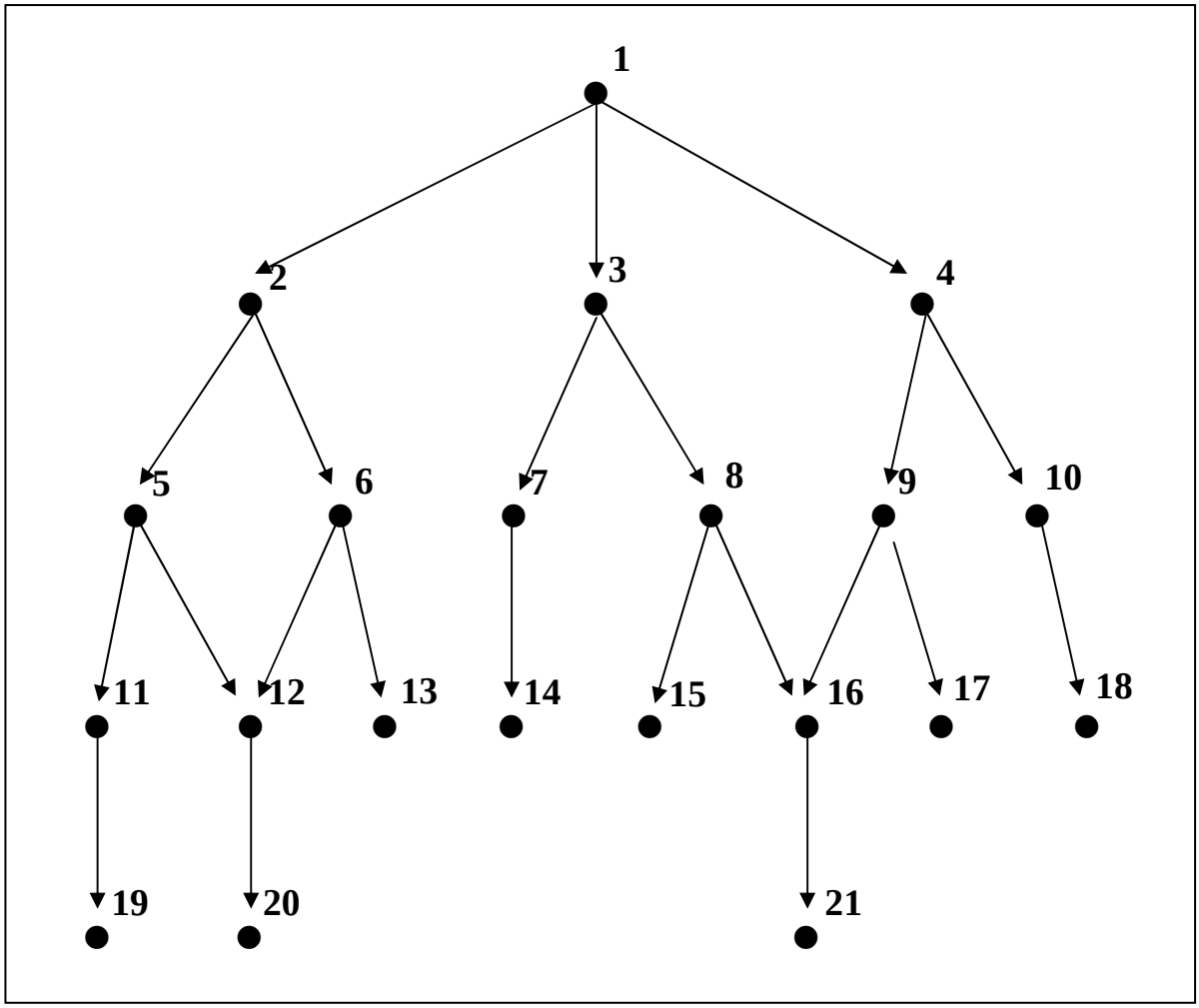
- **Busca em profundidade**: esta técnica, também denominada de primeiro-em-profundidade ou *depth-first*, explora o caminho para o objetivo, dando preferência aos nós que estão mais distantes da raiz da árvore de busca. Esta idéia funciona bem quando as soluções são total e igualmente desejadas ou quando anteriormente foi feita uma varredura, detectando direções incorretas [RAB95].
- **Busca em largura**: é o oposto a *depth-first* e trabalha sob um critério FIFO (*First In first Out*). É também denominada de busca em amplitude, busca em nível e *breadth-first*. É um procedimento em que todos os nós de certo nível da árvore são examinados antes dos do nível abaixo.

2.4.2 BUSCA EM LARGURA

O método de busca em largura foi escolhido para ser utilizado no trabalho devido retornar a primeira solução encontrada como a melhos solução ou a solução de menor custo.

Este método, também chamado de primeiro-em-largura, em contraste com o método de busca em profundidade, consiste em explorar os nós nível-a-nível. Somente quando não existem mais nós para serem explorados num certo nível da árvore o algoritmo considera o nível seguinte, ou seja, os nós descendentes. Neste caso são priorizados os "vizinhos" ou "irmãos" de um nó que tenha sido avaliado em prejuízo dos "descendentes". Na árvore da Figura-5, por exemplo, se o nó 5 tivesse sido avaliado, e fosse dado prioridade aos vizinhos mais à esquerda, o algoritmo passaria a considerar, na ordem, os nós 6, 7, 8, 9 e 10. Somente depois de explorar todos os nós deste nível da árvore é que o nó 11, que é descendente do nó 5, seria considerado [HEI95].

FIGURA-5 - ÁRVORE DE INFERÊNCIAS.



A busca em largura, portanto, segue sempre no sentido de buscar os "irmãos" de um nó e só quando estes estiverem esgotados sem ter-se chegado a um solução, os nós do nível seguinte da árvore são considerados [HEI95]. O algoritmo para a implementação computacional desta técnica é apresentado no próximo tópico.

2.4.2.1 ALGORITMO DE BUSCA EM LARGURA

O algoritmo de busca em largura definido por [HEI95] é apresentado a seguir:

- crie duas filas ABERTO e FECHADOS
- inicialize a fila ABERTOS = [nó início]
- enquanto ABERTOS não estiver vazia faça
- remova o primeiro nó da fila ABERTOS e chame-o de X
- se X é conclusivo termine com sucesso
- senão busque todos os filhos de X
- dispense os filhos de X que já estão em ABERTOS ou FECHADOS
- acrescente os filhos remanescentes de X na fila ABERTOS na ordem que foram buscados
- acrescente X na fila FECHADOS
- fim enquanto

Novamente considerando a situação em que fosse necessária o exame de toda a árvore e adotado o critério "o da esquerda antes", o método de busca em largura avaliaria os nós na seguinte ordem de seqüência: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21 [HEI95].

Aplicando-se este algoritmo sobre a árvore da Figura-5 o comportamento das filhas ABERTOS e FECHADOS seria, para cada iteração do laço enquanto:

1ª iteração: ABERTOS = [1]
FECHADOS = []

2ª iteração: ABERTOS = [2,3,4]
FECHADOS = [1]

3ª iteração: ABERTOS = [3,4,5,6]
FECHADOS = [1,2]

4ª iteração: ABERTOS = [4,5,6,7,8]
FECHADOS = [1,2,3]

5ª iteração: ABERTOS = [5,6,7,8,9,10]
FECHADOS = [1,2,3,4]

6ª iteração: ABERTOS = [6,7,8,9,10,11,12]
FECHADOS = [1,2,3,4,5]

7ª iteração: ABERTOS = [7,8,9,10,11,12,13]
FECHADOS = [1,2,3,4,5,6]

8ª iteração: ABERTOS = [8,9,10,11,12,13,14]
FECHADOS = [1,2,3,4,5,6,7]

9ª iteração: ABERTOS = [9,10,11,12,13,14,15,16]
FECHADOS = [1,2,3,4,5,6,7,8]

10ª iteração: ABERTOS = [10,11,12,13,14,15,16,17]
FECHADOS = [1,2,3,4,5,6,7,8,9]

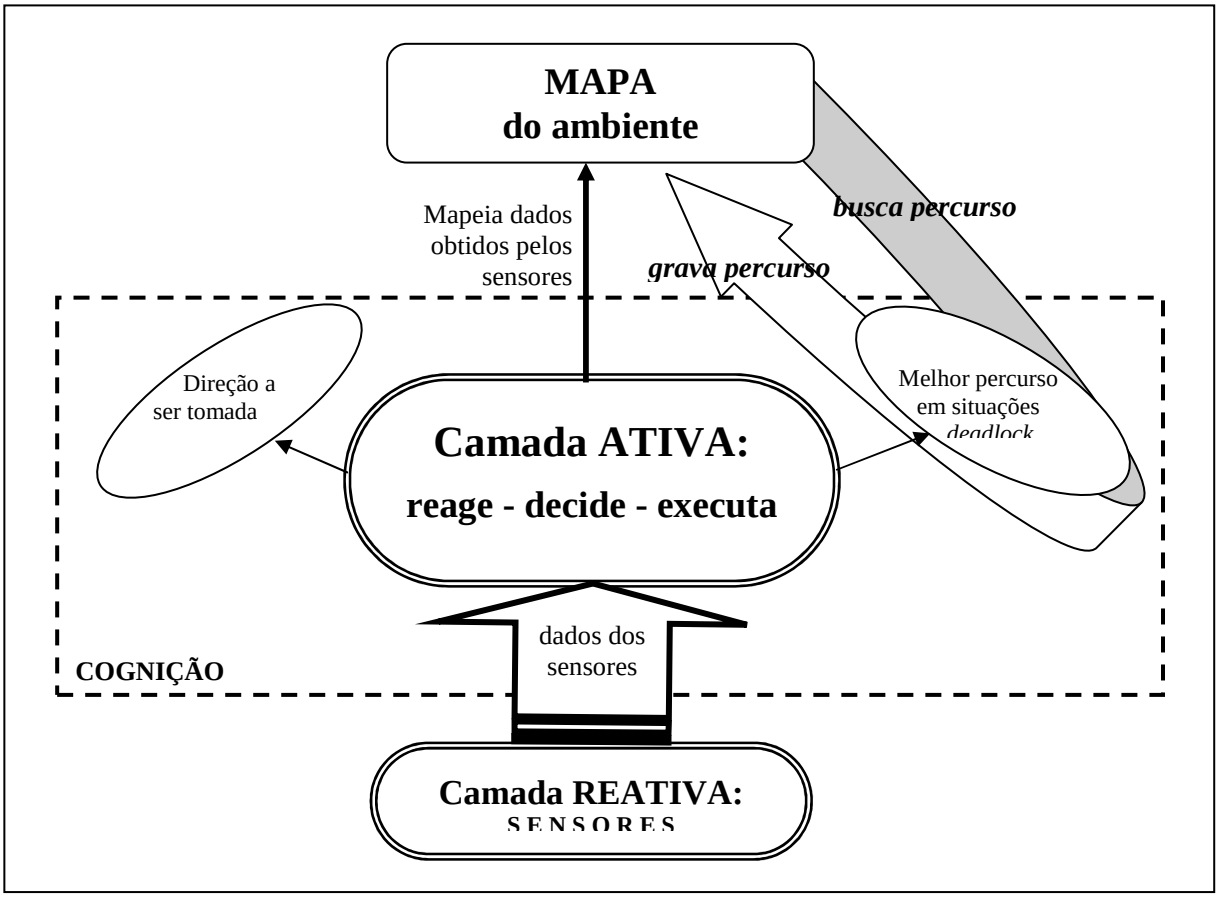
-
-
-

3 A SOLUÇÃO PROPOSTA

3.1 APLICAÇÕES DE AGENTES NO TRABALHO

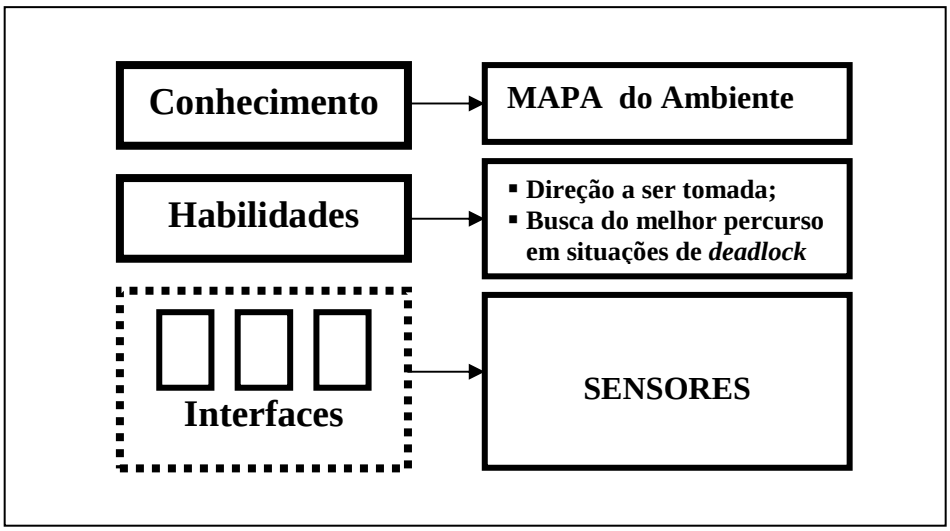
O trabalho propôs definir um rôbo aspirador de pó autônomo. O rôbo será capaz de deslocar-se por todo e qualquer ambiente, em que ele se encontrar inserido, desviando de barreiras e obstáculos. Para que o rôbo possa reconhecer o ambiente, desviar das barreiras e dos obstáculos é utilizado um agente inteligente composto de duas camadas: uma reativa e outra ativa, ver Figura-6.

FIGURA-6 - REPRESENTAÇÃO DOS AGENTES DO RÔBO ASPIRADOR.



A Figura-7 apresenta uma comparação entre o projeto de modelo de um agente apresentado por [JEN98], descrito no item 2.2.2, e o modelo proposto pelo trabalho, Figura-6.

FIGURA-7 - COMPARAÇÃO ENTRE O MODELO PROPOSTO.

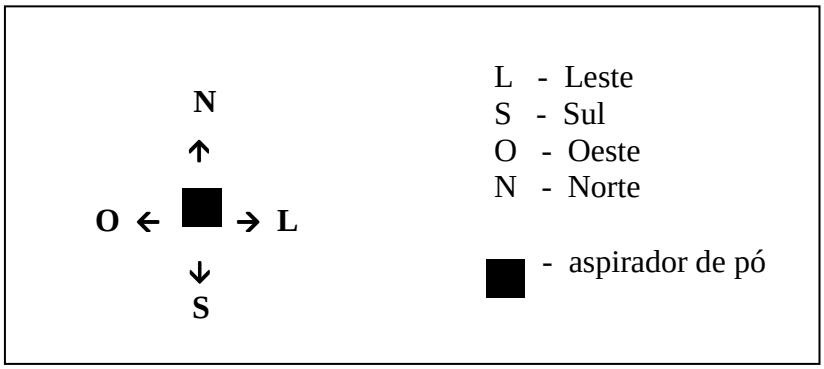


3.1.1 A CAMADA REATIVA

A camada reativa baseia-se no paradigma de solução de problemas gere-e-teste. Ela é responsável por obter os dados dos 4 (quatro) sensores, do rôbo aspirador, de ambiente do agente. Solucionadores de problemas que aderem ao paradigma usam dois módulos básicos. Um módulo, o gerador, enumera as soluções possíveis. O segundo, o testador, avalia cada solução proposta, aceitando ou rejeitando [WIN81].

Os sensores localizados nas laterais do rôbo, chamados de sensores cardinais, tem a função de reconhecer o ambiente e testar as possibilidades de deslocamento do aspirador nas direções: leste, sul, oeste e norte, como mostra a Figura-8. O sistema, a camada ativa, testa as quatro direções, e define se é possível ou impossível se locomover em cada uma delas, ou seja, se existe uma barreira ou não. Portanto a camada reativa tem a função de prever e evitar possivéis acidentes, exatamente por ser o ambiente, adverso e instável.

FIGURA-8 - SENSORES LATERAIS DO RÔBO ASPIRADOR.



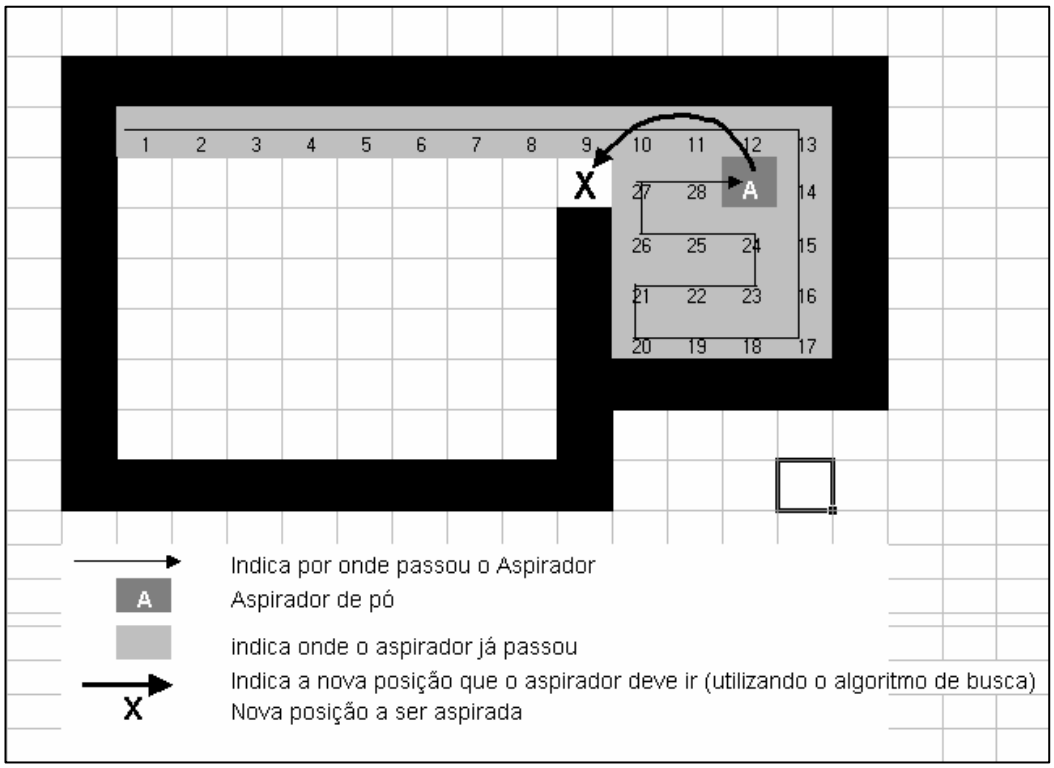
3.1.2 A CAMADA ATIVA

A camada ativa encapsula um sistema inteligente rotineiro e sensitivo, responsável pelo cumprimento de sua diretiva básica: aspirar o pó de todo o ambiente possível. A camada ativa é que define a direção que o aspirador de pó irá tomar.

Cada informação obtida dos sensores, de cada posição em que o aspirador se localizar no ambiente, a camada ativa grava as referidas informação num arquivo/tabela denominado de "mapa do ambiente". Essas informações são importante porque é através dessas informações que é feito o reconhecimento do ambiente e também para que o sistema saiba onde o rôbo aspirador caminhou e onde ainda falta caminhar.

A técnica de algoritmo de busca faz parte da camada ativa e é utilizada por ela para escolher o melhor percurso para uma nova posição que ainda não foi aspirada, em casos de *deadlock*. Um situação de *deadlock* acontece quando o aspirador numa determinada posição/localização, no ambiente, os sensores perceberem que as direções localizadas por eles já foram passadas pelo rôbo aspirador ou encontra-se uma barreira, conforme pode ser visto na Figura-9. O item 3.4 trata em detalhes situações *deadlock* e a utilização do algoritmo de busca.

FIGURA-9 - SITUAÇÃO *DEADLOCK* - UTILIZAÇÃO DO ALGORITMO DE BUSCA.



3.2 RECONHECIMENTO DO AMBIENTE

O reconhecimento do ambiente se realiza através da parte reativa, que são os sensores laterais: leste, sul, oeste, norte; eles informam se uma determinada direção, ou coordenada, é possível (P) ou impossível (I). Quando uma direção ou coordenada é verificada como (P), significa que não existe barreira e que o aspirador de pó pode se locomover até aquela direção; e (I) significa que a direção ou coordenada é impossível, ou seja, que o aspirador não pode se locomover naquela direção. Uma barreira pode ser uma parede, um móvel, ou qualquer outro objeto que possa estar bloqueando o aspirador em uma das quatro direções.

3.2.1 MAPEAMENTO DO AMBIENTE

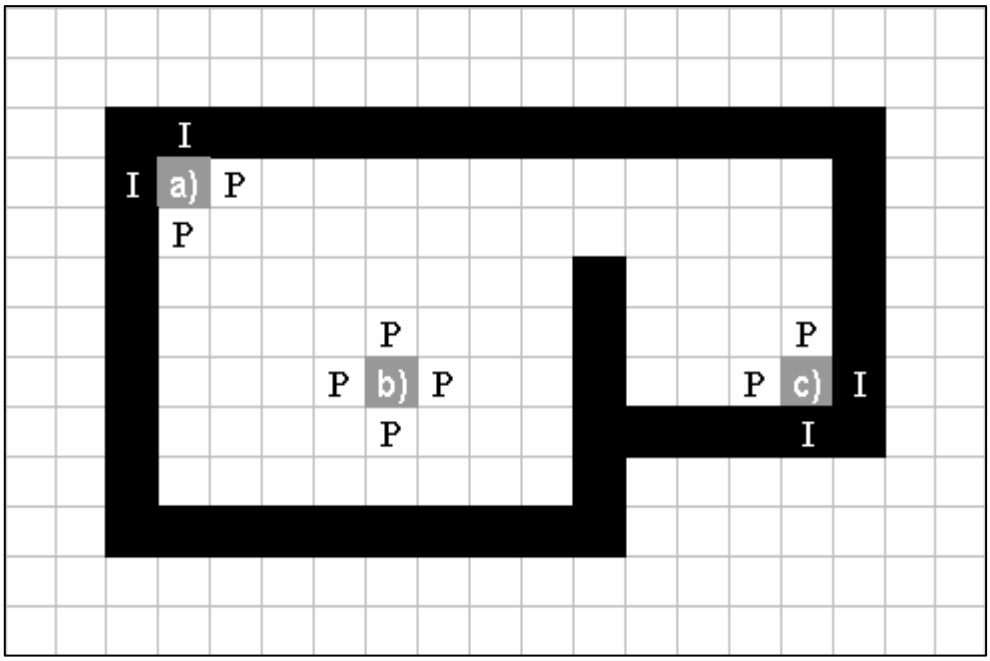
Quando o processo de aspirar pó é iniciado, a posição em que o aspirador se encontrar será tomada como a posição inicial, que é a posição (0,0) em coordenadas cartesianas (X,Y). A partir do momento em que o aspirador de pó se locomover para uma determinada direção, a coordenada (X,Y), que representa a posição atual do aspirador, será atualizar conforme a direção escolhida (veja item 3.3.1 - escolha da direção a ser tomada). Portanto, o reconhecimento do ambiente é sempre obtido a partir da posição em que o aspirador se encontrar. A cada passo ou deslocamento que o aspirador der é feito o reconhecimento do ambiente e essas informações sobre cada posição e suas possíveis direções são armazenadas para que o aspirador de pó possa saber por onde ele caminhou e onde ainda falta caminhar no ambiente.

Partindo desta afirmação, como exemplo, pode-se dizer que o reconhecimento do ambiente da posição atual do aspirador de pó que é (0,0) é verificada através dos sensores laterias como mostra a Figura-10 e a Tabela-1 é:

TABELA-1 - RECONHECIMENTO DO AMBIENTE A PARTIR DA POSIÇÃO ATUAL

			Localização do aspirador na Figura 2		
SENSORES	**(X , Y)**	**(0 , 0)**	**a)**	**b)**	**c)**
L	(X+1, Y)	(1 , 0)	Possível	Possível	Impossível
S	(X , Y-1)	(0 , - 1)	Possível	Possível	Impossível
O	(X-1, Y)	(-1 , 0)	Impossível	Possível	Possível
N	(X , Y+1)	(0 , 1)	Impossível	Possível	Possível

FIGURA-10 - RECONHECIMENTO DO AMBIENTE A PARTIR DA POSIÇÃO ATUAL



3.2.2 ESTRUTURA DO MAPEAMENTO

As informações obtidas através dos sensores sobre as quatro direções (leste, sul, oeste, norte) são armazenadas em um registro que fará parte do mapa do ambiente que o robô aspirador irá reconhecendo, conforme descrito no item anterior. As informações armazenadas são sempre em relação a posição atual do robô aspirador. Portanto, a cada passo ou direção que o aspirador de pó tomar, essa nova posição é tomada como posição atual e os dados dessa nova posição devem ser armazenados em um novo registro.

As informações contidas no registro de ambiente são: a) a coordenada em que o aspirador de pó se encontra (**posição atual** em relação ao ponto de partida); b) as coordenadas das quatro direções (**leste, sul, oeste, norte**); e c) a **situação** dessas quatro direções (**P**-direção possível, **I**-direção impossível, **J**-aspirador já passou nessa direção, **E**-direção escolhida como próximo passo). A Figura-11 mostra o formato do registro que conterá as informações da posição atual do aspirador de pó.

FIGURA-11 - REGISTRO DA POSIÇÃO ATUAL DO ASPIRADOR DE PÓ.

Pos. Atual	Coord. Leste	Sit. Leste	Coord. Sul	Sit. Sul	Coord. Oeste	Sit. Oeste	Coord. Norte	Sit. Norte	Pos. anterior	Pos. próxima
(x,y)	**(x,y)**	**[P,I]**	**(x,y)**	**[P,I]**	**(x,y)**	**[P,I]**	**(x,y)**	**[P,I]**	**(x,y)**	**(x,y)**

A seguir é demonstrado passo a passo a execução do aspirador de pó, utilizando a parte ativa para a escolha da nova posição através dos quatro sensores, sempre na ordem de preferência: 1^o Leste, 2^o Sul, 3^o Oeste, 4^o Norte. Caso nenhuma das direções for possível é utilizado o algoritmo de busca para encontrar uma nova posição a ser aspirada. Conforme aparece na Figura-9, quando o aspirador chegar na posição 29, o aspirador não conseguirá ir para nenhuma das direções porque estas já foram passadas pelo aspirador. Nesses caso é que o algoritmo de busca será utilizado, para que ele encontre o menor caminho a ser percorrido pelo aspirador de pó até uma nova posição a ser aspirada, caso essa nova posição exista, este assunto é abordado no item 2.4. No caso do exemplo da Figura-9 essa nova posição é indicada por um "**X**".

A Figura-12 contém os dados sobre cada posição por onde o aspirador passou. Essas informações representam o mapa do ambiente, e contém informações sobre cada localização aspirada, as localizações que faltam aspirar e as localizações que não são possíveis aspirar. Como pode ser visto na Figura-12, quando o aspirador estiver na posição (11,-1), ele não poderá ir para nehuma das direções porque o aspirador já esteve antes em todas as quatro direções.

FIGURA-12 - REGISTROS DO MAPA DE AMBIENTE.

MAPA DO AMBIENTE: contém os registro de cada posição que o aspirador esteve								
Posição aspira	1 - LESTE		2 - SUL		3 - Oeste		4 - NORTE	
(x,y)	(x,y)	sit.	(x,y)	sit	(x,y)	sit	(x,y)	sit
0,0	1,0	E	0,-1	P	-1,0	I	0,1	I
1,0	2,0	E	1,-1	P	0,0	J	1,1	I
2,0	3,0	E	2,-1	P	1,0	J	2,1	I
3,0	4,0	E	3,-1	P	2,0	J	3,1	I
4,0	5,0	E	4,-1	P	3,0	J	4,1	I
5,0	6,0	E	5,-1	P	4,0	J	5,1	I
6,0	7,0	E	6,-1	P	5,0	J	6,1	I
7,0	8,0	E	7,-1	P	6,0	J	7,1	I
8,0	9,0	E	8,-1	P	7,0	J	8,1	I
9,0	10,0	E	10,-1	P	8,0	J	9,1	I
10,0	11,0	E	11,-1	P	9,0	J	10,1	I
11,0	12,0	E	12,-1	P	10,0	J	11,1	I
12,0	13,0	I	12,-1	E	11,0	J	12,1	I
12,-1	13,-1	I	12,-2	E	11,-1	P	12,0	J
12,-2	13,-2	I	12,-3	E	11,-2	P	12,-1	J
12,-3	13,-2	I	12,-4	E	11,-3	P	12,-2	J
12,-4	13,-4	I	12,-5	I	11,-4	E	12,-3	J
11,-4	12,-4	J	11,-5	I	10,-4	E	11,-3	P
10,-4	11,-4	J	10,-5	I	9,-4	E	10,-3	P
9,-4	10,-4	J	9,-5	I	8,-4	I	9,-3	E
9,-3	10,-3	E	9,-4	J	8,-3	I	9,-2	P
10,-3	11,-3	E	10,-4	J	9,-3	J	10,-2	P
11,-3	12,-3	J	11,-4	J	10,-3	J	11,-2	E
11,-2	12,-2	J	11,-3	J	10,-2	E	11,-1	P
10,-2	11,-2	J	11,-3	J	9,-2	E	10,-1	P
9,-2	10,-2	J	9,-3	J	8,-2	I	9,-1	E
9,-1	10,-1	E	9,-2	J	8,-1	P	9,0	J
10,-1	11,-1	E	10,-2	J	9,-1	J	10,0	J
11,-1	12,-1	J	12,-2	J	10,-1	J	11,0	J

3.3 CAMINHAMENTO DO ROBÔ ASPIRADOR

3.3.1 ESCOLHA DA DIREÇÃO A SER TOMADA

O próximo passo, a ser tomado pelo aspirador de pó, é determinado atráves dos estados obtidos de cada sensor lateral do aspirador de pó. O estado determina se uma determinada direção é possível (P) ou impossível (I).

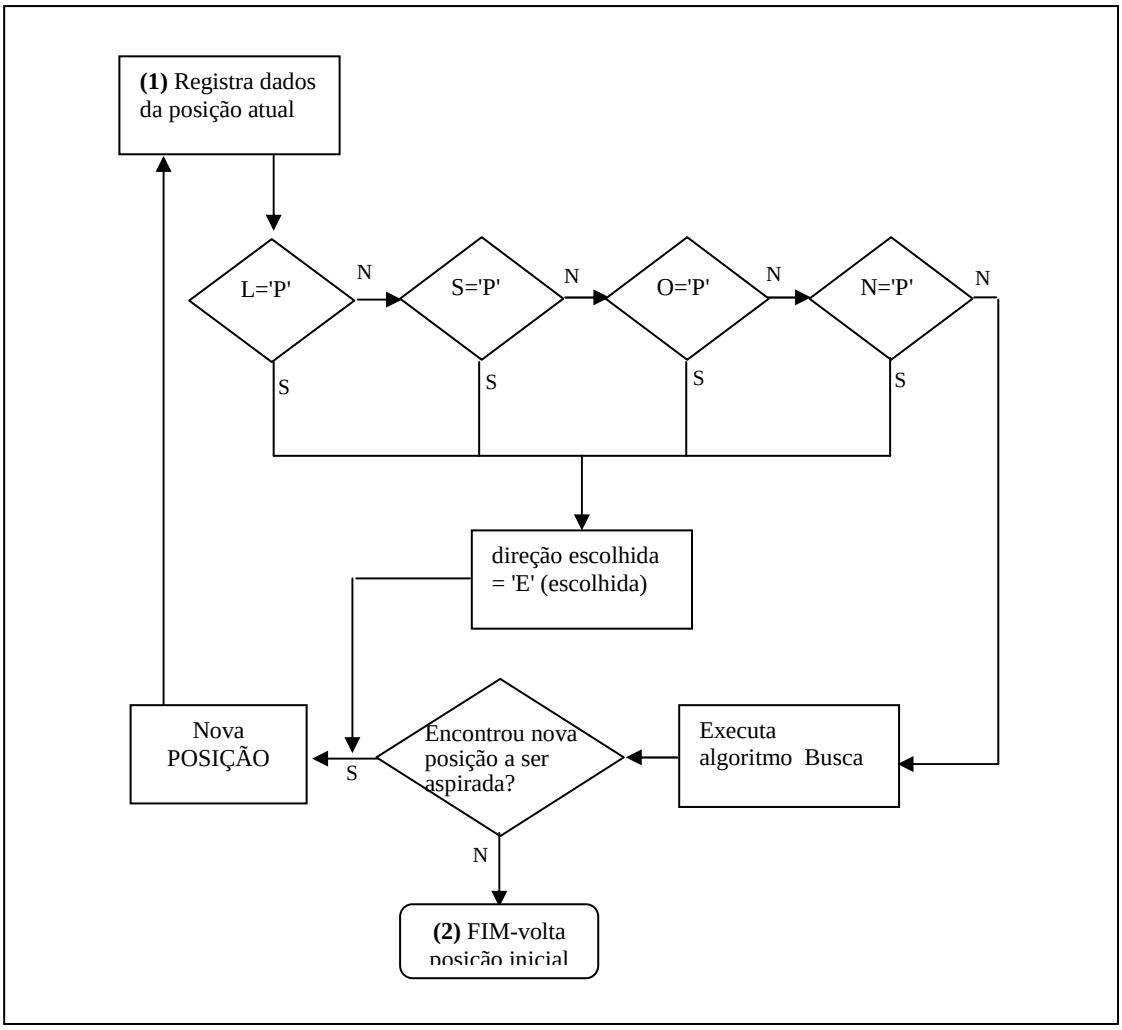
A determinação do próximo passo, é feita através da seguinte ordem: leste, sul, oeste e norte. Primeiro é verificado se é possível ir para a direção leste (L), senão for possível ir para a direção leste então verifica se é possível ir para a direção sul (S), se não for possível ir para a direção sul então ir para a direção oeste (O), se não é possível ir para a direção oeste então ir para a direção (N). Se depois de testar todas as quatro direções e for verificado que não é possível tomar nenhuma das direções, o sistema, do aspirador de pó irá utilizar o algoritimo de busca, para encontrar uma nova posição a ser aspirada, caso haja mais alguma.

A Figura-13 demonstra o fluxo para a escolha da próxima direção a ser tomada pelo robô aspirador. Para maior entendimento são apresentadas duas observações no fluxo:

(1) Registra dados da posição atual: registra a posição atual em que o robô aspirador se encontra e a localização das direções obtidas pelos sensores, juntamente com a situação": (P)-possível, (I)-impossível ou (J)-já foi passada.

(2) Fim - volta posição inicial: para voltar à posição inicial em que o aspirador de pó se encontrava dentro do ambiente ao iniciar o processo é também utilizado o algoritmo de busca.

FIGURA-13 - FLUXO PARA DETERMINAR A ESCOLHA DA NOVA POSIÇÃO A SER TOMADA.



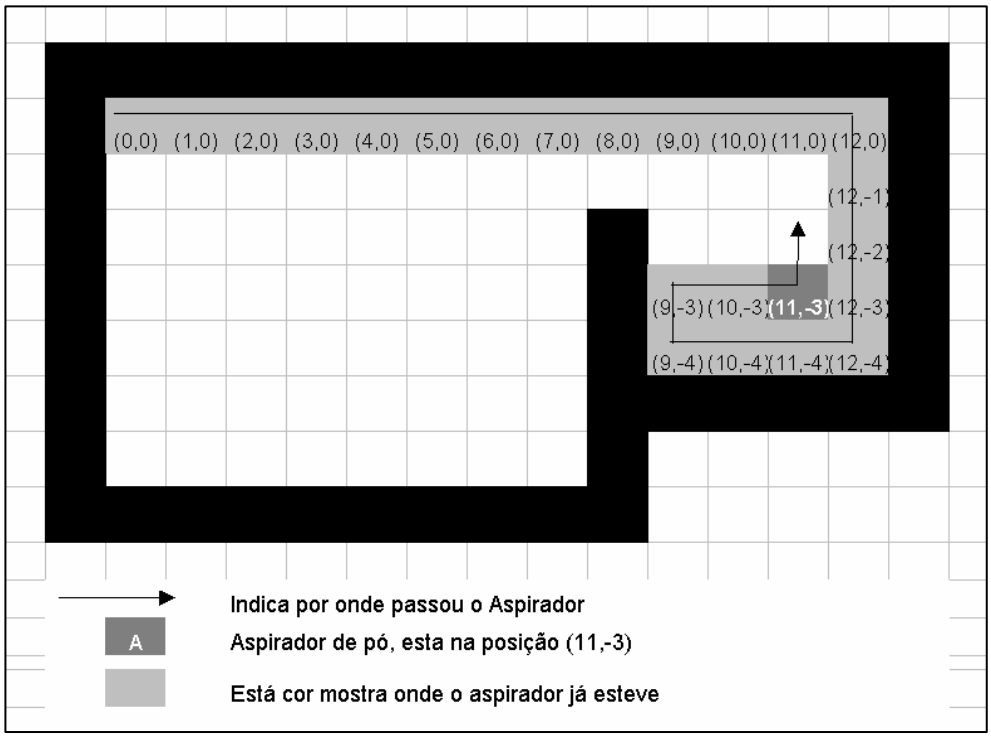
3.3.2 VERIFICANDO SE UMA COORDENADA JÁ FOI ASPIRADA

A verificação de uma coordenada (uma posição do ambiente a ser aspirada) é feita na hora do registro dos dados da posição atual em que se encontra o robô aspirador. É verificado se as coordenadas de cada direção já foram tomadas alguma vez como sendo a posição atual do aspirado de pó, ou seja, se o aspirador de pó já passou por aquela posição. Caso o aspirador de pó não tenha passado pela direção analizada, a informação sobre a situação dessa direção é setada com 'J', que indica que aquela coordenada já foi aspirada, caso contrário a

situação é setada como 'P', indicando que é possível ir nesta direção e indicando também que está em aberto, ou seja, o aspirador ainda não passou naquela posição.

A verificação de um coordenada, para saber se já foi passada alguma vez pelo robô aspirador, é feita comparando a coordenada com os registros da tabela de "mapa do ambiente" sobre cada posição atual que o aspirador de pó já passou. Caso a verificação de coordenada não fosse feita, o robô aspirador, como mostra a Figura-14, iria tomar a direção Leste, pois é possível ir para esta direção, e com isso o aspirador ficaria em um *loop*. Mas como é feita a verificação da coordenada para ver se o aspirador já passou por ela, então o aspirador irá para a direção Norte. Ele toma a direção Norte porque primeiro ele testa em Leste, que falha porque ele já passou; em segundo ele testa em Sul, que também falha pois já passou; em terceiro ele testa em Oeste , que novamente falha porque também já esteve nesta posição e por último testa em Norte, que é possivel e ainda não esteve nesta posição, ou seja, a posição Norte não se encontra registrada nenhuma vez como posição atual na tabela de "mapa do ambiente". Se Norte também falhasse, seria utilizado o algoritmo de busca para encontrar um nova posição a ser aspirada, caso haja uma nova posição.

FIGURA-14 - ESCOLHENDO A PRÓXIMA DIREÇÃO.



3.3.3 ESPECIFICAÇÃO DO CAMINHAMENTO DO ROBÔ ASPIRADOR

No Quadro-1, a seguir, é apresentado a especificação, em portugol, do caminhamento do robô aspirador.

QUADRO-1. DEFINIÇÃO DO CAMINHAMENTO DO ROBÔ ASPIRADOR

```
programa ASPIRA;

início
        tipo RegistroTabMapa   = registro
                                 inteiro:  POSIÇÃO_ATUAL_X,   POSIÇÃO_ATUAL_Y,
                                           LESTE_X,           LESTE_Y,
                                           SUL_X,             SUL_Y,
                                           OESTE_X,           OESTE_Y,
                                           NORTE_X,           NORTE_Y;
                                   caracter: LESTE_SIT,
                                             SUL_SIT,
                                             OESTE_SIT,
                                             NORTE_SIT;
                                    fim registro;

        tipo CoordenadaXY        = registro
                                       inteiro: X;   { coluna }
                                       inteiro: Y;   { linha  }
                                   fim registro;

        tipo     no_Pai          = ponteiro Dados_Pai;
        tipo     Dados_Pai       = registro
                                       CoordenadaXY     : POSIÇÃOXY;
                                       no_Pai           : PAI;
                                       no_Pai           : PRÓXIMO;
                                   fim registro;

        tipo     no_Aberto       = ponteiro Dados_Aberto;
        tipo     Dados_Aberto    = registro
                                       CoordenadaXY     : POSIÇÃOXY;
                                       no_Pai           : PAI;
                                       no_Aberto        : PRÓXIMO;
                                   fim registro;

        tipo     no_Fechado      = ponteiro Dados_Fechado;
        tipo     Dados_Fechado= registro
                                       CoordenadaXY     : POSIÇÃOXY;
                                       no_Fechado       : PRÓXIMO;
                                   fim registro;

        tipo     no_Caminho_Busca        = ponteiro  Dados_Caminho_Busca;
        tipo     Dados_Caminho_Busca= registro
                                           no_Caminho_Busca   : ANTERIOR;
                                           CoordenadaXY   : POSIÇÃOXY;
                                       fim registro;
```

QUADRO-1. DEFINIÇÃO DO CAMINHAMENTO DO ROBÔ ASPIRADOR (CONTINUAÇÃO)

```
tabela          : TabMAPA  RegistroTabMapa;
CoordenadaXY    :  POSIÇÃO_ATUAL,  antPOSIÇÃO;
lógico          : LSON, TERMINOU;

POSIÇÃO_ATUAL·X ← 0;
POSIÇÃO_ATUAL·Y ← 0;

LSON            ← verdadeiro;
TERMINOU        ← falso;

limpa(TabMAPA);


enquanto TERMINOU=falso faça

        antPOSIÇÃO ← POSIÇÃO_ATUAL;

        COLOCA_POSIÇÃO_MAPA(POSIÇÃO_ATUAL);

        abre(TabMAPA);
        posiciona último registro(TabMAPA);

        se tabMAPA·LESTE_SIT = 'P'  então        { LESTE é  'Possível' }
                POSIÇÃO_ATUAL·X  ← TabMAPA ·LESTE_X;
                POSIÇÃO_ATUAL·Y  ← TabMAPA ·LESTE_Y;
                TabMAPA·LESTE_SIT ← 'E';      { escolhida }
        senão
                se tabMAPA·SUL_SIT = 'P'  então
                        POSIÇÃO_ATUAL·X  ← TabMAPA·SUL_X;
                        POSIÇÃO_ATUAL·Y  ← TabMAPA·SUL_Y;
                        TabMAPA·SUL_SIT ← 'E';
                senão
                        se tabMAPA·OESTE_SIT = 'P'  então
                                POSIÇÃO_ATUAL·X  ← TabMAPA·OESTE_X;
                                POSIÇÃO_ATUAL·Y  ← TabMAPA·OESTE_Y;
                                TabMAPA·OESTE_SIT ← 'E';
                        senão
                                se tabMAPA·NORTE_SIT = 'P'  então
                            POSIÇÃO_ATUAL·X  ← TabMAPA·NORTE_X;
                                    POSIÇÃO_ATUAL·Y  ← TabMAPA·NORTE_Y;
                                    TabMAPA·NORTE_SIT ← 'E';
                                senão
                                    LSON ← falso;
                                fim se;
                        fim se;
                fim se;
        fim se;

        fecha(TabMAPA);
```

QUADRO-1. DEFINIÇÃO DO CAMINHAMENTO DO ROBÔ ASPIRADOR (CONTINUAÇÃO)

```
                se  LSON = falso  então

                        se BUSCA_NOVA_POSIÇÃO(antPOSIÇÃO) = falso então
                                TERMINOU ← true;
                        senão
                                POSIÇÃO_ATUAL ← antPOSIÇÃO;
                        fim se;
                        LSON ← verdadeiro;

                senão
                        "Move Aspirador   POSIÇÃO_ATUAL"
                fim se;

        fim enquanto;

        BUSCA_NOVA_POSIÇÃO(POSIÇÃO_ATUAL,verdadeiro);
fim.




{-------------------------------------------------------------------------------------------------------------------------}
procedimento POSIÇÃO_ESTA_TabMAPA( X, Y );
        inteiro: X, Y;

início

        lógico : ACHOU;

        ACHOU ← falso;

        abre(TabMAPA);
        posiciona primeiro registro(TabMAPA);

        enquanto  ACHOU=falso  e  não  fim(TabMAPA)  faça

                se  TabMAPA.POSIÇÃO_ATUAL_X  = X  e
                    TabMAPA.POSIÇÃO_ATUAL_Y  = Y então
                        ACHOU ← verdadeiro;
                senão
                        próximo registro(TabMAPA);
                fim se;

        fim enquanto;

        POSIÇÃO_ESTA_TabMAPA ←  ACHOU;

fim;
```

QUADRO-1. DEFINIÇÃO DO CAMINHAMENTO DO ROBÔ ASPIRADOR (CONTINUAÇÃO)

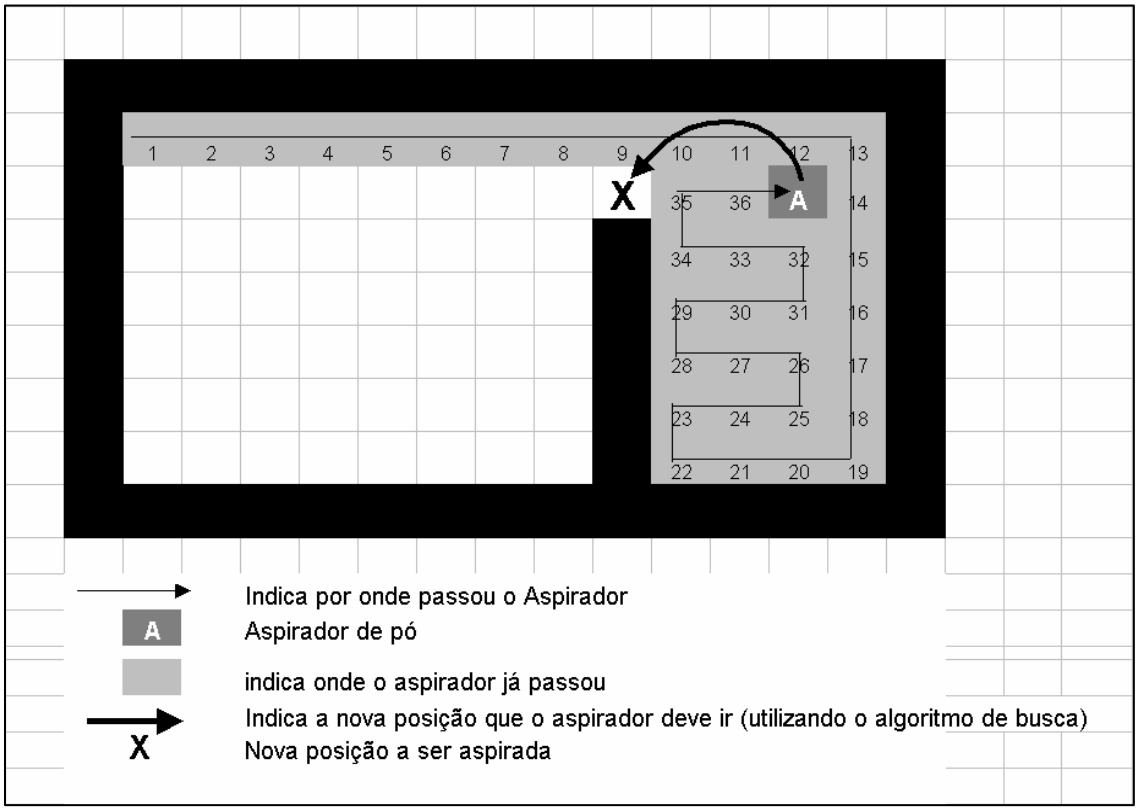
```
{-------------------------------------------------------------------------------------------------------------------------}
procedimento COLOCA_POSIÇÃO_MAPA(POSIÇÃO);
        CoordenadaXY : POSIÇÃO;

início

        inteiro : X, Y;

        abre(TabMAPA);
        novo registro(TabMAPA);

        X ← POSIÇÃO·X;
        Y ← POSIÇÃO·Y;
        tabMAPA·POSIÇÃO_ATUAL_X ← X;
        tabMAPA·POSIÇÃO_ATUAL_Y ← Y;
        tabMAPA·LESTE_X ← X+1;
        tabMAPA·LESTE_Y ← Y;
        tabMAPA·SUL_X   ← X;
        tabMAPA·SUL_Y   ← Y+1;
        tabMAPA·OESTE_X ← X-1;
        tabMAPA·OESTE_Y ← Y;
        tabMAPA·NORTE_X ← X;
        tabMAPA·NORTE_Y ← Y-1;

        { Sensor LESTE detectou barreira? }
        se SENSOR_LESTE = verdadeiro então  tabMAPA·LESTE_SIT ← 'I';          { Impossível }
        senão                               tabMAPA·LESTE_SIT ← 'P'; fim se; { Possível    }

        se SENSOR_SUL = verdadeiro então    tabMAPA·SUL_SIT ← 'I';
        senão                               tabMAPA·SUL_SIT ← 'P';      fim se;

        se SENSOR_OESTE = verdadeiro então  tabMAPA·OESTE_SIT ← 'I';
        senão                               tabMAPA·OESTE_SIT ← 'P';    fim se;

        se SENSOR_NORTE = verdadeiro então  tabMAPA·NORTE_SIT ← 'I';
        senão                               tabMAPA·NORTE_SIT ← 'P';    fim se;

        se POSIÇÃO_ESTA_TabMAPA(X+1, Y) = verdadeiro então  { leste }
                tabMAPA·LESTE_SIT ← 'J';
        fim se;

        se POSIÇÃO_ESTA_TabMAPA(X, Y+1) = verdadeiro então  { sul }
                tabMAPA·SUL_SIT ← 'J';
        fim se;

        se POSIÇÃO_ESTA_TabMAPA(X+1, Y) = verdadeiro então  { oeste }
                tabMAPA·OESTE_SIT ← 'J';
        fim se;

        se POSIÇÃO_ESTA_TabMAPA(X+1, Y) = verdadeiro então  { norte }
                tabMAPA·NORTE_SIT ← 'J';
        fim se;

        fecha(TabMAPA);

fim;
```

3.4 UTILIZAÇÃO DO MÉTODO DE BUSCA NO TRABALHO

O método de busca é utilizado quando o robô aspirador de pó se encontrar em situações *deadlock*. Um situação *deadlock*, como pode ser observado na Figura-15, acontece quando o robô aspirador numa determinada posição/localização dentro do ambiente não puder tomar uma nova direção (leste, sul, oeste, norte), a qual seria a próxima posição a ser aspirada, devido a existência de uma barreira ou de se tratar de uma posição já passada/aspirada pelo robô aspirador. Portanto, o método de busca determinará a escolha do "caminho ótimo", ou seja, mais curto, a ser tomado tornando a execução da tarefa muito mais inteligente e eficiente.

Como no exemplo da Figura-15 pode-se observar que o robô aspirador se encontra numa situação *deedlock*, onde todas as direções, em relação a localização do aspirador no ambiente, já foram aspiradas pelo mesmo.

FIGURA-15 - SITUAÇÃO *DEADLOCK* - UTILIZAÇÃO DO ALGORITMO DE BUSCA.



Fazendo uma comparação, da situação *deedlock* em que se encontra o robô aspirador na Figura-15, entre o método burro, que é realizado através do caminho que já se conhece obtido através do "mapa do ambiente" que o robô aspirador vai registrando, e o método de busca para encontrar a nova posição a ser aspirada, pode-se observar a eficiência do último simplesmente pela quantidade de passos que o robô fará. Essa comparação pode ser observada na Tabela-2.

TABELA-2 - COMPARACÃO ENTRE O MÉTODO DE BUSCA E O MÉTODO BURRO.

	Posição de cada passo no ambiente																												**Total Passos**
Método Busca	36	35	**X**																										**3**
Método Burro	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	**X**	**29**

3.4.1 ESPECIFICAÇÃO DO MÉTODO DE BUSCA UTILIZADO NO ROBÔ ASPIRADOR

O Quadro-2 apresenta a especificação do algoritmo de busca utilizado para encontrar o caminho para uma nova posição a ser aspirada ou para encontrar o caminho para a posição inicial (0,0) em que o robô aspirador de pó se encontrava no ambiente ao inIciar a execução da tarefa.

Para especificar que o algoritmo de busca deve encontrar o caminho para uma nova posição ou se deve ir para a posição inicial é o parâmetro "TERMINOU". Se "TERMINOU" for setado como "falso", significa que ainda pode ter alguma posição a ser aspirada e "verdadeiro" significa que o robô aspirador já passou por todo ambiente e que o algoritmo de busca deve encontrar o caminho, mais curto, até a posição inicial.

QUADRO-2. ESPECIFICAÇÃO DO ALGORITMO DE BUSCA.

```
{-------------------------------------------------------------------------------------------------------}
função BUSCA_NOVA_POSIÇÃO(referência POSIÇÃO_ATUAL, TERMINOU) : lógico ;
        CoordenadaXY:POSIÇÃO_ATUAL;
        lógico    :TERMINOU;   { TERMINOU = verdadeiro voltar para a posição inicial }
                               { TERMINOU = falso encontrar nova posição a ser aspirada}

início
        no_Aberto       : INI_ABERTOS        { Início Lista Abertos   }
        no_Fechados     : INI_FECHADOS       { Início Lista Fechados }
        no_Pais         : INI_PAIS           { Início Lista Pais          }
        no_Aberto       : FIM_ABERTOS        { Fim Lista Abertos     }
        no_Fechados     : FIM_FECHADOS       { Fim Lista Fechados    }
        no_Pais         : FIM_PAIS           { Fim Lista Pais     }
        lógico          : TEM_ABERTO,  ACHOU,  ENCONTROU;
        CoordenadaXY    : POSIÇÃO;
        no_Aberto       : ABERTO;
        no_Pai          : END_PAI;

        INI_ABERTOS     ← vazio;
        INI_FECHADOS    ← vazio;
        INI_PAIS        ← vazio;
        FIM_ABERTOS     ← vazio;
        FIM_FECHADOS    ← vazio;
        FIM_PAIS        ← vazio;
        ACHOU           ← falso;
        TEM_ABERTO      ← verdadeiro;

        COLOCA_PAI( INI_PAIS, FIM_PAIS, POSIÇÃO_ATUAL, vazio );
        COLOCA_ABERTO(INI_ABERTOS, FIM_ABERTOS, POSIÇÃO_ATUAL, vazio);

        ABERTO ← INI_ABERTOS;

        repita

                ENCONTROU ← falso;
                abre(TabMAPA);
                posiciona primeiro registro(TabMAPA);
                enquanto   #fim(TabMAPA)  e  ENCONTROU = falso  faça
                     se tabMAPA·POSIÇÃO_ATUAL_X = ABERTO·POSIÇÃOXY·X e
                          tabMAPA·POSIÇÃO_ATUAL_Y = ABERTO·POSIÇÃOXY·Y então
                         ENCONTROU = verdadeiro;
                      else
                           próximo registro(TabMAPA);
                     fim se;
                fim enquanto;

                END_PAI ← FIM_PAIS;
                se   TabMAPA·LESTE_SIT   # 'I'   então
                    POSIÇÃO·X ← TabMAPA·LESTE_X;
                    POSIÇÃO·Y ← TabMAPA·LESTE_Y;
                    se  VERIFICA_FECHADO(INI_FECHADOS, POSIÇÃO)=falso   então
                         se  VERIFICA_ABERTO(INI_FECHADOS, POSIÇÃO)=falso   então
                             COLOCA_ABERTO(INI_ABERTO, FIM_ABERTO, POSIÇÃO, END_PAI
                        fim se;
                    fim se;
                fim se;
```

QUADRO-2. ESPECIFICAÇÃO DO ALGORITMO DE BUSCA (CONTINUAÇÃO).

```
END_PAI ← FIM_PAIS;

se  TabMAPA·LESTE_SIT  # 'I'  então
    POSIÇÃO·X ← TabMAPA·LESTE_X;
    POSIÇÃO·Y ← TabMAPA·LESTE_Y;

    se VERIFICA_FECHADO(INI_FECHADOS, POSIÇÃO)=falso  então
        se VERIFICA_ABERTO(INI_FECHADOS, POSIÇÃO)=falso  então
            COLOCA_ABERTO(INI_ABERTO, FIM_ABERTO, POSIÇÃO, END_PAI
        fim se;
    fim se;

fim se;


se  TabMAPA·SUL_SIT  # 'I'  então
    POSIÇÃO·X ← TabMAPA·SUL_X;
    POSIÇÃO·Y ← TabMAPA·SUL_Y;

    se VERIFICA_FECHADO(INI_FECHADOS, POSIÇÃO)=falso  então
        se VERIFICA_ABERTO(INI_FECHADOS, POSIÇÃO)=falso  então
            COLOCA_ABERTO(INI_ABERTO, FIM_ABERTO, POSIÇÃO, END_PAI
        fim se;
    fim se;

fim se;


se  TabMAPA·OESTE_SIT  # 'I'  então
    POSIÇÃO·X ← TabMAPA·OESTE_X;
    POSIÇÃO·Y ← TabMAPA·OESTE_Y;

    se VERIFICA_FECHADO(INI_FECHADOS, POSIÇÃO)=falso  então
        se VERIFICA_ABERTO(INI_FECHADOS, POSIÇÃO)=falso  então
            COLOCA_ABERTO(INI_ABERTO, FIM_ABERTO, POSIÇÃO, END_PAI
        fim se;
    fim se;

fim se;


se  TabMAPA·NORTE_SIT  # 'I'  então
    POSIÇÃO·X ← TabMAPA·NORTE_X;
    POSIÇÃO·Y ← TabMAPA·NORTE_Y;
    se VERIFICA_FECHADO(INI_FECHADOS, POSIÇÃO)=falso  então
        se VERIFICA_ABERTO(INI_FECHADOS, POSIÇÃO)=falso  então
            COLOCA_ABERTO(INI_ABERTO, FIM_ABERTO, POSIÇÃO, END_PAI
        fim se;

    fim se;

fim se;

fecha(TabMAPA);
```

QUADRO-2. ESPECIFICAÇÃO DO ALGORITMO DE BUSCA (CONTINUAÇÃO).

```
                { Verifica se o primeiro nó na lista de nós abertos é o nó desejado }

                COLOCA_FECHADO( INI_FECHADOS, FIM_FECHADOS,
                                                ABERTO·POSIÇÃOXY );
                DELETA_ABERTO( INI_ABERTOS, FIM_ABERTOS );

                ABERTO ← INI_ABERTOS;

                se ABERTO # vazio então
                   COLOCA_PAI( INI_PAIS, FIM_PAIS, ABERTO·POSIÇÃOXY, ABERTO·PAI );
                fim se;

                se TERMINOU = falso então       { buscar a nova posição a ser aspirada }

                        se POSIÇÃO_ESTA_TabMAPA( ABERTO·POSIÇÃOXY·X,
                                                ABERTO·POSIÇÃOXY·Y )=falso então
                                POSIÇÃO_ATUAL ←ABERTO·POSIÇÃOXY;
                                ACHOU ← verdadeiro;
                        fim se;

                senão

                        se ABERTO·POSIÇÃOXY·X=0 e ABERTO·POSIÇÃOXY·Y=0 então
                                POSICAO_ATUAL := ABERTO·POSIÇÃOXY;
                                ACHOU ← verdadeiro;
                        fim se;

                fim se;


        até ACHOU=verdadeiro ou ABERTO=vazio;



        se ACHOU = falso então

                BUSCA_NOVA_POSIÇÃO ← falso; { terminou sem encontrar nova posição }
        senão
                RECUPERA_CAMINHO_ESCOLHIDO (FIM_PAIS );

                "liberar memória das listas ABERTOS, FECHADOS, PAIS";

                BUSCA_NOVA_POSIÇÃO ← verdadeiro;        { achou nova posição }
        fim se;


fim;
```

QUADRO-2. ESPECIFICAÇÃO DO ALGORITMO DE BUSCA (CONTINUAÇÃO).

```
{-------------------------------------------------------------------------------------------------------------------------}
procedimento COLOCA_PAI( referência INICIO_PAIS, referência FIM_PAIS,   POSIÇÃO,   PAI);
        no_Pai:  INICIO_PAI,  FIM_PAI,   PAI;
CoordenadaXY:  POSIÇÃO;
início
        no_Pai : PAIS;
        aloca memória(PAIS);
        PAIS·POSIÇÃOXY        ← POSIÇÃO;
        PAIS·PAI              ← PAI;
        PAIS·PRÓXIMO          ← vazio;
        se  INICIO_PAIS = vazio então
                INICIO_PAIS ← PAIS;
        senão
                FIM_PAIS·PRÓXIMO ← PAIS;
        fim se;
        FIM_PAIS ← PAIS;
fim;

{-------------------------------------------------------------------------------------------------------------------------}
procedimento COLOCA_ABERTO( referência INICIO_ABERTOS,
                                        referência FIM_ABERTOS,  POSIÇÃO,   PAI);
        no_Aberto:      INICIO_ABERTO,  FIM_ABERTO;
        CoordenadaXY:  POSIÇÃO;
        no_Pai:  PAI;
início
        no_Aberto : ABERTO;
        aloca memória(ABERTO);
        ABERTO·POSIÇÃOXY  ← POSIÇÃO;
        ABERTO·PAI        ← PAI;
        ABERTO·PRÓXIMO    ← vazio;
        se  INICIO_ABERTOS = vazio então
                INICIO_ABERTOS ← ABERTO;
        senão
                FIM_ABERTOS·PRÓXIMO ← ABERTO;
        fim se;
        FIM_ABERTOS ← ABERTO;
fim;

{-------------------------------------------------------------------------------------------------------------------------}
procedimento COLOCA_FECHADO( referência INICIO_FECHADOS,
                                        referência FIM_FECHADOS,  POSIÇÃO);
        no_Fechado:             INICIO_FECHADOS,  FIM_FECHADOS;
        CoordenadaXY:  POSIÇÃO;
início
        no_Fechado : FECHADO;
        aloca memória(FECHADO);
        FECHADO·POSIÇÃOXY       ← POSIÇÃO;
        FECHADO·PRÓXIMO   ← vazio;
        se  INICIO_FECHADOS = vazio então
                INICIO_FECHADOS ← FECHADO;
        senão
                FIM_FECHADOS·PRÓXIMO ← FECHADO;
        fim se;
        FIM_FECHADOS ← FECHADO;
fim;
```

QUADRO-2. ESPECIFICAÇÃO DO ALGORITMO DE BUSCA (CONTINUAÇÃO).

```
{-------------------------------------------------------------------------------------------------------------------------}
procedimento DELETA_ABERTO( referência INICIO_ABERTO, referência FIM_ABERTO);
        no_Aberto: INICIO_ABERTO,
                   FIM_ABERTO;
início
        no_Aberto : AUX;
        AUX ← INICIO_ABERTOS·PROXIMO;
        se  INICIO_ABERTOS  #  vazio então
                libera memória(INICIO_ABERTOS);
                INICIO_ABERTOS ← AUX;
        fim se;
fim;


{-------------------------------------------------------------------------------------------------------------------------}
função VERIFICA_SE_ABERTO(ABERTOS, POSIÇÃO) : lógico ;
        no_Aberto       : ABERTOS;   { lista de nós abertos }
        CoordenadaXY    : POSIÇÃO;
início
        lógico      : ACHOU;
        ACHOU   ← falso;
        enquanto   ABERTOS # vazio   e   ACHOU= falso   faça
                se        ABERTOS·POSIÇÃO·X = POSIÇÃO·X    e
                          ABERTOS·POSIÇÃO·Y = POSIÇÃO·Y    então
                          ACHOU ← falso;
                ABERTOS ←ABERTOS·PRÓXIMO;
        fim enquanto;
        VERIFICA_SE_ABERTO ← ACHOU;
fim;


{-------------------------------------------------------------------------------------------------------------------------}
função VERIFICA_SE_FECHADO(FECHADOS, POSIÇÃO) : lógico ;
        no_Fechado      : FECHADOS;   { lista de nós fechados }
        CoordenadaXY    : POSIÇÃO;
início
        lógico     : ACHOU;
        ACHOU ← falso;
        enquanto   FECHADOS # vazio   e   ACHOU= falso   faça
                se        FECHADOS·POSIÇÃO·X = POSIÇÃO·X    e
                          FECHADOS·POSIÇÃO·Y = POSIÇÃO·Y    então
                          ACHOU ← falso;
                FECHADOS ←FECHADOS·PRÓXIMO;
        fim enquanto;
        VERIFICA_SE_FECHADO ← ACHOU;
fim;
```

QUADRO-2. ESPECIFICAÇÃO DO ALGORITMO DE BUSCA (CONTINUAÇÃO).

```
{-------------------------------------------------------------------------------------------------------------------}
procedimento RECUPERA_CAMINHO_ESCOLHIDO ( FIM_PAIS );
        no_Pai: FIM_PAIS;

início

        lógico                  : ACHOU;
        CoordenadaXY            : POSIÇÃO;
        no_Pai                  : AUX;
        no_Caminho_Escolhido    : CAMINHO, AUX2, ANT;

        ANT             ← vazio;
        AUX             ← FIM_PAIS;
        ACHOU           ← falso;

        {  Recupera Caminho Pais }
        enquanto   AUX # vazio   faça

                aloca memória(CAMINHO);

                CAMINHO·POSIÇÃOXY ← AUX·POSIÇÃOXY;
                CAMINHO·ANTERIOR  ← ANT;
                ANT ← CAMINHO;
                AUX ← AUX·PAI;

        fim enquanto;


        {  Volta o aspirador pelo caminho escolhido }

        ANT   ← CAMINHO;
        AUX2 ← CAMINHO·ANTERIOR;

        enquanto   AUX2 # vazio   faça

                "Move Aspirador para posição em AUX2";

                CAMINHO·ANTERIOR  ← ANT;
                ANT     ← AUX2;
                AUX2    ← AUX2·ANTERIOR;

        fim enquanto;

fim;
```

4 O PROTÓTIPO

O SimRAPA é o software para o protótipo do simulador de um robô aspirador de pó autônomo para ambientes diversos com barreiras.

O SimRAPA deve ser instalado em computador da família PC 486 ou superior; sistema operacional Windows; ter no mínimo 4MB de memória RAM; monitor SVGA; e ter pelo menos 100 MB livre no disco rígido.

Para instalar o SimRAPA, coloque o disquete de instalação no drive e execute o arquivo SimRAPA.INI que está no drive "A:". O arquivo SimRAPA.INI irá criar o diretório SimRAPA no disco rígido (C:) e copiará todos os arquivos de programa e tabelas para o mesmo. Depois de instalado o SimRAPA, para inicializar o protótipo simulador deve-se ir para o diretório "C:\SimRAPA" e executar o arquivo SimRAPA.EXE para que o programa possa ser carregado. Será então apresentada a tela principal do protótipo, veja Figura-16. Nos tópicos seguintes é apresentado o ambiente de implementação utilizado e as operações oferecidas pelo simulador.

4.1 AMBIENTE DA IMPLEMENTAÇÃO

O ambiente, de desenvolvimento de software, utilizado para a criação do protótipo simulador foi o ambiente *Borland Delphi 2.0*. O mesmo foi escolhido por ser um ambiente gráfico e baseado em eventos, além de possuir um amplo conjunto de recursos que vai desde seu criador de formulários (*Form Designer*) até o suporte transparente a todos os formatos mais comuns de banco de dados.

4.2 OPERAÇÕES DO PROTÓTIPO

O **SimRAPA** é composto das seguintes parteS, mostradas na Figura-16:

- **Grade de Ambiente**: onde é criado/alterado ambientes para realizar simulações do robô aspirador, através das operações de desenho (Barreira, Livre e Apagar); onde é apresentado os ambientes selecionados na caixa de listagem de ambientes e onde é realizada a operação de aspirar pó, a qual é realizada ao pressionar o botão "Aspirar".

- **Cursor**: a execução do processo de aspirar pó, atráves do botão "Aspirar", indica a posição atual do robô; na criação/alteração de ambientes indica a posição a ser desenhada/apagada, dependendo da operação de desenho selecionada (Barreira, Livre e Apagar).

- **Operações de desenho do ambiente**: as operações possíveis na criação de ambientes são:

  - **Barreira**: ativa a colocação de barreiras. Depois de selecionada esta opção, por onde o cursor se movimentar na grade será apresentado um "X" que representará uma barreira na devida posição.

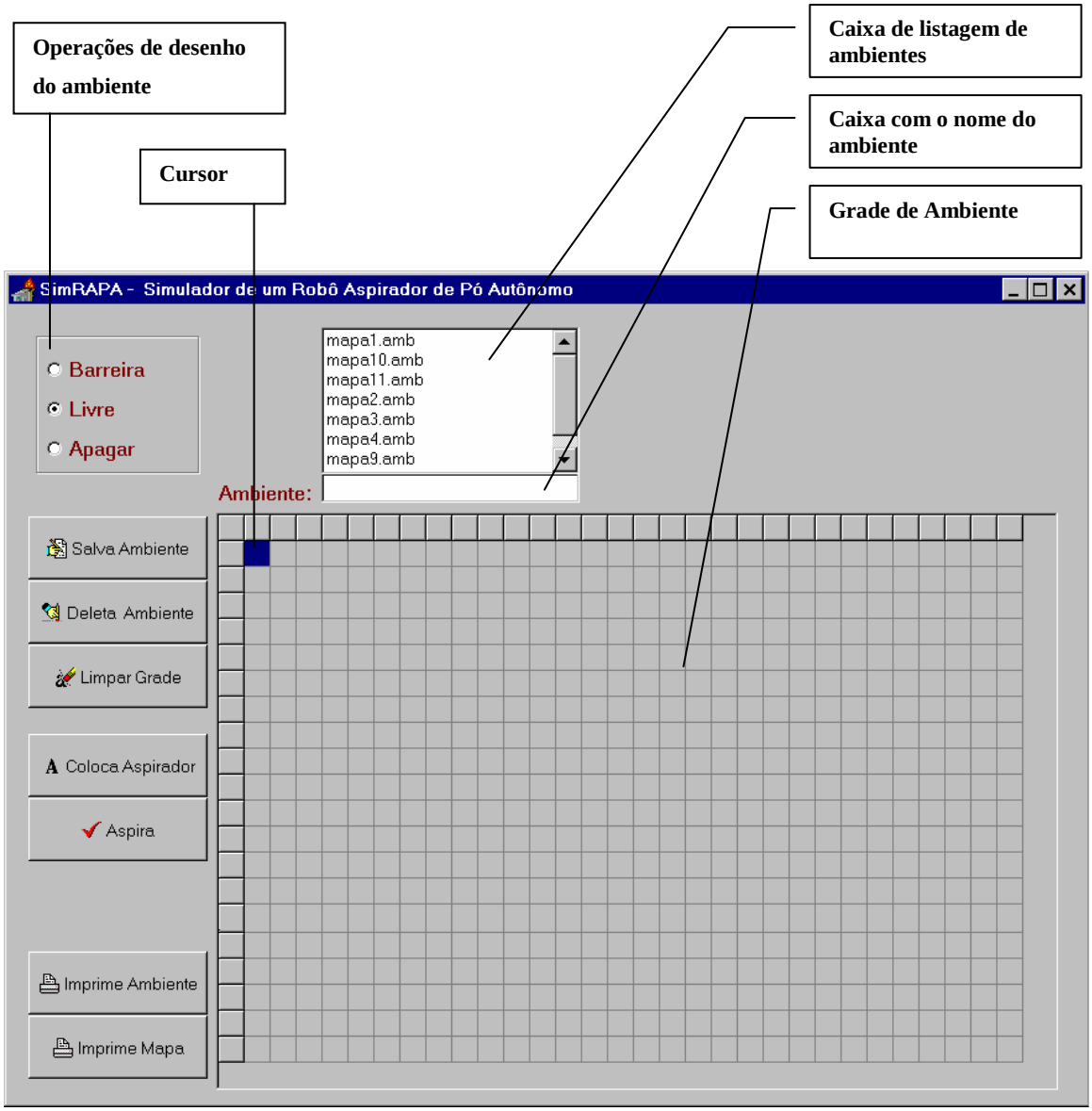
  - **Livre**: depois de selecionada permite movimentar o cursor na grade livremente, ou seja, sem alterar qualquer informação/desenho do ambiente.

  - **Apagar**: depois de selecionada, será apagada as informações do ambiente por onde o cursor se movimentar na grade.

- **Caixa com o nome do ambiente**: onde é informado o nome do arquivo que conterá o ambiente a ser salvo, quando pressionada a tecla "Salva Ambiente". Ambientes existentes podem ser carregados atráves da caixa de listagem, alterados e salvos com o mesmo ou outro nome de arquivo.

- **Caixa de listagem de ambientes**: apresenta os nomes dos arquivos dos ambientes existentes em "C:\SimRAPA". Para carregar um ambiente existente na grade deve-se dar dois "*clicks*" com o botão direito do mouse ou dar um "*click*" no ambiente desejado e depois pressione a tecla "*enter*". Os arquivos de ambientes são arquivos tipo texto e possuem extensão ".amb".

FIGURA-16 - TELA PRINCIPAL DO SIMRAPA.



O SimRAPA permite as seguintes operações, as quais estão apresentadas na forma de botões na tela principal, veja Figura-16:

Este botão salva o ambiente, que se encontra na grade de ambiente, em um arquivo texto, com o nome especificado na caixa "Ambiente:".

Este botão permite eliminar um ambiente existente na caixa de listagem de ambiente. Para eliminar deve-se selecionar o nome do arquivo de ambiente desejado e depois pressionar o botão "Deleta Ambiente".

Este botão, ao ser pressionado, limpa a grade de ambiente para que se possa desenhar/carregar um novo ambiente.

Define a posição inicial do aspirador de pó. Para isso deve-se posicionar primeiro o cursor, da grade de ambiente, na posição desejada como posicão inicial e depois pressionar no botão "Coloca Aspirador". Depois de colocado o aspirador de pó, o botão "Coloca Aspirador" mudará para "Tira Aspirador", para que se possa mudar de posição caso tenha esolhido a posição errada.

Este botão inicializa o processo de simulação do robô aspirador de pó.

Este botão imprime o ambiente carregado na grade. Pode ser impresso antes ou depois da simulação.

Este botão imprime as informações contida na tabela de ambiente da última simulação realizada.

5 CONCLUSÕES

Conforme resultados obtidos pelos experimentos realizados e ambientes criados com o simulador, o mecanismo utilizado para a solução da tarefa de um robô aspirador de pó autônomo mostrou-se eficiente.

O mecanismo proposto para uma implementação de fato ainda é insuficiente, e realmente o é, devido as limitações, descritas no próximo item como: a superfície do chão, o ambiente dinâmico em tempo real, ajuste do passo do robô aspirador, e do ambiente obrigatóriamente ter que ser um ambiente fechado.

A técnica da IA, algoritmo de busca, para encontrar a nova posição a ser aspirada mostrou-se perfeita e ideal para a solução do problema, que é encontrar uma nova posição a ser aspirada no ambiente em situações de *deadlock*, caso haja alguma.

O mapeamento do ambiente é a parte fundamental, por ser a base de conhecimento do robô aspirador e estar presente em todo o processo, pois as decisões realizadas pela camada ativa, que é o cérebro do rôbo aspirador, é totalmente baseada nas informações registradas na tabela/arquivo "mapa de ambiente".

Concluí-se portanto, que o modelo proposto de um agente inteligente, composto de duas camadas: reativa e ativa, para o reconhecimento de ambientes e caminhamento de robôs autônomos utilizado no prótótipo proposto é eficiente e atendeu o que se propôs no presente trabalho. O modelo também pode ser utilizado na automação de outras tarefas como, por exemplo: pintar, lixar ou passar qualquer produto tóxico por todo um ambiente.

5.1 LIMITAÇÕES DO ROBÔ ASPIRADOR PROPOSTO

O protótipo proposto não está preocupado se o robô aspirador está inserido ou não num ambiente fechado. Esse aspecto é muito importante e que deverá ser tratado numa implementação real .

O simulador toma como verdade que o ambiente a ser aspirado é um ambiente fechado, ou seja, o robô aspirador se encontra num ambiente o qual é delimitado por barreiras. Isto implica que numa situação real isto deverá ser tratado para que o robô aspirador não saia aspirando "mundo afora", caso alguém deixe a porta da casa aberta, por exemplo. Isto é um problema mais complexo mas que pode ter várias soluções, algumas caras como a utilização de sensores nas saídas principais do ambiente e outras mais baratas ou simples como delimitar a distância a ser percorrida pelo rôbo aspirador dentro do ambiente a partir da posição atual em que ele se encontra inserido. Nesta última opção essa distância a ser delimitada ao robô aspirador poderia ser váriavel, deixando a critério do usuário.

5.2 TRATAMENTOS PARA UMA IMPLEMENTAÇÃO REAL DO ROBÔ ASPIRADOR

Nume implementação real os seguintes aspectos são muito importantes e deverão ser tratados:

- **superfície do chão**: o simulador ignora aspectos referentes à superfície do chão, que pode ser escorregadia ou ter atrito. Isto ocasionaria o desposicionamento do aspirador, perdendo com isso o conhecimento adquirido do ambiente. O simulador não trata esse tipo de problema, tomando como verdade uma superfície com atrito

suficiente para manter o deslocamento do robô aspirador. Numa aplicação real de um robô aspirador de pó, esse aspecto é de grande importância, tendo que levar em conta o aspecto da superfície e encontrar uma solução ideal para a mesma.

- **ambiente dinâmico em tempo real**: quanto ao ambiente ser dinâmico, o simulador mostra-se suficientemente capaz, mas em relação a mudanças de ambiente em tempo real ocasionaria uma perda da base de conhecimento adquirido, fazendo com que o robô aspirador não percorra todo o ambiente como deveria. O simulador proposto ignora mudanças de objetos no ambiente conhecido, mas não ignorará as do ambiente ainda desconhecido. Mudanças no ambiente podem ocorrer através de um deslocamento de objeto, como uma mesa, um armário, ou até mesmo uma pessoa.

- **passo fixo**: em relação a noção exata de espaço, o simulador trata seu deslocamento dito como "passo fixo", sempre o mesmo tamanho. Espaços de medidas não exatos irão interferir no deslocamento. Numa situação real, este problema deve ser tratado. Por exemplo, um ambiente com 15,35m de largura, e o aspirador com passo de 1m. Os 35cm serão ignorados para efeito de deslocamento, fazendo com que o aspirador se perca na construção do mapa do ambiente e na sua localização no mesmo. Portanto, numa situação real, esse ajuste deverá ser tratado sempre que o robô aspirador encontrar uma barreira e o valor do tamanho do passo, da distância da barreira até o passo anterior, for maior ou menor que o passo padrão. Numa implementação real esse tratamento é simples de ser feito.

5.3 TRABALHOS FUTUROS

Pensando na continuação do trabalho para que se possa apresentar uma solução real nesse sentido, é importante realizar duas etapas de pesquisa, onde cada uma pode ser um novo trabalho de conclusão de curso, devido levantarem novas dificuldades e utilização de novas áreas como por exemplo a automação e eletrônica:

- Elaboração de um robô protótipo real na forma de cubo utilizando quatro sensores laterais de acordo com o modelo proposto para ser testado em ambientes criados e diagramados em quadrados da mesma largura do robô e as barreiras também representadas por cubos, para realizar testes como se propôs o simulador;
- Criar uma implementação de fato em ambientes reais, utilizando o modelo proposto para poder avaliar seu comportamento, juntamente com tratamentos referentes as limitações citadas anteriormente.

- Na utilização de técnicas de IA, poderia se criar um simulador utilizando várias das técnicas para a solução do problema, percorrer todo um ambiente, para que se possa avaliar o desempenho de cada uma delas em termos, por exemplo, de tempo e armazenagem de informação.

- Elaboração de um protótipo real utilizando o modelo proposto para um robô autônomo que percorra todo um ambiente na vertical para a realização de qualquer tarefa, como por exemplo, limpar os vidros de um prédio, pintar ou lixar as paredes de um edifício. Um protótipo real, neste sentido, além de sensores se utilizaria de outros mecanismo para se locomover e identificar o ambiente ao qual ele irá agir, como vidro e parede.

REFERÊNCIAS BIBLIOGRÁFICAS

[AHO95] AHO, Alfred V., SETHI, Ravi, ULLMAN, Jeffrey D. **Compiladores** : princípios, técnicas e ferramentas : Rio de Janeiro : Ed. Afiliada, 1995.

[BEN96] BENDER, Edward A.. **Mathematical methods artificial intelligence** : EUA : IEEE Computer Society Press, 1996.

[BRA97] BRADSHAW, Jeffrey M. **Software agents :** EUA : AAAI Press, 1997.

[ELE98] ELECTROLUX, Group. Electrolux unveils prototype for robot vacuum cleaner 1997. Endereço eletrônico: http://www.welectrolux.co.uk.

[HEI95] HEINZLE, Roberto. **Protótipo de uma ferramenta para criaçãoo de sistemas especialistas baseados em regras de produção.** Florianópolis, 1995. Tese (Mestre em Engenharia) Departamento de Engenharia de Produção e Sistemas, UFSC.

[HUB95] HÜBNER, Jomi Fred. **Migração de Agentes em Sistemas Multi-Agentes Abertos**. Porto Alegre, 1995. Tese (Mestre em Ciências da Computação) dInstituto de Informática, UFRGS.

[JEN98] JENNINGS, Nicholas R., WOOLDRIDGE, Michael J.. Agent Technology : foundations, applications, and markets : Germany : Springer, 1998.

[LUG89] LUGER, George F., STUBBLEFIELD, William A.. **Artificial intelligence** : and the design of expert systems : Califórnia : the Benjamin/Cummings Publishing company, 1989.

[LUG92] LUGER, George F., STUBBLEFIELD, William A.. **Artificial intelligence e design of expert system**: Editora Polytécnica : São Paulo, 1989.

[NIL98] NILSSON, Nils J. **Artificial inteligence** : a new Synthesis : EUA : Morgan Kaufmann Publishers, 1998.

[POO98] POOLE, David, MACKWORTH, Alan, GOEBEL, Randy. **Computational intelligence** : a logical approach : EUA : Oxford University Press, 1998.

[RAB95] RABUSKE, Renato Antônio. **Inteligência artificial** : Florianópolis : Ed. da UFSC, 1995.

[RUS95] RUSSELL, Stuart J., NORVIG, Peter. **Artificial intelligence** : a modern approach : EUA : Prentice-Hall International, 1995.

[TAF96] TAFNER, Malcon A., XEREZ, Marcos de, FILHO, Ilson W. Rodrigues. **Redes neurais artificiais** : introdução e princípios de neurocomputação : Blumenau : Ed. EKO e Ed. da FURB, 1996.

[WIN81] WINSTON, Stuart J.. **Inteligencia artificial** : Ed. Livros Técnicos e Científicos : Rio de Janeiro, 1996.

[WRI97] **Workshop de robótica inteligente (Brasília : Agosto, 1997) :** W.L. Roque and D. A. Barone : Porto Alegre : SBC, 1997.