CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Técnicas de Heurística como Agentes Inteligentes para o projeto de jogos

MÁRCIO DOS SANTOS BERETTA

Marília, 2013

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Técnicas de Heurística como Agentes Inteligentes para o projeto de jogos

Monografia apresentada ao Centro Universitário Eurípides de Marília como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.
Orientador: Prof. MSc. Maurício Duarte

Marília, 2013

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
BACHARELADO EM SISTEMAS DE INFORMAÇÂO

TRABALHO DE CONCLUSÃO DE CURSO – AVALIAÇÃO FINAL

Márcio dos Santos Beretta

Técnicas de Heurística como Agentes Inteligentes para o projeto de jogos.

Banca examinadora da monografia apresentada ao Curso de Bacharelado em Sistemas de Informação do UNIVEM/F.E.E.S.R., para obtenção do Título de Bacharel em Sistemas de Informação.

Nota: 9.0 (nove)

Orientador: Mauricio Duarte

1°. Examinador: Rodolfo Barros Chiaramonte

2°. Examinador: Renata Aparecida de Carvalho Paschoal

Marília, 04 de dezembro de 2013.

“Nossas escolhas não podem ser apenas intuitivas, elas têm que refletir o que a gente é. Lógico que se deve reavaliar decisões e trocar de caminho: ninguém é o mesmo para sempre.” (Pedro Bial).

AGRADECIMENTOS

Agradeço em primeiro lugar a Deus, por presentear-me com uma vida repleta de realizações, pelas oportunidades em minha vida e por me dar forças para superar todos os obstáculos que encontro nos caminhos que escolho.

Agradeço aos meus pais, Ari e Lúcia, pois me deram a vida e me ensinaram a vivê-la com dignidade, iluminaram o meu caminho com afeto e dedicação para que eu trilhasse sem medo e cheio de esperança, se doaram inteiros, sempre me deram força e me ajudaram a vencer.

As minhas irmãs pelo incentivo e pela paciência.

A minha namorada Cinthia, que de forma especial e carinhosa me deu força e coragem, me apoiando nos momentos de dificuldades.

Aos meus amigos que se fizeram presentes quando precisei.

Ao meu orientador Mauricio Duarte, pela ajuda com seu conhecimento, paciência e Incentivo para que eu pudesse concluir este projeto.

Agradeço também a todos os professores que me acompanharam durante a graduação, e ao longo desses quatro anos, partilharam comigo seus conhecimentos.

RESUMO

A heurística estima atingir um determinado objetivo sem que se tenha necessidade de passar por todos os estados possíveis, de uma arvore ou grafo, para encontrar o mesmo. Uma das questões abordadas é a tecnologia de agentes que é estudada por vários profissionais da área de computação, está presente nos dias atuais e no futuro estará constantemente em evidência. O termo "Agente Inteligente" provém da Inteligência Artificial. A Internet é um campo vasto onde os agentes se proliferam e ajudam nas mais diversas tarefas imperceptíveis do ponto de vista do usuário. Neste projeto serão estudadas técnicas de busca heurística e um protótipo de jogo será implementado, neste jogo um agente inteligente será programado para encontrar o melhor caminho para a saída de um labirinto.

Palavras-chave: inteligência artificial, agente inteligente, heurística games.

ABSTRACT

The heuristic is an estimate to achieve a certain goal without having the need to go through all the possible states of a tree or grafo for find the same. One of the issues addressed is the agent, technology that is studied by many professionals in the field of computing, is present nowadays and in the future will be constantly in evidence. The term "Intelligent Agent" comes from Artificial Intelligence. The Internet is a vast field where agents proliferate and help in various tasks imperceptible from the viewpoint of the user. In this project, heuristic search techniques and a prototype of game will be implemented, in this game an intelligent agent will be programmed to find the best way out of a maze.

Keywords: *artificial intelligence, intelligent agent, heuristic games.*

SUMÁRIO

LISTA DE ILUSTRAÇÕES

1. INTRODUÇÃO

O uso da tecnologia é extremamente necessário, hoje principalmente. A partir de seu estudo na área de informática, surgiram sistemas de gestão empresarial, que são muito utilizados para se ganhar tempo na organização de estoques e documentos, cálculos que levariam horas podem ser resolvidos em segundos. Mas não apenas como forma de se ganhar tempo em organizações ou cálculos, a tecnologia, através dos anos, foi muito utilizada para trazer conforto à humanidade.

Juntamente com o crescimento da área o conceito e a valorização estão se tornando sendo cada vez maiores. Uma das grandes áreas e muito procurada por cientistas é a Inteligência Artificial. São várias as aplicações que aderem ao uso de sua desta tecnologia, como jogos, programas de computador, aplicativos de segurança para sistemas informacionais, robótica, dispositivos para reconhecimentos de escrita à mão e reconhecimento de voz, programas de diagnósticos médicos e muito mais.

Inteligência Artificial (IA) é um ramo da ciência da computação que se propõe a elaborar dispositivos que simulem a capacidade humana de raciocinar, perceber, tomar decisões e resolver problemas, enfim, a capacidade de ser inteligente (Ciriaco, 2008).

Este projeto envolve estudar algoritmos consistentes com características associadas à inteligência comportamental dos seres humanos. Essas características se dão pela compreensão da linguagem, o aprendizado, o raciocínio e resolução de problemas. Esses algoritmos inteligentes são denominados “agentes” (Russell e Norvig, 2004).

O projeto tratasse de encontrar uma a solução do problema do labirinto, que aqui representa também um problema de logística. Ficando a critério do algoritmo de busca desenvolvido, conseguir, através de recursos e conhecimentos próprios, percorrer o “labirinto” de um grafo e encontrar a melhor solução para o problema. Desta forma ele será capaz de atingir o final do labirinto (saída) percorrendo o caminho de menor custo.

1.1. Motivação

A partir da facilidade do acesso a informação disponibilizada pela internet e a equipamentos eletrônicos como computadores, celulares, e *tablets*, o mercado de jogos tem se

valorizado cada vez mais, aumentando a procura de empresas de desenvolvimento de jogos por desenvolvedores capacitados.

O mercado brasileiro de TI movimentou 123 bilhões de dólares em 2012 e fechou o ano com crescimento de 10,8% em 2012, na comparação com 2011, o que representa praticamente o dobro do aumento médio mundial, 5,9%. No comparativo com outros mercados, o avanço do setor no País ficou atrás apenas do verificado na China, de 15%, segundo dados da IDC. Com o resultado, o Brasil já é considerado o quarto maior mercado de tecnologia da informação e comunicação (TIC) do planeta e o sétimo em TI (ComputerWorld, 2013).

O mercado brasileiro de jogos que em 2011 movimentou R$840,00 milhões e o quarto maior do mundo, e crescerá em média 7,1% por ano até 2016. Para comparação, o país tem o quinto lugar no setor de automóveis e é o décimo no ramo de cinema. A arrancada é recente e tem vários motivos: queda nos preços de consoles e jogos, traduções para o português e a situação complicada do mercado mundial que passa por crise financeira e saturação. De cada 100 brasileiros, 23 jogam games (cerca de 45,2 milhões de pessoas), a historia de que games era brincadeira de meninos já não vale mais, as mulheres já dominam cerca de 47% do mercado (Orrico, 2012).

Tarqüínio Teles, presidente da Hoplon Infotainment, afirma que: “O desempenho no desenvolvimento de novas tecnologias tem sido tão bom que o Brasil está se tornando a nova ‘menina dos olhos’ da indústria mundial”. Uma das razões é que, enquanto o mercado dos Estados Unidos se retrai (segundo a NPD Group, houve diminuição de 8% em 2009 e 1% em 2010), o brasileiro apresenta um crescimento acelerado (entre 20% e 30% no mesmo período, segundo a IDG Consulting Brasil) (Correio Braziliense, 2011).

1.2. A Escolha do Tema

A escolha do tema foi realizada a partir de duas sugestões apresentadas pelo orientador, que após um breve estudo sobre os tais temas, decidiu-se que haveria um melhor entendimento e um melhor aproveitamento sobre o trabalho de buscas heurísticas.

Com toda a pesquisa em jogos, foi vista a necessidade do uso de um navegador para o usuário se localizar no cenário do jogo e encontrar rotas para chegar ao seu objetivo rapidamente. Houve também, a preocupação de demonstrar exemplos de jogos que fazem uso desse navegador.

Em suma o tema gira em torno de uma questão, onde foram realizadas pesquisas para a execução de toda a teoria que foi vista ao longo do projeto.

1.3. Problematização

Um ponto crucial para o início de toda a pesquisa, foi encontrar um objetivo à ser estudado. No caso deste, o problema é encontrar o melhor caminho para se movimentar de um ponto a outro.

A dificuldade para o encontro do melhor caminho serviu de incentivo para a realização. Em um labirinto pequeno é fácil visualizar qual seria um caminho possível para chegar ao objetivo, mas para saber se este é o melhor caminho já não seria tão simples. Em jogos, o cenário é um ambiente grande, onde para o jogador simplesmente olhar e descobrir um caminho para o seu objetivo já tornasse uma tarefa uma pouco quanto complicada, um algoritmo que pudesse indicar a ele qual seria esse caminho e ainda melhor, qual o melhor caminho, seria muito útil e economizaria muito tempo do jogador em transitar pelo cenário do jogo.

1.3.1. Formulação do problema

A ideia de estudar este problema de solução de caminho surgiu através do interesse em jogos, os mesmos vêm avançando cada vez mais em questão de tecnologia e um exemplo que chamava atenção era a inclusão de painéis de navegação em muitos jogos de diferentes estilos.

O jogo *Snake*, que é conhecido por jovens e adultos, pois não é recente e é simples de ser manuseado, ele foi citado como exemplo também por ser um jogo que abrange uma busca por comida, onde a cobra deverá chegar até o alimento pelo caminho mais curto.

1.3.2. Alternativa para resolução do problema

O uso da IA foi proposto como uma forma de resolução por ter a capacidade de resolver problemas e ter a capacidade de raciocínio como de um humano.

Quando a execução do trabalho estava em andamento, foram encontrados e estudados algoritmos que tinham a possibilidade de preencher o quadro de problematização, e conseguir solucionar a questão dos jogos. Esses algoritmos são conhecidos como Busca Gulosa e algoritmo A* (star).

Esses algoritmos serão encaixados na problematização, mas como solução para encontrar a melhor saída para cada situação. No exemplo do jogo Snake, implementado no projeto, deve ser utilizado a busca Gulosa, pois no jogo não há custos de caminho, o que é necessário para a busca em A*.

1.4. Objetivos

Como objetivo serão implementados métodos de buscas para que o agente trace um caminho a partir de um ponto determinado pelo usuário, até o destino que também será determinado pelo usuário.

1.4.1. Objetivo geral

Implementar um mecanismos de busca com o uso de técnicas de IA, usando como problema base o Labirinto e com isso, coletar resultados a fim de analisar o desempenho do agente e verificar se o melhor caminho de fato foi o escolhido.

1.4.2. Objetivo específico

- Analisar e compreender o andamento e os processos da IA;
- Pesquisar as soluções, ou melhor, os algoritmos para melhor compreensão do tema e da problematização;
- Exemplificar os métodos utilizados para melhor entendimento do leitor;

1.5. Metodologia

Para as pesquisas serem realizadas e o trabalho totalmente desenvolvidos, foram usadas ferramentas de estudos e apoios, sobre o tema, como a internet, reportagens, trabalhos semelhantes, artigos, entre outros itens.

Na internet, foi verificada a viabilidade, e a possibilidade para a realização do tema, seu desenvolvimento e a história da Inteligência Artificial, no mesmo meio verificou-se as condições para que o projeto fosse realizado.

A linguagem de programação escolhida para a implementação do projeto é a Java, devido a sua popularização na área de programação e sua crescente valorização comercial, Java demonstrou ser uma melhor opção para se concretizar este projeto.

> "É possível, e fácil, desenvolver jogos 2D e 3D em Java, para o desktop. Entretanto, não podemos dizer que é uma opção comum. O mercado de jogos triplo A exige otimizações extremas, e boa parte das produtoras já tem uma gigantesca base instalada em C++, sendo difícil para eles uma mudança nessa altura do campeonato. Um exemplo de jogo grande feito em Java é o brasileiríssimo Taikodom." (Mendonça, 2009).

1.6. Estrutura do Trabalho

O trabalho é composto por 6 (seis) capítulos, e cada um possui informações de importância semelhante para complementar o projeto.

No Capítulo 1 (um) foi relatado um pouco do que seria apresentado, dos objetivos, do problema considerado, que seria o encontro de meios para a saída de um labirinto e do argumento de solução, que são os algoritmos pesquisados.

O capítulo 2 (dois) tratará um pouco da história da Inteligência Artificial, do que é, de como ela cresceu, de como seu conceito aumentou com os anos, com o capitulo dois tem-se noções do que abrange esse tema, do que são sistemas que pensam e que agem como humanos.

No capítulo 3 (três), inicia-se a delimitação do tema, falando sobre as estratégias de busca, busca em largura, em profundidade, a busca Gulosa e a busca em A*. Assim como os outros capítulos o terceiro tem explicações em figuras, para facilitar o entendimento do leitor.

O capítulo 4 (quatro) é uma explicação sobre o labirinto, que foi o exemplo escolhido para a realização do trabalho. Para o projeto foram citadas duas buscas heurísticas, busca gulosa pela melhor escolha e busca em A*, usamos também cálculos para compreensão de resultados.

O capítulo 5 (cinco) descreverá o funcionamento de dois jogos criados a partir das classes de busca heurística produzidos no projeto. Um dos jogos se trata das mudanças no cenário do jogo, onde o agente tem a necessidade de encontrar uma nova trajetória, o outro fala de um jogo conhecido como por Snake, onde uma cobra é adicionada ao labirinto e a mesma deve chegar até o objetivo.

Já o capítulo 6 (seis) é a conclusão, onde será descrita algumas conclusões obtidas com o projeto.

Por fim, a Bibliografia, que mostra as referências, de onde foram retiradas as informações para o projeto, ou melhor, de onde se obteve a linha de pesquisa.

2. INTELIGÊNCIA ARTIFICIAL

Primeiramente, a Inteligência Artificial (IA) é um tema que cresce a cada dia, por ser estudada constantemente, possui avanços remotos e moderniza-se continuamente, por isso existem diversos “tabus” quanto ao surgimento da IA, que de fato, seu nascimento aconteceu em 1956.

Inteligência Artificial (IA) é um ramo da ciência da computação que se propõe a elaborar dispositivos que simulem a capacidade humana de raciocinar, perceber, tomar decisões e resolver problemas, ou seja, a capacidade de ser inteligente (Ciriaco, 2008).

Nesse trecho Ciriaco explica um pouco mais sobre o assunto, diz que a ciência da computação, por sua vez, desenvolve dispositivos para parecer o raciocínio humano e tomar decisões sem a necessidade de manejos humanos.

A IA é uma das áreas mais recentes da computação e vem sendo muito trabalhada devido ao grande avanço da tecnologia e dos estudos de cientistas a fim de proporcionar mais conforto e comodidade aos humanos, criando máquinas que possam fazer tarefas difíceis e que tenham capacidade de raciocinar e tomar decisões sozinhas, um exemplo desse grande avanço é o Robô “Capybara" ("Capivara", no português), demonstrado na figura 1.

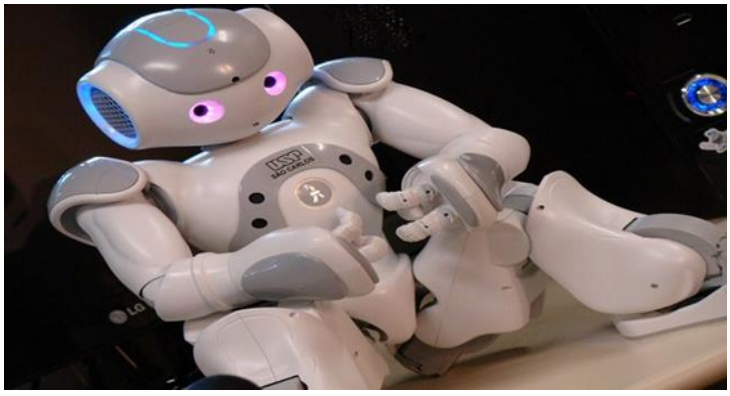

Figura 1 - Robô Capybara (Foto: Divulgação / USP São Carlos).

Esse robô argentino tem habilidades, como, capacidade de raciocinar e tomar decisões sozinho, e ainda trabalha em grupo, o denominado “Capivara” em português, foi batizado com esse nome pelo tamanho, aparência o modo de se movimentar sobre o campo, isso lembra os roedores que vivem no Pampa argentino (Rede Globo Universidades, 2013).

A IA abrange uma enorme variedade de subcampos, desde áreas comuns como aprendizado e percepção, até demonstrações de teoremas e complementações de jogos.

2.1. Ramos da IA

Pode-se dizer que a IA, tem definições que se destacam em meio ao trabalho, são divididas em quatro:

a) Sistemas que pensam como humanos;

b) Sistemas que agem como humanos;

c) Sistemas que pensam racionalmente;

d) Sistemas que agem racionalmente.

As primeiras (A e B) são denominadas empíricas, ou seja, envolvem uma série de hipóteses e necessitam de experimentos, para ver o decorrer das etapas de criação. Já as outras duas (C e D) envolvem mais matemática e engenharia.

2.1.1. Sistemas que pensam como humanos

São sistemas que abordam a representação do conhecimento e que podem aprender, ou melhor, armazenar os resultados para conhecimento próprio.

"A automação de atividades que nós associamos com pensamento humano, atividades tais como tomada de decisão, resolução de problemas, aprendizagem..." (Bellman, 1978 apud Russell e Norvig, 2004).

Para melhor entendimento, a automação é um sistema automático de controle, em que os mecanismos projetados verificam o próprio funcionamento, por isso associamos a IA com a inteligência humana, pois a mesma tem essa autonomia de funcionamento.

Para fazer com que um programa pense de forma humana, tem-se que primeiramente determinar como os seres humanos pensam. Se os resultados e os passos utilizados pelo programa coincidirem com os do ser humano, pode-se então considerar o sistema como um sistema que pensa como humano (Russell e Norvig, 2004).

Essa teoria, diz que para programar uma máquina como os seres humanos pensam, tem que saber o que eles pensam, pois do contrário não se pode dizer que o pensamento seja o mesmo. A minuciosidade dos detalhes dá diferença na hora da comparação dos pensamentos.

Alguns estudiosos desta área, como Allen Newell e Herbert Simon, que desenvolveram o GPS. Para eles, encontrar o resultado certo não era o suficiente, o mais importante era os passos de raciocínio do programa coincidir com o raciocínio humano (Russell e Norvig, 2004).

O sistema de navegação utilizados em GPS foi um exemplo de máquina pensante, onde o mesmo traça um caminho para o destino escolhido, além de dar o caminho mais curto, hoje em dia em suas atualizações, com o comando correto, consegue-se encontrar caminho com menos pedágios, com mais postos de gasolina, dependendo da necessidade de cada usuário.

Nos jogos de consoles de vídeo game, computadores e aparelhos mobile, o uso desses sistemas de navegação para orientação aumenta cada vez mais, para uma chegada rápida a um ponto determinado. Para exemplo tem-se jogos como: “Forza Horizon”, que é um jogo de corrida de carros e possui este sistema, a figura 2 representa um determinado momento do jogo.

Figura 2 - Imagem do jogo Forza Horizon (http://games.softpedia.com/).

Em jogos a navegação tem funções específicas, como, localizar alguma coisa e poder marcar o lugar aonde se quer chegar, assim o próprio jogo demarca o caminho que se deve percorrer.

Em jogos evoluídos como no exemplo do Forza Horizon dá para perceber na figura 2 que no chão por onde o carro passa, tem uma faixa verde, isso acontece por causa da velocidade do carro, ao oscilar a velocidade, a cor muda, para mostrar ao condutor onde ele tem que diminuir a velocidade. O sistema de navegação realiza a busca com uma estimativa de tempo, pois se leva em consideração a mudança de velocidade.

Outro exemplo, conhecido pelos jogadores, é o Euro Truck Simulador, esse também é um jogo onde se utiliza os comandos do sistema de navegação conforme demonstrado na figura 3.

Figura 3 - Imagem do jogo Simulador Euro Truck (http://euro-truck-simulator-2-patch.apportal.com.br).

O Euro Truck também possui as funções do sistema de navegação, os jogadores tem missões a realizarem em certo ponto do mapa, e com a ajuda deste sistema é traçada uma rota com a menor distância, caso o jogador saia da trajetória o caminho é refeito, porém a distância é maior.

2.1.2. Sistemas que agem como humanos

Os sistemas que agem como humanos são intrigantes, para leigos, e absurdamente interessantes para estudiosos que querem cada vez mais aprimorar certos trabalhos, que julgam nunca ter fim.

"O estudo de como fazer computadores realizarem coisas em que, no momento, pessoas são melhores." (Rick e Knight, 1991 apud Russell e Norvig, 2004).

Para avaliar a capacidade de uma máquina exibir comportamento inteligente equivalente a um ser humano, foi introduzido um "teste", denominado Teste de Turing, em que o computador será programado para se passar por um ser humano, e o mesmo confirmará sua posição. No final do teste o computador define se o interrogador é humano ou uma máquina, se ele não conseguir definir quem é quem, conclui-se que o computador pode pensar, segundo esse

teste. O teste não tem como objetivo ver apenas acertos ou se a máquina deu apenas respostas certas, ele tende a perceber, quão perto chegou do acerto completo. O “chat” com ambos são restritas em canais de texto.

O Teste de Turing, proposto por Alan Turing (1950), foi projetado para prover uma definição satisfatória de inteligência. Um computador passa no teste se um humano interrogador, depois de colocar algumas questões escritas, não puder dizer se as respostas escritas são de uma pessoa ou não (Russell e Norvig, 2004).

Como o informado, esse teste tem como finalidade, diferenciar a inteligência humana e a artificial, caso ele consiga, significa que a artificial não está compatível com a humana.

2.1.3. Sistemas que pensam racionalmente

Esse Sistema mede o sucesso com a comparação de idealização de uma inteligência, denominada racionalidade. Por exemplo, o usuário possui informações que são passadas para o sistema para facilitar o trabalho, ao final, os resultados do sistema devem coincidir com os resultados calculados pelo usuário anteriormente.

“O estudo de faculdades mentais por meio do uso de modelos computacionais.” (Charniak e McDermott, 1985 apud Russell e Norvig, 2004).

Há estudos amplos sobre o raciocinar, agir, entre outros, esses mesmos assuntos, são vistos para estudos desses mesmos itens, porém, com computadores.

“O estudo de computações que fazem possível perceber, raciocinar e agir.” (Winston, 1992 apud Russell e Norvig, 2004).

Como uma forma de tentar modificar o pensamento humano, tem-se como referência o “SPA”, essa sigla nada mais é, do que nomes de três, dos filósofos mais famosos, por tentar e persistir em mudar o pensamento humano com suas teorias. Sócrates, Platão e Aristóteles, são uma sequência de raciocínio, ambos seguem a mesma linhagem de pensamentos, pois eles formam três gerações de ideias, como aluno e professor, um dando continuidade as teorias do outro, nesta mesma sequência da sigla.

O filósofo grego Aristóteles foi um dos primeiros a tentar codificar o “pensamento correto” e seus silogismos deu início ao estudo da lógica, “Sócrates é um homem; todos os homens são mortais; então, Sócrates é mortal”. A chamada tradição logicista dentro da inteligência artificial espera criar sistemas inteligentes a partir do desenvolvimento destes programas lógicos (Russell e Norvig, 2004).

O logicismo é quando a matemática como um todo, ou parte dela, se reduz para uma parte da lógica ou para a lógica como um todo, sendo assim a logicista dentro da IA aguarda a criação de máquinas inteligentes a partir do desenvolvimento lógico do programa.

2.1.4. Sistemas que agem racionalmente

Um Sistema "agente" é uma sequência do Sistema pensante, como dito no próprio nome, agentes pensantes, pensam como humanos, os "agentes" agem como tais.

Um agente é algo que age, no entanto, espera-se que um agente computacional tenha outros atributos, tais como operar sob controle autônomo, perceber seu ambiente e adaptar-se a mudanças. Um agente racional é algo que age de forma a alcançar o melhor resultado (Russell e Norvig, 2004).

Nesse esclarecimento, pode-se perceber que um agente racional age em função do melhor resultado dependendo de cada situação, e para o agente computacional, é desejado que ele se adeque a cada situação passada, que caso não seja de sua comum vivência, que mude de acordo.

"Inteligência Computacional é o estudo do projeto de agentes inteligentes." (Poole et. al., 1998 apud Russell e Norvig, 2004).

Para explicação mais soberana, como todos sabem, o computador possui uma "linguagem", para que ele consiga compreender o que se diz, ele necessita de uma linguagem mais específica. Para melhor interpretação entre homem e a máquina, foram criadas essas falas de programação. Mesmo as próprias linguagens usam uma lógica para serem escritas e isso é chamado de algoritmos.

A citação de Poole acima é um complemento do estudo, visto que a Inteligência Computacional, está ligada ao tema de agentes inteligentes, pois aborda a teoria e a aplicação de técnicas computacionais inspiradas em fenômenos que ocorrem naturalmente, como por exemplo:

- Redes Neurais: que compreendem computadores que tentam definir o cérebro humano, para tentar apresentar características parecidas com a do corpo de um ser humano, tais como: aprendizado, generalização e abstração.
- Lógica Fuzzy: serve para tentar modelar o raciocínio, tentando igualar a habilidade humana de tomar decisões racionalmente.

- Computação Evolucionária: São algoritmos probabilísticos, ou melhor, Algoritmos Genéticos, que oferece um mecanismo que busca adaptativamente, baseado no princípio Darwiniano da evolução das espécies.

Ambas explicações são aplicadas com sucesso em diversas áreas da engenharia e tecnologia, resolvendo problemas que eram julgados sem soluções, ou que métodos mais conhecidos, não tinham muito sucesso.

Os Sistemas Computacionais feitos com essa tecnologia são os mesmos utilizados para sistemas que apoiam decisões, classificações, planejamento, otimização, entre outros dados e de resumo de sistemas.

Uma das formas de agir racionalmente é raciocinar de modo lógico até a conclusão de que uma dada ação alcançará as metas pretendidas, e depois agir de acordo com essa conclusão. A inferência correta não representa toda a racionalidade, pois com frequência, ocorrem situações que não possuem ações comprovadamente corretas, mas algo precisa ser feito (Russell e Norvig, 2004).

2.2. História da Inteligência Artificial

O primeiro trabalho que se tem conhecimento, como Inteligência Artificial foi feito por Warren McCulloch e Walter Pitts(1943). Eles mostraram que qualquer função computável podia ser calculada por certa rede de neurônios conectados, e que todos os conectivos lógicos podiam ser implementados por estruturas de redes simples.

Alan Turing foi quem primeiro articulou uma visão completa da IA em seu artigo de 1950 intitulado “Computing Machinery and Intelligence”.

Para melhor explicação, o artigo citado no parágrafo acima foi traduzido por Fábio de Carvalho Hansen com o título de Computação e Inteligência, esse artigo propõe um pensamento, inicial, que é “pode a máquina pensar?”, dentre todas as definições de máquinas e pensamentos, esse artigo, coloca várias sugestões e argumentos, para não descontextualizar citarei alguns dos temas abordados pelo artigo:

- O jogo da imitação: que se refere a um “jogo”, onde os jogadores são: duas pessoas de sexo oposto (homem e mulher) e um interrogador, o intuito é descobrir qual é qual (qual é o homem e qual é a mulher) através de questões feitas para eles.
- As máquinas em questão no jogo: tal, fala sobre técnicas biológicas que servem para construir uma máquina pensante. Sobre as possibilidades e não possibilidades.

- Computadores digitais: Que se refere à ideia de que computadores digitais podem ser melhor explicados dizendo que máquinas possuem a intenção de proceder as operações, que podem ser realizadas por um computador. Que por sua vez, o computador está supostamente seguindo normas fixas, e não tem arbítrio para se esquivar delas em detalhe.

Entre outros pontos, o artigo “Computing Machinery and Intelligence” foi citado, pois é de inteira importância o primeiro parecer completo em artigo, sobre esse tema de IA.

Em um seminário de dois meses em Dartmouth no verão de 1956, os pesquisadores da Carnegie Tech, Allen Newell e Herbert Simon já tinham um programa de raciocínio, o Logic Theorist (LT). Durante o seminário houve um acordo para adotar o nome sugerido por McCarthy para o campo: inteligência artificial.

Foi o primeiro programa inteiramente inventado para copiar as habilidades de resolver alguns problemas do ser humano, alguns até consideraram esse como "o primeiro programa de inteligência artificial", apesar de que outros considerem o programa do jogo de damas de Arthur Samuel, antes desse.

Na IBM foram produzidos alguns dos primeiros programas de IA. Herbert Gelernter (1959) construiu o Geometry Theorem Prover, que podia demonstrar teoremas matemáticos. A partir de 1952, Arthur Samuel escreveu uma série de programas para jogos de damas que eventualmente aprendiam a jogar em um nível amador elevado, seu programa aprendeu rapidamente a jogar melhor que seu criador.

Arthur Samuel jogou consigo mesmo, milhares de vezes e ainda usou técnicas de aprendizagem por recompensa. Assim desenvolveu seu programa que assimilou sua capacidade de se auto avaliar (usada no algoritmo Minimax).

John McCarthy saiu de Dartmouth para o MIT, e lá contribuiu com realizações cruciais em 1958. McCarthy definiu a linguagem de alto nível Lisp, que acabou por se tornar a linguagem de programação dominante da IA. Publicou um artigo intitulado Programs With Common Sense, em que descrevia o Advice Taker, um programa hipotético que pode ser visto como o primeiro sistema de IA completo, foi projetado para usar o conhecimento com a finalidade de buscar soluções para problemas.

Esse Programa de senso comum, como é chamado, foi a primeira declaração de usar a lógica para representar dados em um computador. Dele provavelmente foi o primeiro papel de oferecer capacidade de raciocinar, o que foi chave para a IA. Em um artigo McCarthy defende que programas para manusear em certa formalidade apropriada, certamente em cálculos são declarações instrumentais comuns. Um programa básico tirará conclusões de imediato tirando como coordenada, uma listagem informativa.

Marvin Minsky, um aluno do departamento de matemática de Princeton, que junto com Dean Edmonds construíram em sua tese de doutorado o primeiro computador de rede neural, chamado de SNARC, em 1951. Minsky foi para o MIT, onde orientou vários alunos que escolheram problemas limitados cuja solução parecia exigir inteligência. Esses domínios limitados se tornaram conhecidos como micromundos. O mais famoso micromundo foi o mundo de blocos, que consiste em um conjunto de blocos sólidos colocados sobre uma mesa, como no exemplo citado na figura 4.

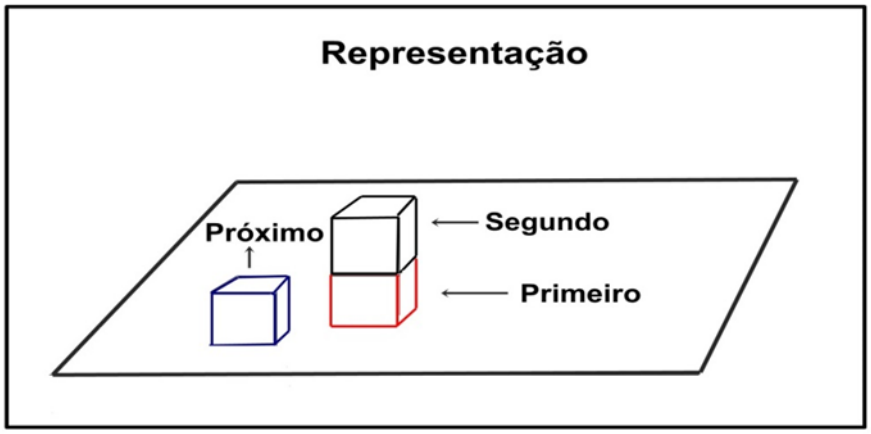

Figura 4 - Representação do braço mecânico, com o exemplo dos cubos

Um trabalho característico no mundo, representado na figura 4, é reorganizar os blocos de certa maneira. Como mostra a representação, utilizando a mão de um robô que pode erguer apenas um bloco por vez.

Os primeiros experimentos de evolução da máquina (agora chamados algoritmos genéticos) (Friedberg et al., 1959 apud Russell e Norvig, 2004) se baseavam na convicção, sem dúvida, correta de que, realizando-se uma série apropriada de pequenas mutações em um programa em código de máquina, seria possível gerar um programa com bom desempenho para qualquer tarefa simples. Quase nenhum progresso foi demonstrado.

O algoritmo genético (AG) é um instrumento de busca usado na ciência da computação, para chegar a soluções próximas de um determinado problema, ou seja, qualquer serviço que tenha como objetivo a otimização, tem um vasto número de componentes, dentre eles tem o espaço de busca, que são conceituadas quaisquer chance de solucionar algum problema e a função de avaliar ou custear, uma maneira de estimar os membros do espaço de busca.

Por um desenvolvimento muito vasto na área, foram criadas melhorias visíveis para empresas e para a humanidade, ou melhor, foram desenvolvidas táticas e sistemas para organizações.

O primeiro sistema especialista comercial bem-sucedido, o R1, iniciou sua operação da Digital Equipment Corporation (McDermott, 1982). O programa ajudou a configurar pedidos de novos sistemas de computadores; em 1986, ele estava fazendo a empresa economizar cerca de 40 milhões de dólares por ano. Essa entrada triunfal, refere-se a década de 80, pois com projetos reais, a I.A. começou a ser necessária e usada em industrias, promovendo cortes de custos e aprimoramento de tarefas.

De modo geral a industrial da IA se Expandiu de alguns milhões de dólares em 1980 para bilhões de dólares em 1988. Logo depois, veio um período chamado de "inverno da IA", em que muitas empresas sofreram à medida que deixaram de cumprir promessas extravagantes.

O "inverno de IA" foram épocas difíceis e repletas de precariedades, até os anos 80, Os cientistas tinham como prioridade a criação de androides que imitassem humanos completamente, com todas as suas reações. Os denominados "Ciborgues" naquele tempo eram precários não muito úteis. Foi o que deu início a esse tão difícil período, por causa de falhas e a falta de muitos recursos que eram necessários, pois governantes e instituições não tinham interesse de investir, então abandonaram a causa quase completamente.

Esse tema é composto por "períodos", pois são fases que estão mudando constantemente, houve após o inverno, a primavera de IA, que foram os projetos para conseguir apresentar a complexa estrutura e arquitetura cerebral, praticamente entraram em hibernação. Avanços recentes nos estudos que visam a mapear o cérebro, porém, combinados com o progresso na capacidade de processamento de computadores, voltaram a mudar a perspectiva. Podendo então dizer que se chegou à primavera da inteligência artificial. (Vilicic, 2013)

Grande parte do trabalho de redes neurais ainda nos anos 80 foi realizada na tentativa de definir a abrangência do que poderia ser feito e de aprender como as redes neurais diferem das técnicas "tradicionais". O campo chegou a uma compreensão tal que, agora, as redes neurais podem ser comparadas a técnicas correspondentes da estatística, do reconhecimento de padrões e do aprendizado de máquina. Como resultado desse desenvolvimento, a tecnologia denominada mineração de dados gerou uma nova e vigorosa indústria.

Pesquisadores começaram a examinar o problema do "agente como um todo". O trabalho de Allen Newell, John Laird e Paul Rosenbloom no SOAR (Newell, 1990; Laird et al., 1987 apud Russell e Norvig, 2004) é o exemplo mais conhecido de uma arquitetura completa de agente. O assim denominado movimento estabelecido tem como objetivo entender o

funcionamento interno de agentes incorporados a ambientes reais com entradas sensoriais contínuas. Um dos ambientes mais importantes para agentes inteligentes é a Internet.

SOAR é uma estrutura de uma cognição humana expressa na formação de um sistema de produção. Isto é, envolve o favorecimento de um tanto de pesquisadores considerável, contando com, Allen Newell, John Laird, Paul Rosenbloom entre outros, de várias instituições diferenciadas. A teoria é edificada em cima de muito empenho e de iniciativas que envolvem os pesquisadores Newell em conceito como GPS e GOMS (Card, Moran & Newell, 1983 apud Russell e Norvig, 2004).

Uma consequência da tentativa de construir agentes completos é a percepção de que os subcampos anteriormente isolados da IA talvez precisem ser reorganizados quando seus resultados tiverem de ser reunidos. Em particular, agora é amplamente aceito que os sistemas sensoriais não podem oferecer informações perfeitamente confiáveis sobre o ambiente. Em consequência disso, os sistemas de raciocínio e planejamento devem ser capazes de lidar com a incerteza.

Os sistemas tratados isoladamente, como sistemas sensoriais e Sistemas de raciocínio e planejamento, tem que ser reorganizados, tais tem que lidar com certas objeções, por exemplo, os sistemas sensoriais, não oferecem informações absolutamente confiáveis sobre o ambiente, sistemas de raciocínio e planejamento precisam saber trabalhar com a incerteza de Tendência de abordagem dentro da I.A. examinando problemas como um agente completo.

3. ESTRATÉGIAS DE BUSCA

A estratégia é uma arte de escolher onde, quando, como e com que travar uma batalha ou até mesmo encontrar uma forma de resolver problemas. A busca, tratasse da procura do fim de algo, ou quando traça-se um objetivo, vamos à busca do mesmo. Nesse capítulo, serão apresentados as definições de estratégia de busca, aplicada no tema do trabalho.

Alguns tipos de problemas para serem solucionados passam por determinados estados, gerando assim o que se conhece por árvores ou grafos, onde cada nó desta árvore ou grafo representa um estado. Para serem resolvidos esses tipos de problemas é realizada uma busca por todo o espaço de estados.

Existem técnicas de busca que utilizam uma árvore de busca explicita, gerado pelo estado inicial e pela função sucessora, que juntos definem o espaço de estados. A raiz da árvore de busca é um nó de busca correspondente ao estado inicial e o estado objetivo corresponde ao estado final da busca (Russell e Norvig, 2004).

Ao iniciar a busca, o primeiro passo é testar se o nó inicial é o nó objetivo, caso não seja, precisa-se considerar alguns outros estados, para isso aplica-se a função sucessor que irá se estender entre os estados que fazem ligação com o estado atual gerando um novo conjunto de estados. Este processo é realizado até que se encontre o estado objetivo ou até que não tenha mais estados para serem expandidos. A escolha de qual estado será ampliado é determinado pela estratégia de busca (Russell e Norvig, 2004).

Independente da estratégia utilizada, alguns fatores devem ser levados em consideração quando se está procurando uma estratégia de busca para resolver um determinado problema, algumas buscas podem não encontrar o melhor caminho para o seu problema, apenas o primeiro caminho possível, ou outras que poderiam se perder pelo caminho e nunca encontrar uma solução para o problema. Existem muitos tipos de estratégias de busca, elas são classificadas em dois tipos, buscas informadas e buscas não informadas.

3.1. Buscas Não Informadas

As buscas não informadas, também conhecidas como busca cega, são tipos de busca que não possuem nenhuma informação adicional sobre os estados, essas buscas podem apenas gerar sucessores e distinguir um estado objetivo de um estado não objetivo. As buscas não informadas

percorrem todos os estados possíveis de uma arvore ou grafo para encontrar a solução. Os algoritmos mais comuns são os de busca em largura e profundidade (Russell e Norvig, 2004).

3.1.1. Busca em largura

A busca em Largura, também conhecida como busca em extensão, é uma estratégia simples, que primeiro analisa o estado de seu nó atual e caso não seja seu objetivo, amplifica todos os seus sucessores e passa a analisar o próximo nó. Em geral, todos os nós de certa profundidade na árvore de busca são estendidos antes que os nós do nível seguinte sejam expandidos, isto pode ser mais bem entendido analisando a figura 5.

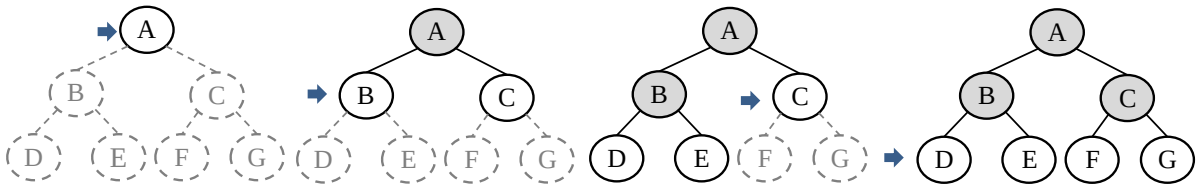

Figura 5 - Busca em largura

Esta é uma busca completa, pois são analisados todos os nós, e se o nó objetivo existir ele será encontrado. A busca em largura só pode ser considerada uma busca ótima se o custo do caminho não for necessário ou o custo de todos os caminhos forem iguais a 1, pois nem sempre o nó de menor profundidade é a melhor escolha, na busca em largura o custo do caminho percorrido não é analisado, assim o caminho escolhido poderá não ser o melhor caminho para a solução do problema. Outra desvantagem da busca em Largura é o gasto de tempo e memória para se encontrar a solução, pois não se tem uma estimativa de qual nó escolher primeiro e todos os nós amplificados precisam ser armazenados na memória porque fazem parte de um ancestral de um nó.

Uma técnica conhecida como **busca de custo uniforme**, foi criada a partir da técnica de busca em largura, mas que pudesse tratar o caminho de menor custo. Essa técnica ao invés de expandir o nó mais raso, ela alastra o nó de custo de caminho mais baixo, para a busca de custo uniforme não importa o número de passos do caminho, apenas o custo total do mesmo, por isso se houver um caminho de custo zero levando para o mesmo estado o algoritmo ficará em loop infinito. Para garantir que a busca seja ótima e completa, deve-se tratar o grafo para

que se tenha um custo de passo que seja sempre maior ou igual a alguma constante positiva pequena.

3.1.2. Busca em Profundidade

A busca em profundidade, após analisar o estado de seu nó atual, caso não seja seu objetivo, expande os nós sucessores e analisa primeiramente o nó de maior nível de profundidade até encontrar um nó que não tenha mais sucessores, e passa a analisar um nó de menor nível e que ainda não foi analisado. Se os nós já analisados não possuírem sucessores que ainda não foram analisados, eles são removidos da memória, pois não são mais necessários. Analisando a figura 6 pode-se ter uma melhor demonstração do funcionamento desta busca.

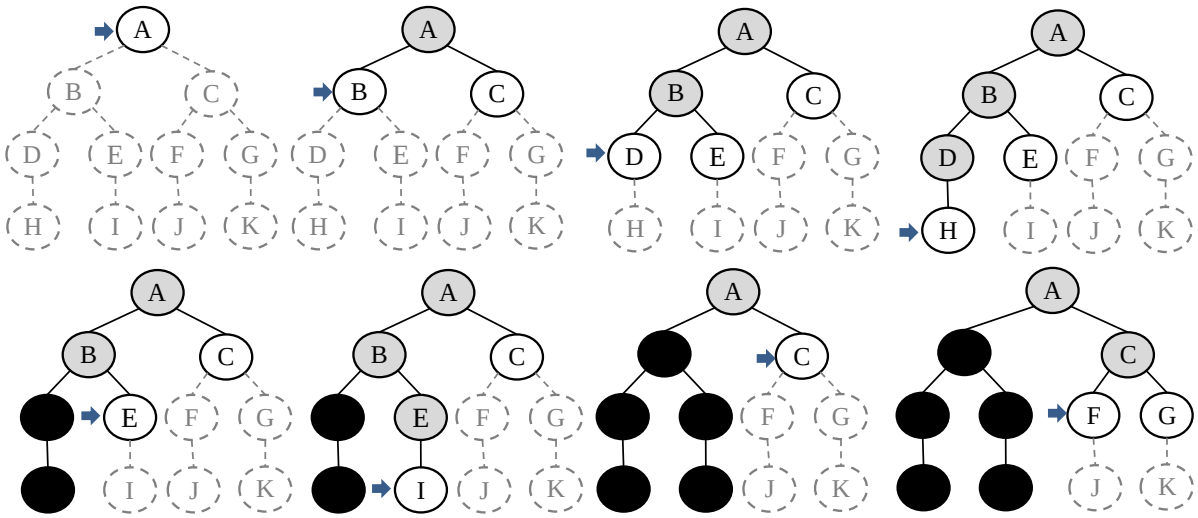

Figura 6 - Busca em profundidade

A busca em profundidade tem menos gasto de memória em relação à busca em largura, mas nem sempre essa busca é completa. Se a busca em profundidade não possuir um tratamento para verificar se está analisando um nó repetido, o algoritmo da busca poderá ficar preso em um loop infinito até causar um estouro de memória.

Uma tentativa de solucionar esse problema foi a implementação do algoritmo de **busca com profundidade limitada**. Esse limite é definido a partir do conhecimento do problema para limitar a profundidade máxima na qual a árvore deve se expandir, mas tratasse apenas de um palpite, e caso a profundidade for menor que a necessária para solucionar o problema, a busca

passa a não ser completa, pois serão analisados todos os nós possíveis e o objetivo não será encontrado.

As técnicas de busca em profundidade e busca em profundidade limitada não são buscas completas, com isso, foi criada outra solução para a busca em profundidade, a **busca com aprofundamento iterativo em profundidade**.

Esse algoritmo apenas adiciona um contador de profundidade à técnica de profundidade limitada. A busca inicia em nível 0, se o nó de nível zero não for o nó objetivo então remove-se o nó da memória e soma-se um no limite de profundidade. A busca inicia novamente com nível 1, então são analisados os nós até profundidade de nível 1 e assim por diante até encontrar o nó objetivo.

Essa técnica torna a busca completa, pois sempre será encontrado o nó objetivo se ele existir, e ótima se não for considerado o custo de caminho. Em relação à técnica de busca em largura, a técnica de aprofundamento iterativo tem menos consumo de memória, mas possui maior gasto de tempo, pois sempre que a busca chega ao limite de profundidade especificado a busca inicia novamente da raiz.

3.2. Buscas Informadas

As buscas informadas são parecidas com a busca em amplitude, com a exceção de que a busca não procede de forma uniforme a partir do nó-raiz da árvore, em geral utilizam funções heurísticas específicas para solucionar tipos diferentes de problemas. De acordo com informações específicas do problema, é possível definir uma ordem de preferência entre os caminhos possíveis a partir do nó-raiz.

As funções de busca informada também são baseadas em “objetivo”, porém durante uma pesquisa, a função heurística faz uma estimativa determinando quanto melhor é um caminho para atingir o nó objetivo em relação a outro caminho qualquer. São também conhecidas como busca pela melhor escolha ou busca heurística.

Uma heurística é uma função que quando aplicada a um estado retorna um número que corresponde a uma estimativa da qualidade do estado atual em relação a um estado final. Heurísticas podem subestimar ou superestimar a qualidade de um estado (Baranaukas, 2009).

Heurísticas que subestimam são desejáveis e são chamadas admissíveis, heurísticas admissíveis são otimistas, pois estimam que o custo da solução de um problema seja menor do que é na realidade.

Esses algoritmos de busca só são válidos junto a problemas para os quais, conforme aumenta à altura da árvore, o custo da solução também aumenta, ou seja, o custo de cada aresta da árvore não pode ser menor ou igual à zero.

Um componente fundamental para esses algoritmos é a função heurística, denotada por *h(n),* h é o custo estimado do caminho mais econômico do nó n até um nó objetivo. Existem várias técnicas de busca heurística, as mais populares são a Busca Gulosa e a Busca em A *, conhecida como A estrela.

Para exemplificar e comparar os resultados das buscas será utilizado as informações da figura 7, que representam a distância em quilômetros de algumas cidades, calculada pela função h(n), até a cidade de Bucareste, segundo informações retiradas do livro de Russell e Norvig, 2004.

Arad	366	**Mehadia**	241
Bucareste	0	**Neamt**	234
Craiova	160	**Oradea**	380
Dobreta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	224	**Zerind**	374

Figura 7 - Valores de h(n) - distância em linha reta até Bucareste (Russell e Norvig, 2004).

3.2.1. Busca gulosa pela melhor escolha

A busca gulosa tenta ampliar o nó mais próximo ao objetivo, na suposição de que isso provavelmente levara a uma solução rápida. Desse modo ela avalia nós utilizando apenas a função heurística f(n) = h(n).

A função h(n) no projeto, subestima a solução do problema, pois calcula a distância estimada do nó atual até o nó objetivo em linha reta, uma linha reta sempre é o caminho mais curto entre dois pontos quaisquer no plano.

A figura 8 representa o caminho percorrido pela busca gulosa da cidade de Arad até Bucareste utilizando apenas a função h(n) para encontrar o melhor caminho. O primeiro nó a ser expandido a partir de Arad será Sibiu, porque está mais próxima de Bucareste do que Zerind

e Timisoara. O próximo nó a ser amplificado será Fagaras por estar mais próximo do nó objetivo, que por sua vez gera Bucareste que é o nó objetivo.

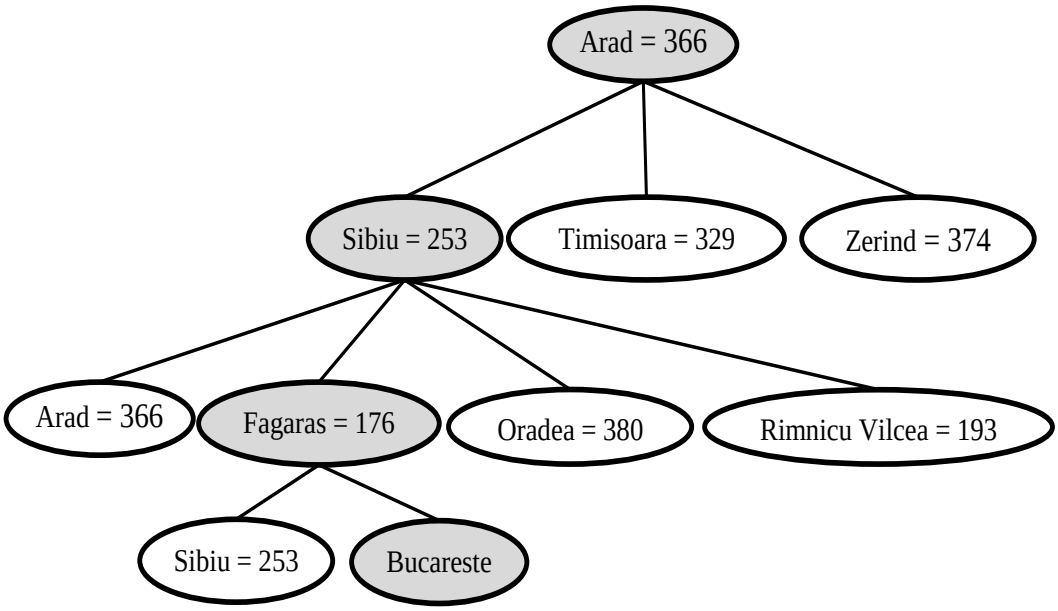

Figura 8 - Caminho de uma Busca Gulosa da cidade de Arad a Bucareste (Russell e Norvig, 2004).

A busca realizada na figura 8 encontrou uma solução sem precisar estender nenhum nó que não esteja no caminho da solução, consequentemente, seu custo de busca é mínimo. Mas ela não é ótima, pois o caminho até Bucareste passando por Sibiu e Fagaras são 32 quilômetros mais longo que o caminho por Rimnicu, Vilcea e Pitest. Por esse motivo o algoritmo é chamado de "guloso", porque expande sempre o nó mais perto do objetivo sem pensar no custo do caminho.

A busca Gulosa é parecida com a busca em profundidade, pelo fato de preferir seguir sempre um único caminho até o objetivo, mas voltará caso encontre um beco sem saída. Por esse motivo a busca gulosa também possui os mesmos defeitos, se o algoritmo não tiver um tratamento para verificar se algum estado já foi analisado, o algoritmo poderá nunca encontrar o objetivo. Por esse motivo a busca gulosa não é ótima e pode ser incompleta.

3.2.2. Busca em A*

A busca pela melhor escolha mais conhecida é a busca A*, ela avalia os estados de um problema combinando uma função que verifica o custo para alcançar cada estado, a uma função que verifica o custo para ir do estado n corrente até o estado objetivo (Russell e Norvig, 2004).

Função Heurística de A*: f(n) = g(n) + h(n)

Onde g(n) fornece o custo do caminho desde o estado inicial até o estado n corrente, e h(n) fornece o custo estimado do caminho de custo mais baixo desde o estado n corrente até o estado objetivo. Desse modo, se o objetivo é encontrar a solução de custo mais baixo, precisamos experimentar primeiro o nó com menor valor de g(n) + h(n).

A busca em A*, desde que a função heurística h(n) satisfaça a certas condições e seja admissível, a busca será completa e Ótima. Tendo em vista que g(h) é o custo exato para se alcançar n, então f(n) nuca ira superestimar o custo verdadeiro de uma solução passando por n.

A figura 9 demonstra o progresso de uma busca em A* da cidade de Arad até Bucareste onde os valores de h(n) são retirados da tabela 1, os nós em cinza representam os nós que foram alastrados e as setas numeradas representam a ordem em que cada nó foi analisado.

A busca iniciada em Arad ampliou três novas cidades, Sibiu, Timisoara e Zerind, no qual após análise da função heurística f(n) pode-se verificar que o melhor caminho a ser seguido até o momento era pela cidade de Sibiu. Sibiu deu acesso a mais 4 cidades, que repetindo o processo de análise da função, foi escolhido aumentou a cidade de Riminicu Vilcea. Reparando na figura 9, nota-se que o algoritmo como próximo passo não expandiu uma das cidades que se ligavam ao nó anterior, pois depois de analisar a função f(n) nestas cidades, a cidade de Fagaras mostrou-se como uma melhor opção. Após expandir Fagares surgiu a cidade de Bucareste que seria o objetivo do problema, mas que ainda não seria a melhor opção, por isso o algoritmo estendeu a cidade de Pitesti que também tinha a cidade de Bucareste como uma sucessora e com um custo de 32 quilômetros a menos. Com isso prova-se que a busca em A* é Ótima.

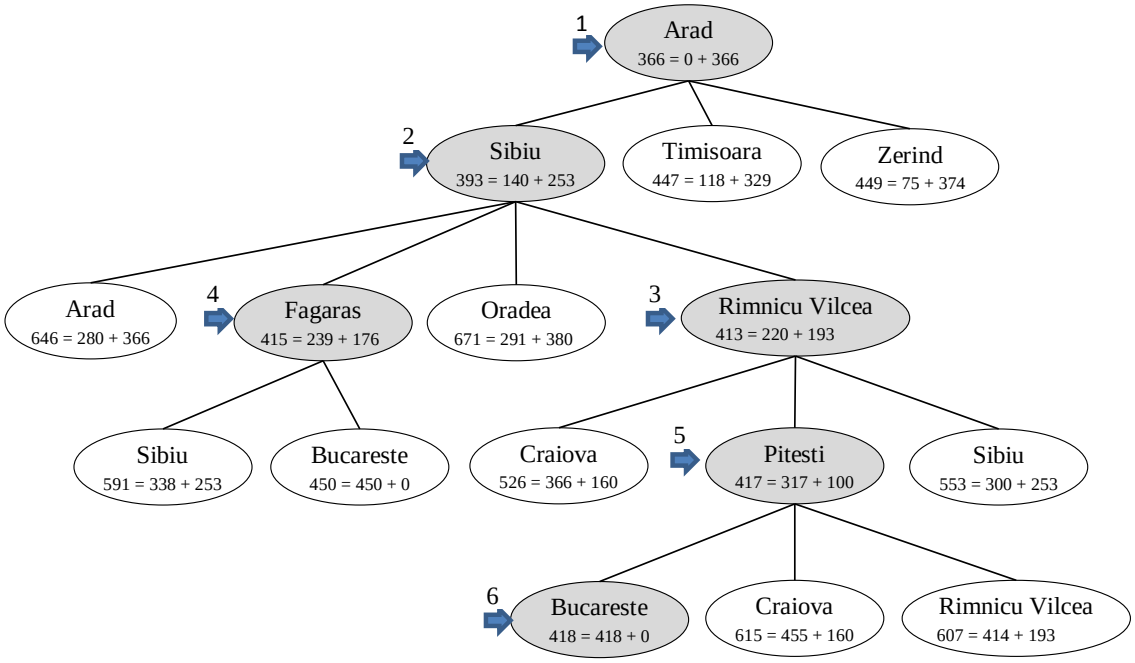

Figura 9 - Caminho de uma Busca A* da cidade de Arad a Bucareste (Russell e Norvig, 2004).

Entre algoritmos ótimos que estendem os caminhos de busca a partir da raiz, A* é otimamente eficiente para qualquer função heurística dada. Nenhum outro algoritmo ótimo tem a garantia de expandir menos nós que a A*, desde que sejam heurísticas consistentes.

> "Uma heurística é consistente se, para todo nó n e sucessor n' de n gerado por qualquer ação a, o custo estimado de alcançar o objetivo a partir de n não é maior que o custo do passo de se chegar a n' somado ao custo estimado de alcançar o objetivo a partir de n': $h(n) \leq c(n, a, n') + h(n')$" (Russell e Norvig, 2004).

O fato de a busca em A* ser completa, ótima e otimamente eficiente entre todos esses algoritmos é muito interessante, mas isto não significa que ela seja a solução para todos os problemas. Uma desvantagem da Busca em A* é o gasto de memória, pois ela precisa guardar na memória todos os estados dos nós anteriores, e em uma busca de grande escala ela não seria uma boa pratica.

Por esse motivo muitas vezes é preferível utilizar buscas não ótimas para resolver determinados problemas, pois uma busca ótima em grande escala poderia sobrecarregar qualquer computador.

4. O LABIRINTO

O jogo de labirinto proposto, desenvolvido em linguagem Java, consiste de um vetor dinâmico bidimensional simples e seu tamanho será informado pelo jogador, cada célula, ou nó, representa cada estado por onde o agente inteligente decidirá passar ou não para atingir o estado final (saída do labirinto).

Durante o estudo realizado, foi focada somente a parte funcional do sistema, não se teve uma preocupação quanto à parte gráfica do mesmo. Portanto, para esse sistema não existe uma visão gráfica atraente para a sua visualização. Nesse caso, para as paredes do labirinto, os caminhos a serem percorridos, o caminho percorrido e o melhor caminho serão levados em consideração apenas valores numéricos.

Como exemplo para o projeto, será utilizado como labirinto uma matriz de tamanho 12x12 representado na figura 10, onde os valores indicados abaixo foram escolhidos aleatoriamente:

- 0 representara os caminhos possíveis;
- 1 representara as paredes do labirinto;
- 3 representara a saída do labirinto;
- 5 representara o estado inicial do labirinto;
- 7 representara o caminho que for analisado pelo algoritmo;
- 9 representara o melhor caminho a ser percorrido.

Figura 10 - labirinto gerado pelo algoritmo

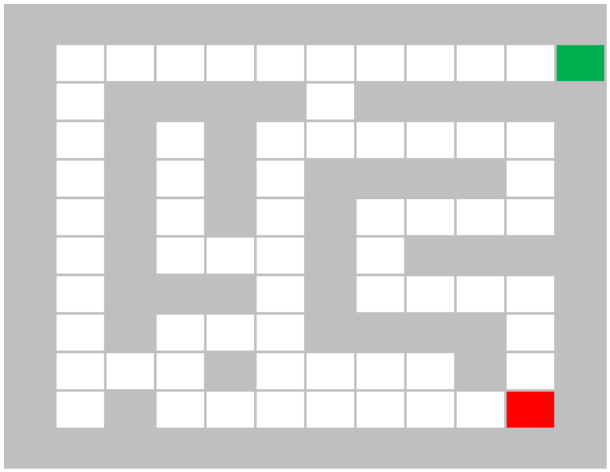

Figura 11 - labirinto criado para melhor visualização

Para uma melhor visualização do exemplo utiliza-se o labirinto da figura 11, que trata-se do mesmo labirinto da figura 10, com uma visualização gráfica melhor, onde:

- A parte cinza são as paredes do labirinto;
- A parte branca são os caminhos;
- O vermelho é a localização do agente inteligente;
- O verde é o objetivo, saída do labirinto;
- O amarelo será os estados analisados;
- O azul serão os estados percorridos.

Para o projeto foi implementado duas buscas heurísticas, busca gulosa pela melhor escolha, utilizando a função f = h e busca A*, utilizando a função f = g + h, desta forma pode-se realizar comparações entre as buscas em demonstrar o seu funcionamento em prática.

4.1. Calculando H

Calcular a função h é o primeiro passo para iniciar uma busca heurística, sabe-se que o menor caminho entre dois pontos A e B é um segmento de reta. Dado esses dois pontos, apresentando-os em um plano cartesiano, é possível encontrar o valor da distância entre eles. Deve-se calcular o custo h da distância entre um determinado ponto A e outro B considerando suas coordenadas (x, y) onde estão localizados, como mostra a figura 12.

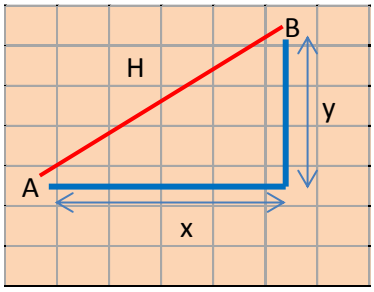

Figura 12 - Exemplo de um caminho de A até B

Para calcular h como sendo a distância de A até B então tem-se a função $h = \sqrt[2]{X^2 + Y^2}$. Para facilitar a programação do calculo, ao inves de precisar identificar qual o maior numero e subtrair pelo menor, utiliza-se o modulo de x e y. Desta forma a função final para calcular h seria:

$$h = \sqrt[2]{(|Xa - Xb|)^2 + (|Ya - Yb|)^2}$$

Esta função no programa é representada pela figura 13. Ao utilizar números para exemplificar esta função, lembrando que em uma matriz de um algotimo o valor de y aumenta de cima para baixo, e o valor de x aumenta da esquerda para a direita, de acordo com as cordenadas da figura 12 os valores de A e B seriam respectivamente (1,5) e (6,1).

$$h = \sqrt[2]{(|1 - 6|)^2 + (|5 - 1|)^2} = \sqrt[2]{5^2 + 4^2} = \sqrt[2]{25 + 16} = \sqrt[2]{41} = 6{,}4031242374$$

```
for(int i=0;i<desenho.length;i++){
  for(int j=0;j<desenho[i].length;j++){
    if(desenho[i][j] == 5 || desenho[i][j] == 0){
      // distx = |x1-x2|
      float distx = Math.abs(i-chegada[0]);
      // disty = |y1-y2|
      float disty = Math.abs(j-chegada[1]);
      // H = raiz quadrada(X²+Y²)
      double res = Math.sqrt(Math.pow(distx,2)+Math.pow(disty, 2));
      distancia[i][j] = res;
    }
  }
}
```

Figura 13 - Algoritmo do cálculo da função h

A figura 13 representa o calculo da função h utilizado no projeto que após identificar o ponto de inicio do agente inteligente e a saida do labirinto, calcula o valor h de todos os

caminhos possiveis do labirinto, demarcados pelo valor do tipo inteiro 0 e 5 antes de começar a analise de qual caminho seguir.

4.2. Calculando G

A função G é utilizada apenas pela busca heurística em A*, ela é responsavel por calcular o custo do caminho para atingir um determinado estado. Em um ambiente simples pode-se determinar o custo do caminho para todos os passos com valores iguais a um.

Em ambientes mais complexos, como em um jogo de labirinto pode-se determinar caminhos com obstaculos, como por exemplo, o custo de um caminho normal seria igual a 1, uma caminho com lama teria um custo maior, desta forma atraves da função g, o algoritmo poderá determinar se vale a pena ou não passar pelo caminho com lama, sendo ele mais curto, mas não necessariamente o melhor caminho. Analisando a figura 14 pode-se compreender melhor a importância do custo do caminho.

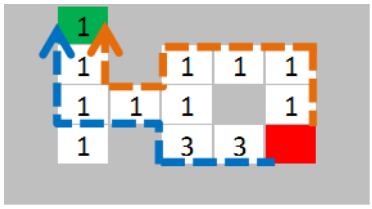

Figura 14 - Demonstração de custo de caminho

A partir da figura 14 pode-se identificar 2 caminhos possiveis, os valores na figura representam o custo de cada passo. O caminho azul seria o caminho mais curto e utiliza-se apenas 7 movimentos para atingir o objetivo enquanto pelo caminho laranja utiliza-se 9 movimentos, mas através do custo do movimento o caminho laranja passa a ser a melhor opção, pois por ele tem-se um custo total igual a 9 enquanto pelo caminho azul teríamos um custo total igual a 11. Portanto o caminho laranja seria a opção otima. No algoritmo, a verificação desta opção é demonstrada na figura 15.

```
andou[salvax+1][salvay] = andou[salvax][salvay]+custocaminho[salvax+1][salvay];
```

Figura 15 - Calculo de custo de caminho no algoritmo

A figura 15 representa a linha do algoritmo que calcula a função g, a variavel *andou* é uma matriz que armazena o valor do custo do caminho de um determinado estado somado com o valor para se atingir o mesmo.

4.3. Calculando F

O cálculo de f dá-se pela função f = g + h. Para esclarecer o funcionamento desta função será apresentado agora um exemplo simples levemente relacionado ao problema proposto.

Deve se imaginar um universo simples onde se parte de um ponto inicial pretendendo-se atingir um ponto final, determinados por A e B respectivamente. O ponto inicial A não tem um caminho ligado diretamente ao ponto objetivo, para atingir B partindo de A é necessário passar pelos pontos Z ou W.

Primeiramente calcula-se a diferença entre a coordenada x do nó a ser alcançado (B) e a coordenada x do ponto corrente (A, W ou Z), após calcular a diferença entre as coordenadas x, calcula-se tambem as diferenças entre as coordenadas y. Após obter os resultados das diferenças entre as duas coordenadas aplica-se a função $h = \sqrt[2]{X^2 + Y^2}$.

Agora que se tem conhecimento da distância entre cada ponto (A, W e Z) e o ponto final (B), assim é possivel determinar qual o caminho mais curto para o problema, mas ainda não da para se ter certeza se seria o melhor caminho. Para isso deve-se utilizar o cálculo de g em conjunto ao cálculo de h para se obter f, representado no algoritmo pela figura 16.

Após obter os valores de h e g a heurística deve comparar qual dos dois nós possuem o menor custo estimado.

```
F[salvax+1][salvay] = andou[salvax+1][salvay]+ distancia[salvax+1][salvay];
```

Figura 16 - Calculo da matriz F no algoritmo

A figura 16 representa a linha do algoritmo que calcula a função f na *classe* Aestrela, a variavel *F* é uma matriz que armazena o valor da matriz *andou* somado com o valor da matriz *distancia* de um deteminado estado.

4.4. O Agente Inteligente.

Neste capitulo será descrito o funcionamento do algoritmo desenvolvido para o projeto, optou-se por fazer a programação em Java e para isso foram criadas 4 classes:

- Parâmetros: classe que receberá os valores do usuário;
- Labirinto: classe responsavel por fazer a verificação das condições de uso do labirinto e acionar as classes Aestrela ou Gulosa de acordo com a busca escolhida.
- Aestrela: classe que realiza os processos da busca em A*.
- Gulosa: classe que realiza os processos da busca Gulosa.

A classe principal é a classe parâmetros, pois é ela que recebe os dados e retorna o melhor caminho para o usuário. Para criar o labirinto é necessario primeiramente que o usuario passe alguns dados para a classe parâmetros, como:

- Uma matriz *desenho* de valores inteiros e positivos: que se trata do formato do labirinto, demarcando as parades, os caminhos, o ponto inicial e final que deseja. Para o bom funcionamento do algoritmo, essa matriz precisa ter o mesmo tamanho para todas as linhas e o mesmo tamanho para todas as colunas;

- Uma matriz *custo* de valores inteiros e positivos, que demarca o custo de passo de cada posição da matriz *desenho*, por esse motivo as duas matrizes devem possuir tamanhos perfeitamente iguais;

- Uma variavel *busca* do tipo inteiro, que se trata do método de busca escolhido, um para utilizar a busca em A* e 2 para utilizar a busca gulosa.

Após definir as variáveis, é necessario acionar a classe Labirinto passando as variáveis *desenho, custo e busca* respectivamente nesta ordem, cujo as quais foram definidas anteriormente.

A classe Labirinto recebe os valores da classe parâmetros e cria as variáveis: início e chegada, cujo as quais armazenarão as coordenadas do ponto de início e o objetivo, detectados por uma busca realizada em toda a matriz *desenho*.

Depois de detectados os pontos de inicio e fim do sistema é realizado uma varredura na matriz *desenho* para verificar se a mesma não possui algum valor indesejado ou se tem tamanho inadequado. Os valores que poder ser utilizador são apenas os valores inteiros positivos 0 e 1, e apenas uma posição com valor 3 e uma com valor 5.

Se algumas das verificações forem negativas o algoritmo acusa uma mensagem referente ao problema encontrado e finaliza o processo, caso todas tenham confirmação

positiva, se a variável busca possui valor 1 é chamada a classe Aestrela passando as variáveis *desenho*, custo, início e chegada, se a busca possuir valor igual a 2 então aciona-se a classe Gulosa passando as variáveis *desenho*, início e chegada.

As classes mais importantes para o projeto são as classes Gulosa e Aestrela, são elas responsáveis por determinar o melhor caminho utilizando as funções h e g. As duas classes possuem o mesmo processo, mas com uma única diferença, ao determinar o valor da matriz *F*, conforme demonstrado na figura 16, na busca em A estrela é somado o valor da matriz *distância* e com a matriz *andou*, determinada pela soma da matriz *custo* com o custo de caminho já percorrido. Mas a busca gulosa não leva em consideração o custo do caminho, por isso não existe a matriz *andou* na classe Gulosa, por isso a matriz *F* recebe apenas o valor da matriz distância.

Após determinar os valores dos possíveis estados a serem expandidos e demarcar com o valor inteiro 7 na matriz *desenho* as posições já analisadas, uma varredura na mesma encontra qual o possível passo com o menor custo e movimenta o agente, salvando na matriz *anterior* do estado atual o valor das coordenadas x e y do estado anterior ao movimento.

Quando o próximo passo com menor custo for o objetivo, então a variável do tipo inteiro, cujo nome determinasse por *chegou*, recebe valor 1 e encerra o loop de verificação de caminho. Quando um caminho é encontrado, uma variável do tipo inteiro, determinada com o nome de *semcaminho*, recebe valor 0, isso faz com que ao chegar no final das verificações de caminhos possiveis, caso esta variavel ainda esteja com valor 1, o algoritmo se encerra e apresenta ao usuário uma mensagem indicando que não existi um caminho possível.

Ao atingir o objetivo, um loop cria o caminho analisando os valores da matriz *anterior*, demarcando na matriz *desenho* o valor 9 para indicar o caminho percorrido pelo agente e retorna para a classe Labirinto, então a classe Labirinto imprimi para o usuário a matriz *desenho*, com as informações, utilizando o loop demonstrado na figura 17.

```
for(int i=0;i<desenho.length;i++){
  for(int j=0;j<desenho[i].length;j++){
    System.out.printf("%d  ",desenho[i][j]);
  }
  System.out.printf("\n");
}
```

Figura 17 - loop para imprimir matriz desenho

4.5. Simulando o Labirinto

Para um melhor entendimento do algoritmo proposto, o mesmo será utilizado para resolver o problema da figura 11. Aplica-se a simulação de três métodos, primeiro com a busca gulosa, depois a busca em A* com custo simples, ou seja, custo 1 para todos os passos, e depois aplica-se uma simulação de uma busca com custos de caminhos diferentes.

4.5.1. Busca Gulosa

Após a classe labirinto detectar os pontos de início e fim do problema e acionar a classe Gulosa, o algoritmo fará uma busca em toda a matriz calculando o valor de h para todos os caminhos possíveis. O resultado pode ser verificado na figura 18.

10,00	9,00	8,00	7,00	6,00	5,00	4,00	3,00	2,00	1,00	
10,04					5,09					
10,19		8,24		6,32	5,38	4,47	3,60	2,82	2,23	
10,44		8,25		6,7					3,16	
10,77		8,94		7,21		5,65	5,00	4,47	4,12	
11,18		8,43	8,60	7,81		6,40				
11,66				8,48		7,21	6,70	6,32	6,08	
12,20		10,6	9,89	9,21					7,07	
12,80	12	11,3		10,00	9,43	8,94	8,54		8,05	
13,45		12,04	11,40	10,81	10,29	9,84	9,48	9,21	9,05	

Figura 18 - Calculo da distância dos caminhos

Após o término do cálculo dos caminhos o agente irá analisar o valor dos estados de seus possíveis movimentos, e seguir para o estado com menor custo. Conclui-se a partir da figura 18 que o agente terá analisado apenas dois movimentos possíveis, para a esquerda que teria um valor de 9,21 de distância e para cima que teria um valor de 8,05 de distância, como o caminho para cima é de menor custo aparentemente, o agente se movimentará para cima.

O agente repetirá as verificações até encontrar o seu objetivo, repare que em certo período será necessário que o agente escolha um caminho que tenha maior custo que o seu estado atual, esse processo só ocorre porque no algoritmo é realizado o tratamento para não percorrer estados repetidos, caso contrário, a busca gulosa nunca encontraria a solução para este problema de labirinto utilizado.

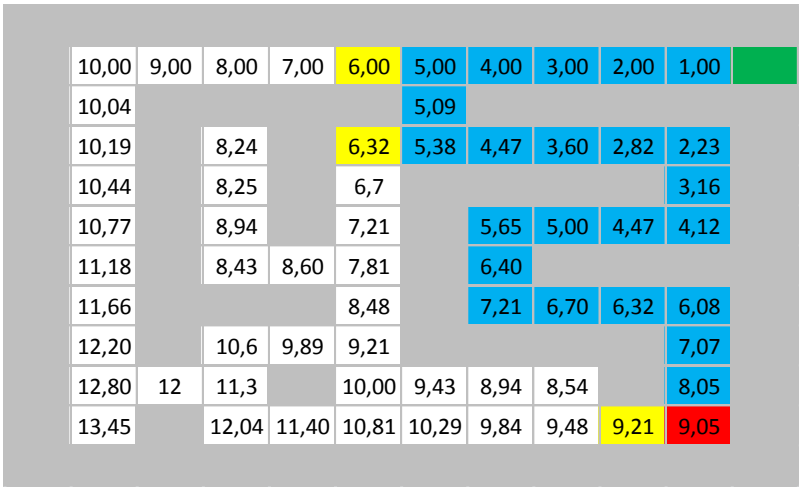

Figura 19 - Resultado da Busca Gulosa

A figura 19 representa o resultado obtido ao final da busca Gulosa, é possível reparar que o algoritmo obteve a solução com 24 movimentos e precisou analisar apenas 27 possíveis movimentos. Esta ainda não seria a solução ótima, pois a busca gulosa não considera o custo de caminho, esta questão será analisada na busca em A*.

4.5.2. Busca em A* com custo simples

Inicialmente o algoritmo da classe *Aestrela* realiza o mesmo processo da classe Gulosa, como calcular os valores da matriz *distancia*, a exceção desta classe é a análise do valor da matriz *F*, cujo qual, representa o valor da matriz *distancia* somado com o valor da matriz *custo* de caminho.

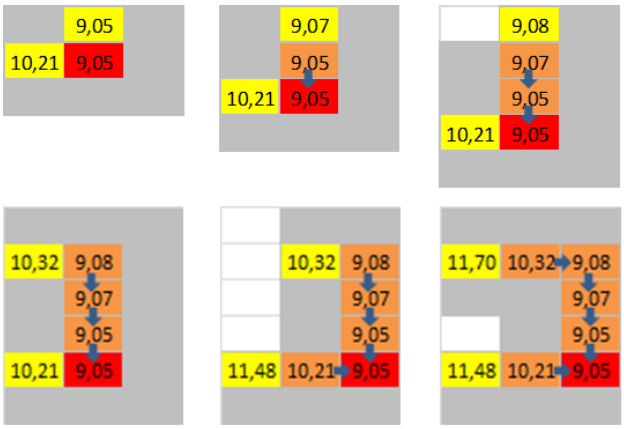

Figura 20 - Movimentação do agente na busca A*

A figura 20 representa à movimentação do agente, cujos valores representam o valor de “f” para um determinado estado, a cor vermelha representa a posição inicial do agente, a cor laranja representam os estados já analisados, a cor amarela representa os estados que estão sendo analisadas no momento, as setas indicam o estado anterior, e os espaços em branco representam estados ainda não analisados.

Como primeiro passo o algoritmo analisa os seus possíveis caminhos, representados na cor amarela no primeiro segmento da figura 20. Os caminhos possíveis são acima, com valor de f = 9,05, ou à esquerda com valor de f = 10,21. O caminho acima se mostra ser mais econômico ao algoritmo, por isso o agente o escolhe como próximo passo, mas o caminho à esquerda não está descartado, o mesmo permanece na memória para ser comparado com outros possíveis caminhos posteriormente. Este processo se repete até que o agente encontre seu objetivo.

Diferente do algoritmo da busca gulosa, os valores de “f” aumentam conforme o agente avança pelo labirinto, isto se explica pelo uso da função g. Como exemplo analisa-se o sexto segmento da figura 20, onde se tem dois possíveis caminhos em amarelo, um com valor de f = 11,48 e outro com valor de f = 11,70. Para o primeiro elemento, tem-se h = 9,48 e g = 2, pois o custo do passo é um e o custo para se chegar até este passo é igual a um, então f = h + g = 11,48. Para o segundo elemento, tem-se h = 6,70 e g = 5, pois o custo do passo é um e o custo para se chegar até este passo é igual a quatro, então f = h + g = 11,70. Portanto o caminho escolhido seria o primeiro elemento.

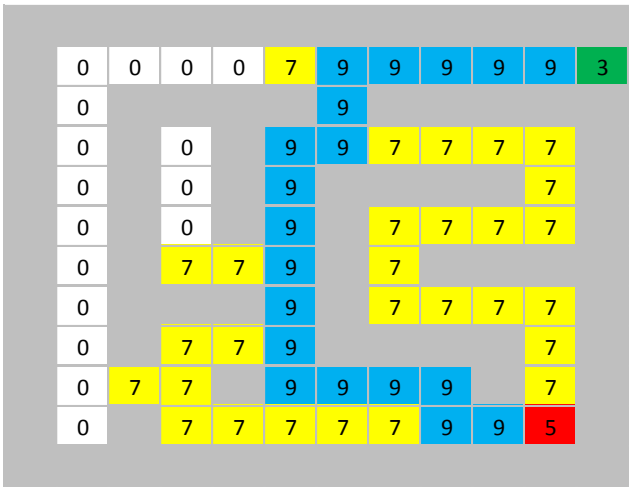

Figura 21 - Resultado da busca em A*

A figura 21 representa a matriz *desenho* como resultado final do algoritmo da busca em A*. A partir deste algoritmo pode-se encontrar o caminho ótimo para esse problema com um custo de caminho total igual a 20, como o exemplo tratasse de custo simples, ou seja, custo de caminho igual a 1 para todos os passos, a quantidade de movimento mínima para se chegar até a saída do labirinto é igual ao custo total de caminho, mas por outro lado, foi necessário analisar 48 estados, o que se teve um gasto de memória e tempo muito maior que através da busca gulosa.

4.5.3. Busca em A* com custo não simples

Se utilizar o algoritmo em A* com custos de caminho variados tem-se então, um resultado diferente do que foi visto nos exemplos anteriores. Para exemplificar, nos caminhos percorridos pelos algoritmos de buscas gulosa e A* acrescentou-se alguns passos com custo de caminho igual a 5. O resultado desta busca em A* pode ser conferido na figura 22.

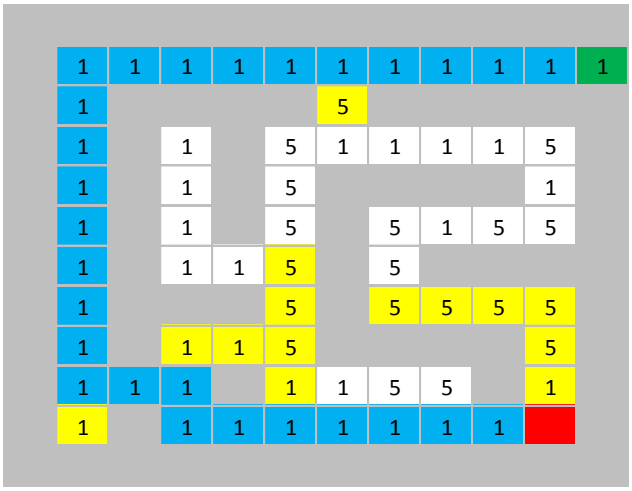

Figura 22 - Resulta da busca em A* com custo não simples

A figura 22 mostra o caminho percorrido pela busca em A*, e os valores na figura representam o custo de cada passo. Este caminho foi escolhido pelo algoritmo, pois teria um custo de caminho total igual a 28, que para este exemplo seria o melhor caminho possível, uma vez que o custo total dos caminhos escolhidos anteriormente pela busca gulosa e pela busca em A* com custo simples são respectivamente de 78 e 56, devido ao total dos custos de seus passos terem um custo maior, que os passos do caminho escolhido.

5. JOGOS COMPLEMENTARES

Este capítulo descreverá o funcionamento de dois jogos criados a partir das classes de busca heurística produzidos neste projeto. Um dos jogos trata-se de um método interativo de mudar o cenário para dificultar o trajeto do agente, o outro se trata de um jogo conhecido como Snake, que em português significa Cobra, possui esse nome, pois o jogo se trata realmente de uma cobra que se move em busca do alimento.

5.1. Labirinto Interativo

Este jogo não tem muita diferença em relação ao projeto principal, para ele foi adicionada apenas uma classe Movimento, que a cada passo realizado pelo agente o jogo pergunta ao usuário se o mesmo pretende alterar o cenário do labirinto adicionando um obstáculo. Esse método pode ser comparado, por exemplo, em um jogo moderno, onde o jogador está transitando pelo caminho apontado por seu sistema de vanegação e acontece um acidente no meio do caminho traçado, então o sistema precisara traçar uma nova rota.

A casa movimento realizado pelo agente, o jogo apresenta ao usuário o labirinto completo e pergunta se o mesmo pretende realizar alguma alteração, se ele escolher a opção de não alterar, o agente continuará a percorrer o caminho apontado pela heurística definida anteriormente, mas se o usuário escolher alterar o labirinto, então será solicitado ao usuário a posição da matriz que ele pretende alterar e informar o valor a ser alterado, 1 para inserir uma parede e 0 para inserir um caminho. Só é possível realizar uma alteração por vez, e ao inserir o valor e as coordenadas da matriz onde será realizada a alteração, o algoritmo realiza uma serie de verificações para comparar se os dados passados pelo usuário são compatíveis, o algoritmo verifica se o valor é 0 ou 1, pois os valor não pode ser inserido pois prejudicaria o funcionamento do algoritmo e se as coordenadas passadas existem na matriz.

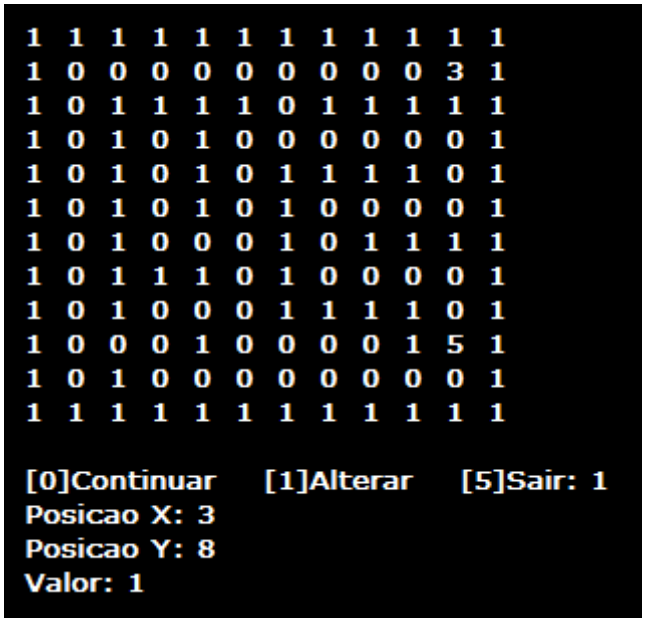

Figura 23 - Visualização do jogo do labirinto interativo

A figura 23 representa o momento em que o usuário passa as coordenadas de x e y, e valor igual 1, para inserir uma parede na posição (3,8) da matriz. O jogo encerra quando o agente chega ao objetivo ou quando o usuário escolhe a opção sair.

5.2. Jogo Snake

Para o jogo do Snake não se tem diferença nos resultados ao optar por escolher a busca em A* ou busca gulosa, pois neste jogo não se faz uso da matriz *custo*.

Para o jogo do labirinto, o usuário deverá acionar a classe Movimento passando apenas a matriz *desenho*, então o algoritmo irá gerar uma posição de início para o agente de forma aleatória e demarcar na matriz com o valor inteiro 5.

O agente ainda não possui um objetivo, ou seja, não tem nenhuma maça no jogo para a cobra comer, então o algoritmo solicita ao usuário que o mesmo digite uma coordenada x e y para inserir a maça com o valor inteiro 3. Caso o usuário escolha uma posição invalida da matriz, onde seria uma parede ou corpo da cobra no jogo, o algoritmo acusará uma mensagem e solicitará uma nova coordenada.

A partir de validado os dados para a busca, o algoritmo repetirá os movimentos do agente mostrando ao usuário cada passo realizado, até que a cobra chegue até a maça. Quando a cobra comer a maça o corpo dela irá aumentar, representado pelo número 4, aumentará uma unidade.

O jogo será encerrado quando o cobra não tiver um caminho possível para chegar até seu destino, ou quando o usuário digitar 0 nas coordenadas x e y para a maça.

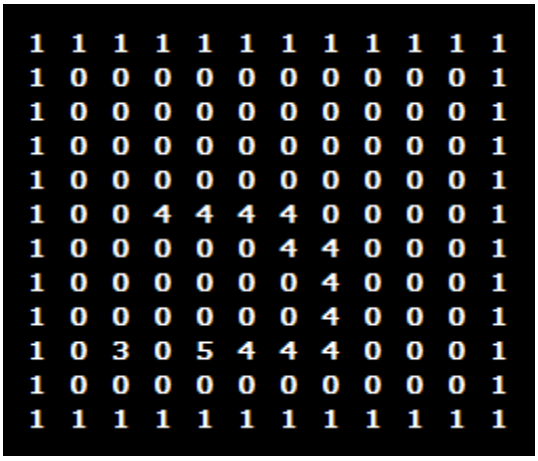

Figura 24 - Visualização do jogo do Snake

A figura 24 representa a matriz *desenho*, onde o valor 1 são as paredes, 0 são os caminhos que podem ser percorridos, 5 é a cabeça da cobra, 4 é o corpo da cobra e 3 é a maça (objetivo do agente).

6. CONCLUSÃO

Através de pesquisas relacionadas à área da Inteligência Artificial com foco no estudo de Agentes Inteligentes em buscas heurísticas, ainda em paralelo o estudo e pesquisa referentes às condições de ambiente relacionadas ao problema proposto, foi possível a elaboração de um algoritmo eficiente, capaz de percorrer somente os estados necessários para se obter o resultado esperado da melhor forma possível.

O objetivo do trabalho, inicialmente era apenas realizar a programação de um algoritmo capaz de controlar um agente inteligente que pudesse, através de uma técnica heurística, encontrar o melhor caminho para a saída de um labirinto. No começo do projeto teve-se grande dificuldade com o entendimento do funcionamento das técnicas heurísticas, mas através de estudos em livros, pesquisas na internet e auxílio dos professores, foi possível atingir o objetivo proposto para o projeto.

Em síntese, neste projeto não apenas se teve sucesso em programar uma técnica heurística, mas ainda da opção ao usuário de definir os dados para a busca e escolher entre duas técnicas muito conhecidas. Com a programação destas técnicas, foi possível ainda neste projeto, programar 2 jogos que podem ser considerados um exemplo simples para o auxílio na iniciação ao estudo na área de jogos, que torna-se cada vez mais importante no mundo acadêmico.

Através dos exemplos demonstrados no projeto, pode-se chegar às conclusões de que o algoritmo de busca Gulosa é melhor aplicado quando não se precisa obter o melhor caminho, mas que queira economizar tempo e memória. Enquanto o algoritmo de A* é melhor aplicado quando não se tem preocupação quanto ao tempo e gasto de memória para se encontrar a solução, e há necessidade de se obter sempre o melhor caminho.

7. REFERENCIAS

ALBERTO, Luiz; **Representação do conhecimento**; s.d.; Disponível em: <http://www.professorluizalberto.com.br/prof/images/stories/Documentos/FAPEN/2013-1/Aula_07_-_IA_-_Representao_do_Conhecimento.pdf> Acesso em 15 de Março de 2013.

BARANAUKAS, José A; **Estratégias de Busca Informada**; Departamento de Computação e Matemática FFCLRP – USP; 2009; Disponível em: <http://dcm.ffclrp.usp.br/~augusto/ia/IA-Busca-Informada.pdf> Acesso em 20 de Março de 2013.

BRAGA, Bruno T. R., PEREIRA, José L. A.; **Agentes Inteligentes – Conceitos Características e Aplicações**; Universidade da Amazônia; Belém – PA; 2001; Disponível em: <http://www.nead.unama.br/site/bibdigital/monografias/agentes_inteligentes.pdf > Acesso em 20 de Abril de 2013.

CÂMARA. Marco S. A. L.; **Inteligência Artificial: Representação do Conhecimento**; Universidade de Coimbra; s.d.; Disponível em: <http://student.dei.uc.pt/~mcamara/artigos/inteligencia_artificial.pdf> Acesso em 25 de Abril de 2013.

CARD, S., Moran, T. & Newell, A. (1983). The Psychology of Human-Computer Interaction. Hillsdale, NJ: Erlbaum.

Ciriaco, Douglas; **O que é Inteligência Artificial?**; 25 de Novembro de 2008; Disponível em: <http://www.tecmundo.com.br/intel/1039-o-que-e-inteligencia-artificial-.htm> Acesso em 20 de Julho de 2013.

ComputerWorld, **Mercado brasileiro de T.I. cresce 10,8% em 2012**, 11 de abril de 2013, Disponível em: <http://computerworld.uol.com.br/negocios/2013/04/11/mercado-brasileiro-de-ti-cresce-10-8-em-2012-segundo-idc/> Acesso em 15 de maio de 2013.

Correio Braziliense, **Mercado de criação de jogos digitais tem potencial de crescimento no país**; 08 de maio de 2011, Disponível em: <http://www.correiobraziliense.com.br/app/noticia/economia/2011/05/08/internas_economia,251281/mercado-de-criacao-de-jogos-digitais-tem-potencial-de-crescimento-no-pais.shtml> Acesso em 26 de Agosto de 2013.

FERRARI, Fabricio; **Introdução à Inteligência Artificial**; 2005; Disponível em: <http://www.ferrari.pro.br/home/documents/FFerrari-Introducao-IA.pdf> Acesso em 05 de Março de 2013.

MENDONÇA, Vinicius G.; Java no desenvolvimento de jogos; 2009; Disponivel em: <http://pontov.com.br/site/java/47-javageral/87-java-no-desenvolvimento-de-jogos> Acesso em: 08 de dezembro de 2012.

ORRICO, Alexandre. Mercado brasileiro de games já é o quarto maior do mundo e deve continuar a crescer, Folha de S. Paulo. 08 de Outubro de 2012. Disponivel em: < http://www1.folha.uol.com.br/tec/1165034-mercado-brasileiro-de-games-ja-e-o-quarto-maior-do-mundo-e-deve-continuar-a-crescer.shtml> Acesso em: 26 de Maio de 2013.

PINHEIRO, Josiane M., SILVA, Sérgio R. R.; **Agentes Inteligentes**; s.d.; Disponível em: <http://www.din.uem.br/~jmpinhei/SI/07%20Agentes.pdf> Acesso em 25 de Julho de 2013.

Rede Globo Universidade; **Área da inteligência artificial tenta tornar os robôs mais sociáveis**; 11 de Maio de 2013; Disponível em: <http://redeglobo.globo.com/globouniversidade/noticia/2013/05/area-da-inteligencia-artificial-tenta-tornar-os-robos-mais-sociaveis.html> Acesso em 22 de Agosto de 2013.

RUSSELL, Stuart J.; NORVIG, Peter; **Inteligência artificial**. Rio de Janeiro, Elsevier, 1021p. 2004.

SILVA, Luciano V. D.; **Conceitos e Métodos de Representação do Conhecimento**; ESAB; Vila Velha – ES 2010; Disponível em:

<http://www.esab.edu.br/arquivos/monografias/luciano-vieira-da-silva.pdf> Acesso em 05 de Abril de 2013.

SIMÕES, Alexandre S.; **IAC - Inteligência Artificial Aplicada a Controle**; s.d.; Disponível em: <http://www.gasi.sorocaba.unesp.br/assimoes/lectures/iac/downloads/ia_aula01.pdf> Acesso em 15 de Março de 2013.

VILICIC, F.; CAPUTO, V.; CARNEIRO, H.; **A primavera da Inteligência Artificial**; VEJA 09 de Março de 2013; Disponível em: <http://veja.abril.com.br/blog/ricardo-setti/tema-livre/a-primavera-da-inteligencia-artificial/> Acesso em 20 de Julho de 2013.

ZUBEN, Fernando J. Von; **Estruturas e Estratégias de Busca**; Unicamp; s.d.; Disponível em: <ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ea072_2s06/notas_de_aula/topicoP2.4_06.pdf> Acesso em 20 de Março de 2013.