UNIVERSIDADE DE SOROCABA
PRÓ-REITORIA ACADÊMICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

Danilo de Lucas Moraes Dias

APRENDIZADO DE MÁQUINA APLICADO Á RESOLUÇÃO DE PROBLEMAS DE DECISÃO

Sorocaba
2016

Danilo de Lucas Moraes Dias

APRENDIZADO DE MÁQUINA APLICADO Á RESOLUÇÃO DE PROBLEMAS DE DECISÃO

Trabalho de conclusão do curso de graduação apresentado na Universidade de Sorocaba como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Fernando Cesar Miranda

Sorocaba/SP

2016

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

IA	Inteligência Artificial
NEAT	Evolving Neural Networks through Augmenting Topologies
SMW	Super Mario World
RNA	Redes Neurais Artificiais
SNES	Super Nintendo Entertainment System
TAS	Tool Assisted Speedrun

SUMÁRIO

1. INTRODUÇÃO 5

2. ESTADO DA ARTE 8

3. O PROJETO 9

3.1. Snes9x – Emulador 9

3.2. TAS – *Plugin* 9

3.3. Lua – Linguagem 10

4. CONSIDERAÇÕES FINAIS 12

1. INTRODUÇÃO

Segundo John McCarthy (2007), “Inteligência Artificial é a ciência e a engenharia de fazer maquinas e especialmente programas de computador inteligentes”. Nos jogos, onde as IAs começaram com tarefas simples, como movimentação de inimigos, hoje podem desenvolver tarefas complexas, como adequar as fases e dificuldades com base na habilidade do jogador.

Existem alguns problemas que são muito difíceis, ou mesmo inviáveis, de se resolver com métodos simples de busca, como por exemplo, dizer se uma fase de um jogo de plataforma é possível ou não de ser passada. Segundo Turing, quando se tem um problema de decisão onde não é possível construir um algoritmo que para uma infinidade de entradas não haja sempre uma saída do tipo sim ou não, o problema é caracterizado como indecidível.

O objetivo deste trabalho é apresentar um algoritmo que será capaz de levar o personagem “Mario” do jogo “Super Mario World” até o final de fases propostas no decorrer dos testes, o intuito do programa é que ele “aprenda” a cada nova fase passada e que com isso seja capaz de passar cada vez mais rápido cada nova fase nunca antes concluída.

Alguns jogos dão a liberdade do jogador criar a própria fase, como é o caso do jogo “Super Mario Maker”, onde o jogador pode criar fases e submete-las online para outras pessoas jogarem, porém, neste tipo de jogo, os jogadores se deparam com uma situação onde são forçados a passar da própria fase para “provar” que ela é possível de ser concluída. Isso pode ser um infortúnio para muitos jogadores que tem a habilidade de criar fases complexas e muito trabalhadas, mas não tem a habilidade, ou mesmo a persistência necessária para passar as próprias fases. Isso pode ser resolvido com um algoritmo que seja capaz de jogar as fases que os jogadores desenvolveram e dizer se ela é possível ou não de ser passada. Existem infinitos casos e situações que podem ser criados em jogos que dão este tipo de liberdade ao jogador, por tanto, é preciso que o algoritmo seja capaz de resolver situações difíceis, ser capaz de aprender com novas situações passadas e dar a liberdade de os desenvolvedores poderem criar soluções para situações que levariam muito tempo para o algoritmo resolver.

Por se tratar de um algoritmo de *Machine Learning*, ele pode ser executado por tempo indefinido e “aprender” de forma autônoma. O algoritmo será capaz de resolver uma gama de situações facilmente. Ao se deparar com uma situação nunca

antes vista, ele irá, por meio de uma tabela de possibilidades ponderada pelo algoritmo, executar ações até que encontre uma solução para a situação. Todas as situações que são resolvidas são armazenadas e podem ser generalizadas para outras situações.

Jogos que permitem que o jogador edite suas próprias fases podem utilizar o algoritmo em diversos modelos de uso. A execução do algoritmo pode ser definida como a forma principal de julgamento da fase, garantindo que toda fase editada é possível de ser concluída, ou utilizando-o em um modelo mais comercial, onde o jogador precise gastar pontos para utilizar um "julgamento automático", podendo cobrar ou não por isso.

A solução apresentada neste trabalho não se aplica exclusivamente a jogos, o algoritmo pode ser utilizado para outras finalidades, como em mecanismos de busca, ou implementações de sistemas que requerem tomadas de decisão. Por se tratar de um algoritmo que utiliza de algumas técnicas e conceitos de Inteligência Artificial em conjunto, seus resultados podem ser utilizados como base para futuros projetos que tenham interesse na utilização dos métodos nele descritos.

As limitações são referentes a plataforma que é necessária para ter acesso aos inputs no SMW, se trata de um plug-in para emuladores de SNES que libera acesso para *scripting* na linguagem de programação Lua. Algumas coisas não são conseguidas facilmente via *plugin*, sendo necessário que seja feita uma varredura na memória para ter acesso a alguns valores no decorrer da execução do algoritmo. Os scripts aceitos pelo *plugin* são escritos de forma estrutural, o que os tornam grandes e, como consequência, torna o processo de abstração do código mais trabalhoso.

Para a criação do código fonte do algoritmo será utilizada a linguagem de programação Lua. Para a execução do jogo SMW será utilizado o emulador Snes9x em conjunto com um plug-in denominado TAS para tornar possível a execução de scripts que permitem a manipulação dos inputs e valores de endereços de memória utilizados pelo jogo.

O algoritmo é executado pelo TAS, ele faz a injeção de comandos no emulador que refletem no jogo. Por meio de códigos hexadecimais, são acessados endereços de memória para ter acesso as variáveis utilizadas pelo jogo, como variáveis de posicionamento e *flags* de estado. Com estes *inputs* é possível saber qual é o estado atual do jogo e assim criar as reações que devem ser executadas para

resolver as situações. Foram utilizados conceitos de *Machine Learning* para o armazenamento de situações solucionadas e foi implementado um algoritmo de generalização de situações para utilizar os resultados de uma solução em problemas semelhantes, o que torna o algoritmo de certa forma “inteligente”. Isso também resolve os problemas de excesso de treinamento e crescimento desnecessário da base de dados.

Para o desenvolvimento do projeto foram utilizados somente *softwares* de código aberto.

Atividades	Início	Fim
Pesquisa	**01/03/2016**	**30/04/2016**
Machine Learning		
Programação Genética		
Inteligência Artificial		
Algoritmos Evolutivos		
Documentação	**01/05/2016**	**30/06/2016**
Pesquisa Bibliográfica, Estado da Arte, Problema e Hipótese		
Objetivos, Problema, Benefícios, Público-alvo, Justificativa, Limitações		
Material, Métodos, Custos, Cronograma		
Conclusão - Finalização do texto parcial		
Slides para apresentação		
Desenvolvimento	**15/07/2016**	**15/11/2016**
Escolha das tecnologias		
Controle de versionamento		
Configuração de ambiente		
Desenvolvimento teorico do algoritmo		
Desenvolvimento tecnico do algoritmo		
Desenvolvimento da interface de usuario		
Possiveis refatorações e otimizações no codigo		
Criação de bando de dados		
Teste		

2. ESTADO DA ARTE

Existem algoritmos de Inteligência Artificial desenvolvidos para resolução de problemas indecidíveis, de forma a encontrar soluções heurísticas para tais problemas. O conceito de programação genética é muito utilizado nestes algoritmos, utilizando-se da ideia de “Seleção Natural” de forma a desenvolver e evoluir uma solução para o problema. Segundo Darwin (1859), um indivíduo que obtém características que o torna mais apto que outros no ambiente em que se encontra, poderá passar seus genes mais facilmente as próximas gerações e com isso tornar seus descendentes mais “evoluídos” a situação, o que é justamente o conceito utilizado em algoritmos genéticos.

Algumas empresas têm como objetivo principal a otimização de seus algoritmos de busca, como é o caso do Google, que atualmente tem posse do melhor mecanismo de busca da atualidade, de mesmo da empresa.

Existem algumas competições onde os participantes devem criar algoritmos que sejam capazes de jogar jogos, o competidor que criar um algoritmo que complete a fase em menor tempo ganha. Essas competições geram soluções muito especificas e pouco genéricas para situações diferentes das propostas pelos organizadores da competição.

Pode ser encontrado no link https://www.youtube.com/watch?v=qv6UVOQ0F44 um vídeo onde é demonstrado um algoritmo capaz de jogar fases do jogo “Super Mario World” de forma totalmente autônoma. Nele é explicado que foram utilizados conceitos do NEAT para o desenvolvimento do algoritmo e que ele consegue completar as fases sem nenhum conhecimento prévio, sem nem mesmo as ações que o personagem Mario pode realizar, como andar, pular, etc. NEAT utiliza conceitos de Redes Neurais Artificiais e Programação Genética, para fazer a evolução das topologias.

3. O PROJETO

3.1. Snes9x – Emulador

Snes9x é um conhecido emulador para jogos de Super Nintendo. Ele permite algumas facilidades, como a utilização de *plugins* para debug, ou mesmo para a criação de script capazes de modificar conteúdos nos jogos.

Ele é um programa de código aberto, seu código pode ser encontrado no site GitHub, onde pode ser baixado e modificado.

O código do emulador pode ser encontrado no link:

https://github.com/snes9xgit/snes9x

Figura 1: Emulador utilizado para rodar o jogo e que permite a utilização do *plugin* TAS

3.2. TAS – *Plugin*

Tas é um *plugin* que permite que o programador utilize scripts para modificar conteúdos no jogo. Ele pode ser utilizado em uma grande variedade de emuladores e apresenta diversas facilidades para o desenvolvimento.

O *plugin* tem uma documentação no site oficial que demostra todos os comandos e funções nativas que podem ser utilizadas nele, e ainda apresenta exemplos para melhor entendimento.

O *plugin* pode ser encontrado no link:

http://tasvideos.org/LuaScripting.html

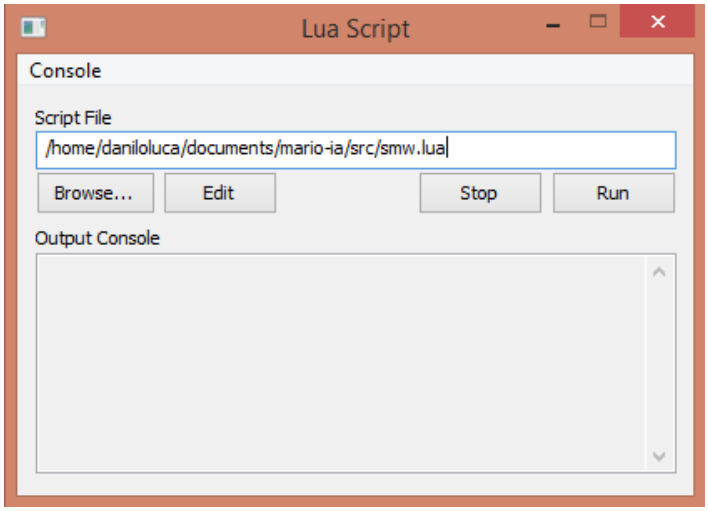

Figura 2: Janela exibida pelo emulador para apresentação de *outputs* do *plugin* TAS.

3.3. Lua – Linguagem

Lua é uma linguagem de programação de *script* de multiparadigma, é muito utilizada para a criação de *scripts* para jogos, pela sua flexibilidade e leveza. O que a torna extremamente poderosa é a sua excelente construção de descrição de dados, baseada em tabelas associativas e semântica extensível.

A linguagem foi utilizada para o desenvolvimento do *script* referente ao algoritmo de *machine learning*. Foram utilizadas as boas práticas de programação descritas na documentação da linguagem.

O código foi escrito de formar modularizada por meio de funções para melhor manutenção e escalabilidade. Isso possibilitará a migração do algoritmo para outros jogos de forma mais simples, além de isolar possíveis *bugs*.

```lua
function getBlocks()
    local blocks = {};
    local size = 160;

    for m16_y=-size, size, 16 do
        for m16_x=-size, size, 16 do
            local game_x, game_y, tile = getTile(m16_x, m16_y);

            -- debugger(game_x, game_y, tile);

            -- Green ground
            if block.semi[tile] then
                local screen_x, screen_y = screenCoordinates(game_x, game_y, s16(camera.x), s16(camera.y));
                drawBlock(screen_x, screen_y, 15, 15, "green");
            end

            if block.solid[tile] then
                local screen_x, screen_y = screenCoordinates(game_x, game_y, s16(camera.x), s16(camera.y));
                drawBlock(screen_x, screen_y, 15, 15, "red");

                local b = {
                    x = game_x,
                    y = game_y,
                    st = 0,
                    num = tile
                };

                table.insert(blocks, b);
            end
        end
    end

    return blocks;
end
```

Figura 3: Trecho de código escrito na linguagem de programação Lua.

4. CONSIDERAÇÕES FINAIS

O algoritmo já se apresenta em um estado funcional. Já é possível executá-lo por um tempo indefinido e observar sua evolução no decorrer das reinicializações. As soluções encontradas para os problemas passados já são armazenadas e reutilizadas para problemas semelhantes, porem sua busca por soluções ainda não se apresenta muito eficiente, podendo levar muito tempo para chegar a um resultado válido.

Para otimizar a busca por soluções do algoritmo, será implementado um *script* utilizando os princípios de programação genética, que o permitirá buscar soluções de forma mais eficiente e rápida. Uma heurística simples é necessária para determinar qual é a ordem com que o algoritmo irá executar as ações e assim fazer a “tentativa e erro” de forma mais assertiva. Futuramente será preciso realizar a implementação de uma base de dados mais robusta, pois com o decorrer do tempo, e com a variedade de situações apresentadas ao algoritmo, a base ficará muito extensa, fazendo com que o método atual de busca por situações solucionadas seja mais lento que a execução do próprio algoritmo para encontrar uma solução.

Com as pesquisas e implementações de diversos tipos de algoritmos relacionados a Inteligência Artificial, foi possível adquirir um bom conhecimento sobre o assunto, mais especificamente na subárea de Inteligência Artificial dita *machine learning.* As pesquisas relacionadas ao estado da arte do projeto ajudaram a ter uma boa visão de como se encontra o estado atual dos projetos que foram e que vem sendo desenvolvidos, se tratando de Inteligência Artificial. Com isso foram desenvolvidas algumas ideias inspiradas em outros projetos como os citados no Estado da Arte.

Como o algoritmo já se encontra em um estado relativamente aceitável, cerca de quatro a cinco meses devem ser suficientes para concluir o projeto. Sempre é possível evoluir e otimizar algoritmos do tipo. O projeto apresenta uma boa margem para continuação em futuros projeto, ou mesmo como base para novos projetos.

REFERÊNCIAS BIBLIOGRÁFICA

DARWIN, Charles. **On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life.** 1859.

MCCARTHY, John. **What is artificial intelligence?** Stanford University, 2007.

RICH, Elaine.; KNIGHT, Kevin.; NAIR, Shivashankar B. **Artificial Intelligence.** 3. ed. Tata McGraw-Hill, 2009.

STANLEY, Kenneth O.; MIIKKULAINEN, Risto. **Evolving Neural Networks through Augmenting Topologies.** Massachusetts Institute of Technology, 2002.

TURING, A. M. **On Computable Numbers, With An Application To The Entscheidungsproblem.** The Graduate College, Princeton University, New Jersey, U.S.A, 1936.

ANEXOS