ANALISE DE ALGORÍTIMO GENÉTICO USANDO TÉCNOLOGIA WEBASSEMBLY PARA SOLUÇÃO DO CUBO DE RUBIK

Luiz Fernando Brogliatto Ferreira, Marcelo Preti Braga

Centro Universitário Assis Gurgacz (FAG)

Av. Min. Cirne Lima, 2565 - Jardim Coopagro, CEP 85903-590 – Toledo – PR – Brazil

Sistemas de Informação

luiz0067@gmail.com, marcelo.braga@fag.edu.br

**Abstract.** The present study has the purpose of adapting a solution for the Rubik's cube with the method of genetic algorithms and using the WebAssembly technology. The focus of the research is to carry out tests of a solution that can contemplate wide compatibility between different architectures and operating systems without using resources such as external servers, plugins, extensions etc. The concept proposed is to use only internet browsers that have support for WebAssembly technology and in this concept the prototype will be running directly on the computer or device with only resources from the local machine. In addition to having broad compatibility, another characteristic of the WebAssembly technology to be observed in the study is the difference in performance in relation to the Java Script technology. The tests in this study serve as the basis for future research.

Key Words: Rubik Cube, Genetic Algorithms, WebAssembly.

**Resumo.** O presente estudo tem o propósito de adaptar uma solução para o cubo de Rubik com o método de algoritmos genéticos e utilizando a tecnologia WebAssembly. O foco da pesquisa concentra em realizar testes de uma solução que possa contemplar ampla compatibilidade entre diversas arquiteturas e sistemas operacionais sem uso de recursos como servidores externos, plugins, extensões etc. O conceito é proposto é usar apenas navegadores de internet que possuam suporte para a tecnologia WebAssembly e neste conceito o protótipo será rodando diretamente no computador ou dispositivo contanto com somente recursos da máquina local. Além de possuir ampla compatibilidade outra característica da tecnologia WebAssembly a ser observada no estudo é a diferença de desempenho em relação a tecnologia Java Script. Os testes deste estudo servem de base a pesquisas futura.

Palavras Chave: Cubo de Rubik, Algorítmico genéticos, WebAssembly

Introdução

Segundo Flanagan e Ferguson (2002) *javascript* é uma linguagem de programação que também roda em navegadores de internet. O objetivo deste estudo é adaptar uma solução com o método de algoritmo genético, que opere em uma ampla gama de arquiteturas de hardware e sistemas operacionais, com desempenho superior ao *javascript*, e nesta solução deve assim ser capaz de resolver o cubo de Rubik.

Para Esteves (2008) o cubo de Rubik, também conhecido como cubo mágico e Saeidi (2018) define que o cubo mágico é um jogo de quebra cabeça tridimensional que tem um grau de complexidade que dificulta sua solução de forma aleatória. As características de complexidade na solução do cubo foram abordadas por Saeidi (2018) e Esteves (2008) com as técnicas de algoritmos genéticos. Pontos importantes devem ser considerados um deles é que existem uma tendência crescente das aplicações que tem tenham o formato WEB que é compatível com navegadores de *internet* conforme Duraes(2008). Outro ponto é que essas aplicações WEB tem um desempenho inferior a aplicações em formato nativo assim descreve Rourke (2018).

Duraes(2008) afirma que o longo dos anos, a popularização da Internet proporcionou a migração das aplicações que antes eram restritas ao ambiente *desktop* para um modelo distribuído devido às diversas facilidades oferecidas pelo ambiente web. A construção de páginas para web tornou-se popular com o padrão HTML (*HyperText Markup Language*). Com o crescimento da web, novas tecnologias foram aparecendo focadas na construção de páginas dinâmicas. Além disso Duraes (2008) relata que há estimativas de que em 2010 existam mais celulares no planeta do que seres humanos, e a cada dia mais e mais pessoas têm acesso a sistemas corporativos através de outros dispositivos além do computador pessoal em uma mesa. Isso abre uma série de novas oportunidades e desafios na criação de software. Cada vez mais temos de pensar que nossos usuários irão acessar nossas soluções não apenas em um computador pessoal, usando um browser desktop, mas também de uma gama de dispositivos e browsers diferentes.

Apesar da vantagem de performance de aplicações nativas em relação aplicações para a *internet com* cada aplicação nativa tem que ser compilada novamente para plataforma de destino. Isso envolve vários tipos de adaptações para cada tipo de arquitetura de hardwares descrito por Telkapalli (2015).

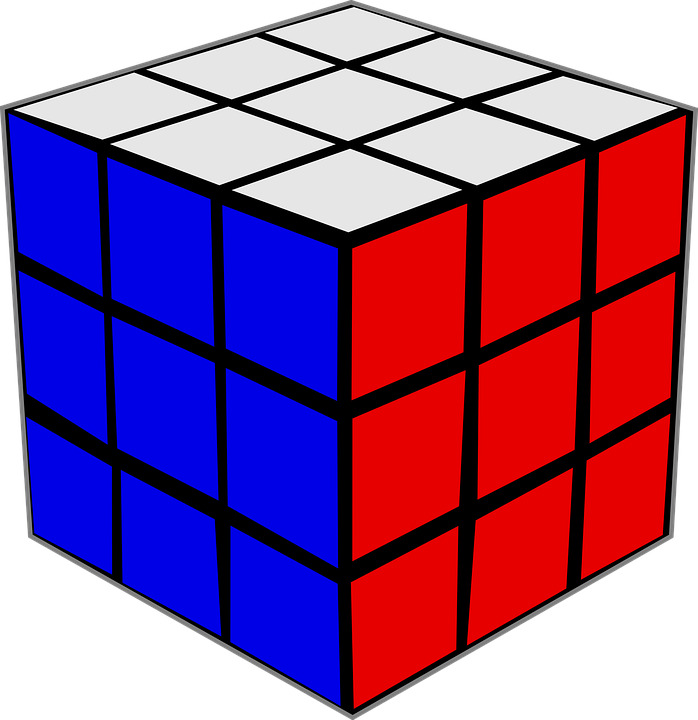
Segundo W3C (2018) a primeira versão documentada do Webassembly foi em março de 2017 a tecnologia propõem a união o que há de melhor dos dois formatos ela promete o desempenho superior das aplicações em *javascript* em sistemas em formato para internet Battagline (2019) confirma que com WebAssembly há grande compatibilidade entre sistema e hardwares além de funciona na maioria dos navegadores modernos, por estes motivos será esta tecnologia é empregada neste estudo.

Revisão de Literatura

* 1. **Cubo de Rubik**

Segundo Clifford (2009) o cubo de Rubik foi inventado pelo húngaro Ernő Rubik em 1974, foi patenteado em 1975 surgiu no mercado húngaro em 1977, em 1982, cerca de 10 milhões de cubos tinham sido vendidos na Hungria, mais do que a população do país. Estima-se que mais de 100 milhões foram vendidos em todo o mundo. Para Clifford (2009) o cubo é uma matriz 3 x 3 x 3 de cubos menores que são coloridos de tal forma que as seis faces do cubo grande têm cores distintas. Clifford Considera o cubo tem 26 subcubos externos são articulados internamente para que essas faces possam ser giradas sobre si. O objetivo do quebra-cabeça é retornar um cubo embaralhado a um estado em que cada face tenha uma única cor.

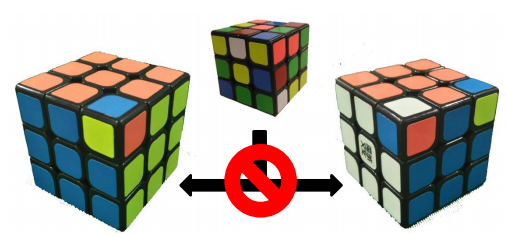
**Figura 1** – cubo de rubik



Fonte: pixabay (2020)

Oliveira e Souza (2016) acrescenta que o cubo de Rubik pode ter 43.252.003.274.489.856 combinações possíveis. Rubikaz (2013) descreve o cálculo desde grande número de possiblidades da seguinte forma começando pela permutação dos 8 vértices multiplicado pela permutação das 12 arestas deste resultado obtido apenas metade das possibilidades acima são verdadeiras, uma vez que não é possível permutar duas arestas sem trocar também a posição de dois vértices, e vice-versa. E Oliveira e Souza (2016) completa que existem 8 cantos que se movem em 3 posições diferentes alternando entre as 3 cores exceto os casos representados na figura 2. Com isso, apenas 1÷3 das 38 são possíveis, sendo 38 ÷ 3 assim resultando a 37.

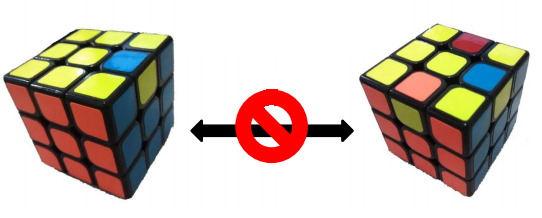
**Figura 2** - Caso impossíveis dos cantos



Fonte: Oliveira e Souza (2016)

Oliveira e Souza (2016) descreve que semelhante aos cantos os a permutação dos meios é transportada aos pares, não sendo pos meios que podem 12 posições alternado entre suas 2 cores exceto os casos apresentados na figura 3 totalizando 212 ÷2 assim resultando 2

**Figura 3** - Caso impossíveis dos meios



Fonte: Oliveira e Souza (2016)

Multiplicando todas as permutações das arestas, vértices, cantos, meios dividido pelas posições impossíveis e também metade das possibilidades acima são verdadeiras a equação se resolve conforme representa a figura 4.

**Figura 4** – Cálculo de combinações possíveis do cubo de Rubik



Fonte: Elaboração dos autores (2020)

Apesar do grande número de possibilidades descritas por Rubikaz (2013) e Oliveira e Souza (2016) um grupo pesquisadores Tomas Rokicki, Herbert Kociemba, Dr. Morley Davidson, e John Dethridge da Universidade Kent, nos Estados Unidos conclui que é possível resolver qualquer combinação do quebra-cabeças conhecido como “cubo mágico” em apenas 20 movimentos ou menos (Rokicki, Et Al, 2014).

Rubikaz (2013) calcula que se fossem executadas cada umas das possibilidades em 1 segundo demoraria 1400 trilhões de anos, supondo que nunca repetisse a mesma combinação. Agora após detalhar o cálculo de todas as possibilidades será descrito a nomenclatura das faces e movimentos do cubo de Rubik segundo Bomberger (2017) no cubo cada face é representada pela letra inicial de seu nome em inglês conforme ilustra a tabela1:

**Tabela 1** – Siglas das faces

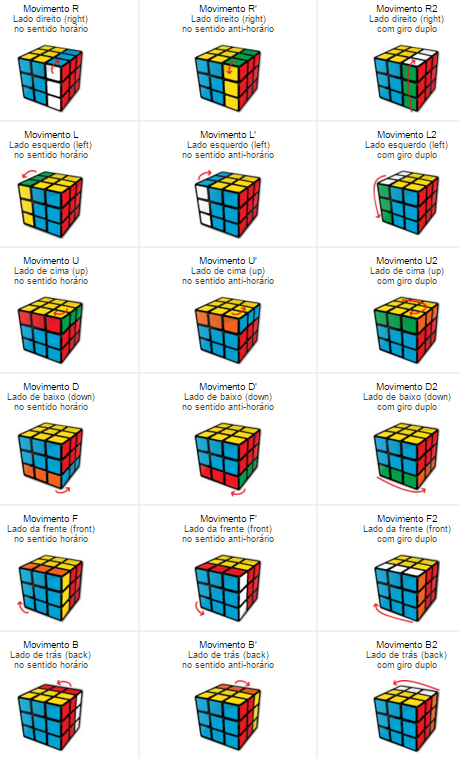
|  |  |  |
| --- | --- | --- |
| **F** front (frete) Verde | **B** Back (atrás) Azul | **D** Down (baixo) Amarela |
| **R** Right (direita) laranja | **L** Left (esquerda) Vermelho | **U** Up topo Branca |

Fonte: Bomberger (2017)

De acordo com Bomberger (2017) pode-se entender que os movimentos de cada face variam entre em sentido horário e anti-horário em um ângulo de 90º de giro. A notação de cada movimento é feita pela sigla da face no sentido horário, e da mesma forma apenas acrescentando o símbolo ‘ no final da sigla para diferenciar caso for sentido anti-horário. Além do giro duplo de 180º identificado pelo número 2 inserido logo após a sigla da face exemplo conforme a figura 5 abaixo na próxima página.

Omelyan (2018) desenvolveu um simulador do cubo mágico 3d em formato web utilizado HTML, CSS (linguagem de marcação de estilos) e *javascript*, este simulador é capaz realizar todos os movimentos explicados na figura 5 este simulador também permite rotacionar o cubo para ver todas as faces.

**Figura 5** – notações usadas para os movimentos no cubo

.

Fonte: Rafael e Lucas, (2017)

Segundo Falcão e Oliveira (2010) a repetição consecutiva dos movimentos na mesma face pode sempre ser condensados em um movimento simples exemplo F2F=F’ e este outro exemplo FRR’D=FD até anula os movimentos da face R, assim como esses o exemplo é o giro duplo descrito por Bomberger (2017), Rafael e Lucas (2017) FF=F2.

Os movimentos já mencionados são usados no método Fridrich, este método foi desenvolvido por Jessica Fridrich professora de engenharia elétrica da universidade Binghamton, enquanto tinha apenas 17 anos em abril 1982 ganhou o seu primeiro campeonato da Tchecoslováquia resolvendo o cubo em 23.55 segundos. Além disso em sua melhor forma Fridrich afirma que foi capaz de ter resolvido o cubo em 17 segundos usando seu método. Segundo Fridrich (1995) o seu algoritmo permite qualquer pessoa resolver o cubo em até 56 movimentos. Apesar de ter surgido em 1995 mesmo depois de 15 anos esse método ainda é utilizado pelos recordistas mundiais. Contudo após a invenção do método Fridrich (1995) segundo Toshniwal Golhar (2019) devido que o cubo tem uma grande complexidade estima-se que apenas 5,8% da população mundial possa encontrar a solução.

O Método Fridrich(1995) é formado por 4 etapas sequenciais onde começa formando uma cruz, sege uma face branca, depois inicia-se a construção das outras faces e termina com todas as cores no lugar conforme ilustra a tabela 2:

**Tabela 2** – Método Fridrich

|  |  |  |  |
| --- | --- | --- | --- |
| Descrição das ações | N° médio de movimentos | Tempo | Resultado |
| Coloque as quatro arestas da primeira camada formando uma cruz. | 7 | 2s |  |
| Coloque quatro blocos, cada um consistindo de um canto da primeira camada e um correspondente borda da segunda camada. | 4 x 7 | 4 x 2s |  |
| Orientar simultaneamente os cantos E bordas para que a última camada tenha a cor necessária (um algoritmo em 40). | 9 | 3s |  |
| Simultaneamente permute os 8 cubos no  última camada sem girar os cantos ou virar arestas (um algoritmo em 13). | 12 | 4s |  |
| TOTAL | 56 | 17s |  |

Fonte: Fridrich (1995)

* 1. Assembly

Tanenbaum (2006) define Assembly como: Nossa tradução (Uma linguagem *assembly* é uma linguagem pura na qual cada instrução produz exatamente uma instrução de máquina. Em outras palavras, há uma correspondência individual entre instruções e instruções da máquina no programa de montagem.)

A pure assembly language is a language in which each statement produces exactly one machine instruction. In other words, there is a one-to-one correspondence between machine instructions and statements in the assembly program (Tanenbaum, 2006, p. 508)

A linguagem de máquina descrita por Tanenbaum (2006) Segundo Battagline (2019) sucedeu a outro formato binário compilado conhecido como *webassembly. Nesse formato o* texto *webassembly* é um código que acompanha o binário pemitindo ao usuário visualizar as instruções de *bytecode* em uma forma legível por humanos, semelhante à forma como uma linguagem *assembly* permite que você veja quais *opcodes* são executados em um código legível por máquina.

* 1. WebAssembly

Segundo Rourke (2018) *webassembly* não é uma linguagem de programação de alto nível como o javascript, mas um formato binário compilado que todos os principais navegadores são capazes de executar. Para Rourke (2018) webassembly não foi projetado para ser executado diretamente em nenhum hardware apesar ser compilado. Ele se assemelha a conceitos da máquina virtual Java webassembly onde por exemplo um *bytecode* é compilado independentemente da plataforma. R ourke (2018) afirma que WebAssembly é executado no mecanismo JavaScript integrado a cada navegador. O problema com o bytecode JavaScript relatado Rourke (2018) é o requisito para que um plugin seja baixado e instalado no navegador para que o byte code funcione. Rourke (2018) pontua que além de WebAssembly ter sido projetado para ser executado diretamente em um navegador sem plug-in, WebAssembly também tem a intenção de produzir um formato binário compacto que executa com eficiência dentro de um navegador da web. Outro fato destacado por Rourke (2018) é que os desenvolvedores que usam a versão atual do WebAssembly já notaram melhorias de desempenho JavaSript de 10% até 800%

* 1. História da Genética

Segundo Darwin (1859) os seres vivos evoluem por um processo chamado seleção natural, neste processo descrito em sua obra “a origem das espécies” descreve a lei da sobrevivência do mais forte e mais apto as situações do ambiente. Depois de Darwin (1859) em seu estudo Mendel (1865) separou plantas por características em comum exemplo Verde, amarela, lisa e rugosa ele cruzou diferentes pares. E ao contabilizar as características herdadas nas plantas descentes Mendel encontra um padrão matemático probabilístico 3 para 1 que varia conforme as características dos ancestrais. Ne sse padrão matemático identificam dois tipos de características segundo Mendel (1865).

Dominantes: Representadas por letra maiúscula A se manifestam mesmo se fizer para com característica recessiva representada por uma letra minúscula a exemplo AA, Aa e aA.

Recessivas: Representadas por letra minúscula a se manifestam somente se fizer para com característica recessiva exemplo: aa.

Mendel (1865) define as características observáveis nos seres vivos são definidas como fenótipo, a pós Mendel (1865) segundo Reynolds (1999) em 1905 Nettie Stevens cientista descobriu que a determinação do sexo, outro fenótipo, era ligada ao cromossomo de origem paterna, Stevens foi sucedida pelas pesquisas de Thomas Hunt Morgan. P ara Shine (2009) os estudos de Morgan sobre as moscas de frutas onde Morgan descobriu os cromossomos e dentro deles encontrou os genes, enfileirados que são os responsáveis pelos traços hereditários.

Depois da descoberta dos cromossomos e os genes Segundo Shine (2009) Morgan descobriu uma mutação que afetava a cor dos olhos da mosca. Ele observou que a herança da mutação agia de forma diferente pelas moscas macho e fêmea.

* 1. Algoritmos Genéticos

O algoritmo genético — AG, uma técnica de Inteligência Artificial utilizada com sucesso na resolução de problemas de otimização e no aprendizado de máquina, foi apresentado de forma sistematizada por Holland (1975). O algoritmo foi inspirado no processo de evolução, adaptação e seleção natural da teoria de Charles Dawin (1859) (2005 Azevedo, pg. 271)

Nossa tradução (Mais importante ainda, o paradigma da "programação genética" é geral e fornece uma única e igualada abordagem para ser usada em vários problemas aparentemente diferentes para várias de áreas.)

Most importantly, the "genetic programming" paradigm is general and provides a single, unified approach to a variety of seemingly different problems in a variety of areas. (Koza,1990)

Segundo Barreto (1997) os algoritmos genéticos são fundamentados em analise probabilista apresentando mais de uma solução para o mesmo problema diferente dos algoritmos tradicionais que são determinísticos com uma solução e projetados para apenas uma exclusiva função. Programação Evolucionária, Estratégias Evolucionárias, Algoritmos Genéticos e Programação Genética. Estes métodos estão sendo utilizados, cada vez mais, pela comunidade de inteligência artificial para obter modelos de inteligência computacional. Banzhaf (1998) acrescenta que a computação evolucionária compreende um conjunto de técnicas de busca e otimização inspiradas na evolução natural das espécies. Desta forma, cria-se uma população de indivíduos que vão reproduzir e competir pela sobrevivência. Os melhores sobrevivem e transferem seus fenótipos descritos por Mendel (1865) a novas gerações.

Assim como no cubo de Rubik já foram utilizadas essas técnicas em jogos, os primeiros jogos utilizando Inteligência Artificial foram experimentos para pesquisa técnica para solução de problemas, como por exemplo, os jogos clássicos, tais como xadrez, Go e damas(2005 Azevedo, pg. 247)

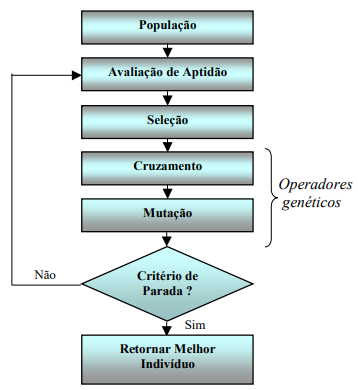
Segundo Silva (2006) o conceito básico da mecânica dos algoritmos genéticos é a de gerenciar as possíveis soluções do problema como "indivíduos" de uma "população", que irá "evoluir" a cada iteração ou "geração". O indivíduo representa uma solução em algoritmos genéticos ele é um mero portador de código genético, o indivíduo serve para selecionado ou seus genes na próxima geração. Isso segue o mesmo conceito da descoberta biológica de Morgan dos genes descrito por Shine (2009).

Silva (2006) define que a população é o conjunto destes indivíduos. O funcionamento dos algoritmos genéticos compreende selecionar o melhor grupo de “indivíduos” presentes uma “população” que irão evoluir a cada “geração” ou “interação” seguintes passos:

* Gerar a população inicial normalmente é formada por indivíduos criados aleatoriamente.
* Avaliar todos os indivíduos da população com critério determinado pela função que pontua a qualidade do indivíduo (função de aptidão ou "fitness").
* Selecionar os indivíduos de melhor valor (dado pela função de aptidão). A seleção serve de base para a criação de um novo conjunto de possíveis soluções, chamado de nova "geração".
* Cruzar os genes dos indivíduos da população gerando novos indivíduos com a carga genética de seus antepassados baseado no parâmetro de porcentagem informado. O "cruzamento" ("crossover") é um operador genético é feita com base no parâmetro de porcentagem informado.
* "Mutação" é obtida aplicando-se sobre os indivíduos alteração das características aleatórias (chamadas "genes") a quantidade de mutações é feita com base no parâmetro de porcentagem informado.
* Geração é a repetição da avaliação, seleção, cruzamento, mutação até encontrar o critério de parada.
* O critério de parada pode ser por tempo de execução do algoritmo, qualidade da solução encontrada, número máximo de gerações, etc.

Os principais componentes mostrados na figura 6 são descritos a seguir em mais detalhes.

**Figura 6** - Estrutura básica de um Algoritmo



Fonte: Silva, (2006)

Segundo Esteves (2008) de forma similar à teoria da evolução de Darwin, a seleção natural restringe a sobrevivência do código genético somente àquele presente nos mais aptos seja porque eles “morrem” antes seja por simplesmente não conseguirem um parceiro. Existem várias técnicas usadas na seleção de indivíduos em algoritmos genéticos como: torneio, truncada, seleção proporcional à aptidão, seleção por ‘rank’. A técnica de seleção por torneio porque para Esteves (2008) é simples e eficaz, nesta técnica são colocados K indivíduos que serão avaliados entre si o melhor será selecionado para a próxima geração. Esteves (2008) considera o algoritmo gerado por Korf (1997) é até hoje aceito como a melhor solução para qualquer estado do cubo, o seu único fator limitativo é o tempo, porque em uma máquina munida de um processador Core 2 Quad Q6600 (com quatro núcleos de processamento), a resolução de cada estado leva, em média, quinze minutos.

Outra solução é apresentada por Saeidi, Shahram (2018) onde seu método baseia-se quantidade geral de todas as faces do cubo 54 subtraído pela quantidade de peças fixas que são 6 dos centros restando 48 peças moveis. Entre essas peças móveis contabiliza-se quantas estão na posição correta. Saeidi, Shahram apresenta um pseudo código RS (Reconhecimento Simulado) com um loop interno em um conceito de temperatura relacionado ao distanciamento de cada peça a sua posição original descrito na figura 6.

**Figura 6** – O Código Pesudo da Proposta de RS

|  |
| --- |
| 1. Begin  2. Initialize Temp;  3. Get input cube, called x;  4. Compute f(x) using equation 3;  5. Repeat until frozen  a. Do N times  i. y := FindNeighbor(x);  ii. Delta := f(y)- f(x);  iii. if Delta≤0 then x:=y ; (Accept y)  iv. else if U(0,1) < e-(Delta)/Temp then x:=y; (Accept y)  v. else reject y;  b. End Do  6. Temp := Temp θ;  7. The Solution will be best so far;  8. End |

(Saeidi,2018)

Metodologia

Para realizar os testes foram necessárias duas etapas a codificação do algoritmo genético e um simulador de movimentos 3d do cubo mágico. Para ligar estas duas etapas foi realizado conforme Rourke (2018) descreve, nesta descrição através da ferramenta online *WasmFiddle* o código escrito na linguagem C é compilado se torna um texto binário, este texto é importado no *javascript* pelo *webassembly.*

O algoritmo genético está escrito na linguagem de programação C, ele tem o objetivo receber uma combinação de movimentos processa-la e retornar uma solução de movimentos onde todas as cores do cubo fiquem iguais. A prototipação do algoritmo genético foi preciso usar as seguintes ferramentas de programa de computador o mingw-64 versão 8.1 que contém o compilador GCC para Windows, a ide *Codeblocks* versão 20.30, todos os testes foram realizados no sistema operacional Windows 10 professional compilação 19042.630 64 bits.

Já o simulador feito por Omelyan (2018) foi adaptado para executar os movimentos descritos por Bomberger (2017), Rafael e Lucas (2017) através da escrita da nomenclatura e também foi adicionado botões de visualização do cubo em ângulos predefinidos. Para as adaptações do simulador foi preciso usar as seguintes ferramentas de programa de computador o editor de texto Notepad++ versão 7, o navegador de internet Google Chrome Versão 86.0.4240.193 64 bits.

A programação do algoritmo genético descrito por Silva (2006) neste estudo é elaborada com a técnica de seleção por torneio apontada por Esteves (2008). O método *fitness* foi baseado na solução de Saeidi, Shahram (2018) onde pontuação de cada indivíduo é calculada pela razão entre quantidade de peças na posição correta denominada X sobre 48 que representa o número total de peças móveis na equação X/48.

O operador de mutação necessitou que não apenas troque o cromossomo como antes de garantir que o novo cromossomo não fosse criado o efeito de condensação de movimentos descrito Falcão e Oliveira (2010).

O operador *crossover* seleciona 2 indivíduos e cruza os genes a fim de gerar um filho, esse operador também evita o efeito de condensação onde caso haja o efeito será trocado o movimento anterior ou posterior a ele.

Os paramentos usados no estudo a população inicial do algoritmo testada é de 150 indivíduos, o torneio é de 3 participantes para 1 ganhador, a porcentagem de cruzamento é de 25%, a mutação é de 3%, a pontuação é o critério de parada da solução o valor da pontuação é de 100%, são 100 gerações iniciais;

Resultados e discussões

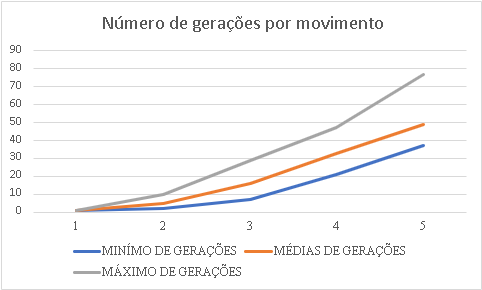
Durante o processo de testes na linguagem C foram feitos inicialmente com sequência de até 5 movimentos para encontrar a solução do cubo de Rubik, os movimentos dos testes foram criados de forma aleatória com até 10 elementos conforme é descrito na tabela 3 e o gráfico da figura 7. Segundo Barreto (1997) o algoritmo genético é probabilístico e não determinístico esse fato reflete no estudo através da diferença de número de gerações para dado o mesmo problema.

**Tabela 3** – Número de gerações por movimento

|  |  |  |  |
| --- | --- | --- | --- |
| **NÚMERO DE MOVIMENTOS** | **MINÍMO DE GERAÇÕES** | **MÉDIAS DE GERAÇÕES** | **MÁXIMO DE GERAÇÕES** |
| 1 | 1 | 1 | 1 |
| 2 | 2 | 4,73 | 10 |
| 3 | 7 | 15,82 | 29 |
| 4 | 21 | 32,45 | 47 |
| 5 | 37 | 49,18 | 77 |

Fonte: Elaboração dos autores (2020)

**Figura 7** – Gráfico do número de gerações por movimento



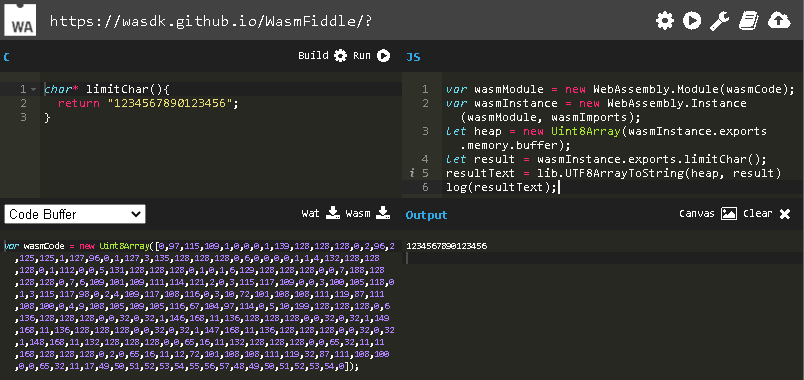
Fonte: Elaboração dos autores (2020)

O efeito de condensação de movimentos descrito por Falcão e Oliveira (2010) é observado nos resultados das soluções encontras, exemplo teste R’ a solução R, ao aumentar o tamanho da sequência de movimentos observa-se basicamente uma das soluções é basicamente o movimento que gera o efeito de condensação na sequencia inversa, exemplo para B2L’TR’ a solução RT’LB2. Outra situação é observada entre faces paralelas como direita e esquerda onde o movimento R2 e R’ for intercedido por L e L2 no exemplo LR2L2R'B2 a solução é B2R'L.

Após estes breves testes iniciou-se a fase de transmitir a sequência de movimentos do sistema de adaptado de Omelyan (2018) para o algoritmo genético escrito na linguagem C. Esta transmissão foi feita com 20 variáveis com tipo inteiro cujo seu valor varia de 0 a 17 correspondia ordem dos 18 movimentos possíveis do cubo de Rubik além das variáveis de paramentos do algoritmo genético. O número de 20 variáveis corresponde o número de Deus que segundo Rokicki, Et Al, (2014) é 20. Além disso a forma de transmitir por várias variáveis inteiras a sequência de movimentos isso ocorre por que apesar da linguagem C suportar variáveis de sequência ou vetorescomo inteiros e caracteres assim como trata textos em sequênciade caracteres *webassembly* não tem esse suporte segundo Rourke (2018).

Rourke (2018) descreve que *webassembly* pode retornar resultados em sequência de caracteres formando um texto, mas durante testes o número máximo obtido foi de 16 caracteres. Ao testar com 17 caracteres a ferramenta WasmFiddle retornava a menssagem de erro “*line 1041: Uncaught ReferenceError: UTF8Decoder is not defined*” A notação dos movimentos ocupa até 2 caracteres isso significa que o *webassembly* pode retornar sequencia composta entre 8 e 16 movimentos. Abaixo os códigos dos testes usados na ferramenta WasmFiddle na figura 8.

**Figura 8** – Código *webassembly* limite de caracteres



Fonte: Elaboração dos autores (2020)

Funções de operação com 2 números inteiros como soma, subtração, adição, e divisão foram integradas com sucesso, mas o algoritmo genético possui mais de 1000 linhas de código e conforme eram adicionadas funções a ferramenta WasmFiddle retornava o seguinte erro: *“Uncaught RangeError: WebAssembly.Compile is disallowed on the main thread, if the buffer size is larger than 4KB. Use WebAssembly.compile, or compile on a worker thread.*” esse erro basicamente indica que o limite de memória de no 4KB foi superado.

Após esse erro não foi possível progredir na integração do algoritmo genético com o *webassembly* por conta deste fato este estudo será dedicado para futuras pesquisas relacionadas e aprimoramento da tecnologia *webassembly.*

Referências

Banzhaf, W; NORDIN, P.; KELLER, R. E. & FRANCONE, F. D. Genetic Programming: an introduction. ISBN 155860510X. Morgan Kaufmann, 1998.

Barreto, J. Inteligência Artificial. No Limiar do Século XXI. ISBN 859003822X. ρρρ Edições, 1997.

Battagline, Rick. (2019) “Hands-On Game Development with WebAssembly: Learn WebAssembly C++: programming by building a retro space game, Packt

disponível em:

https://books.google.com.br/books?id=sfeaDwAAQBAJ&printsec=frontcover&hl=pt-BR#v=onepage&q&f=false Acesso em 07 de setembro de 2020

Bomberger, Chad. (2017) “How to Solve a Rubik's Cube: The Easy Solution to The Rubik's Cube, A Beginner's Guide to Solving This Puzzle, Quick and Easily! (3x3 Cube) (Rubix) (2nd Edition)” , CRB Publishing

Clifford A. Pickover. (2009) “The Math Book: From Pythagoras to the 57th Dimension, 250 Milestones in the History of Mathematics” Sterling Milestones 1ª Edição.

Darwin, Chales Robert.(1859) On the Origin of Species. London: Jhon Murray disponivel em: http://darwin-online.org.uk/converted/pdf/1859\_Origin\_F373.pdf Acesso em 07 de agosto de 2020 disponível em: https://www.up.edu.br/blogs/engenharia-da-computacao/wp-content/uploads/sites/6/2015/06/2006.15.pdf consultado 07 de setembro de 2020

Duraes, Ramon. Desenvolvendo para web usando o Visual Studio 2008: ASP.NET 3.5 Ajax LINQ Segurança Silverlight 2.0 Web Mobile. 1. ed. R. Pardal Mallet, 23 - TijucaRio de Janeiro - RJ, 20270-280: BRASPORT, 2008. ISBN 978-85-7452-393-4. Disponível em: books.google.com.br/books?id=lPyCIn-VsjUC&printsec=frontcover&dq=Desenvolvendo+para+web+usando+o+Visual+Studio+2008+-+P%C3%A1gina+2&hl=pt-BR&sa=X&ved=2ahUKEwi5ocC5w4PsAhW-IrkGHZRoD4cQ6AEwAHoECAAQAg#v=onepage&q=Desenvolvendo%20para%20web%20usando%20o%20Visual%20Studio%202008%20-%20P%C3%A1gina%202&f=false. Acesso em: 25 set. 2020.

Esteves, Bruno Abrantes. Soluções Inteligentes Para O Cubo De Rubik. Orientador: Prof. Dr. Raul Fonseca Neto. 2008. Bacharelado em Ciência da Computação (2008) - Universidade Federal de Juiz de Fora Instituto de Ciências Exatas, Juiz de Fora, MG, 2008. Disponível em: http://monografias.nrc.ice.ufjf.br/tcc-web/exibePdf?id=2. Acesso em: 8 out. 2020.

Falcão, Karla Polyana Silva; Oliveira, Anderson Aleksander da Silva. Simetrias, permutações e grupos no cubo de Rubik. V Bienal da SBM Sociedade Brasileira de Matemática UFPB - Universidade Federal da Paraíba, p. 2-3, 20 out. 2010. Disponível em: silo.tips\_simetrias-permutaoes-e-grupos-no-cubo-de-rubik.pdf. Acesso em: 17 nov. 2020.

Flanagan, David; Ferguson, Paula (2002). JavaScript: The Definitive Guide 4th ed. [S.l.]: O'Reilly & Associates. ISBN 0-596-00048-0 Disponível em http://www.arenahome.org/dir/B%20Per%20imparare%20e%20capire/informatica/musica/JavaScript%20-%20The%20Definitive%20Guide.pdf Acesso em: 15 novembro. 2020.

Fridrich Jessica (1995) “My system for solving Rubik's cube” disponivel em: http://www.ws.binghamton.edu/fridrich/system.html Acesso em 07 de agosto de 2020

KORF, R.E. Finding optimal solutions to Rubik's Cube using pattern databases. 1997. In: AAAI-97 Fourteenth National Conference on Artificial Intelligence. Providence, United States of America, 1997.

Koza, J.R. (1990). Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems, Stanford University Computer Science Department technical report disponível em: http://www.genetic-programming.com/jkpdf/tr1314.pdf Acesso em 07 de agosto de 2020

Mendel, Gregor Johann. (1866). Experiments in Plant Hybridization. disponível em: http://www.esp.org/foundations/genetics/classical/gm-65.pdf Acesso em 07 de agosto de 2020

Mendel, Gregor. (1870). On Hieracium-Hybrids Obtained by Artificial Fertilisation .   
disponível em: http://www.esp.org/foundations/genetics/classical/holdings/m/gm-69.pdf consultado em 07 de agosto de 2020

Oliveira L. P. Christian, Souza P. Fernando (2016) “ANALISE COMBINATÓRIA: UMA APRENDIZAGEM AO CALCULO FINITO DE POSSIBILIDADES DO CUBO DE RUBIKS” disponível em: http://www.unoeste.br/site/enepe/2016/suplementos/area  
/Exactarum/Matem%C3%A1tica/ANALISE%20COMBINAT%C3%93RIA%20UMA%20APRENDIZAGEM%20AO%20CALCULO%20FINITO%20DE%20POSSIBILIDADES%20DO%20CUBO%20DE%20RUBIKS.pdf Acesso em 07 de agosto de 2020

Omelyan, Dmytro “Rubik's cube in Javascript” Kyiv, Ukraine criado em 6 de abril de 2016

atualizado em 9 de agosto de 2018 disponível em: https://codepen.io/Omelyan/pen/BKmedK Acesso em 16 de novembro de 2020

Rafael e Lucas (2017) “Aprenda ler algoritmos no cubo mágico” disponível em:  
https://rafaelelucas.wordpress.com/2017/11/22/aprenda-ler-algoritmos-no-cubo-magico/ Acesso em 02 de setembro de 2020

Reynolds, Moira Davison “American Women Scientists: 23 Inspiring Biographies, 1900-2000”. North Carolina: McFarland & Company, Inc., Publishers

Rokicki T, Kociemba H, Davidson M, Dethridge J. (2014) The diameter of the Rubik’s cube group is twenty. SIAM Rev.;. disponível em: tomas.rokicki.com/rubik20.pdf Acesso em 07 de setembro de 2020

Rubikaz (2013) “Resumiendo, el número de combinaciones del cubo de Rubik son” disponível em: http://www.rubikaz.com/combinaciones-del-cubo Acesso em 07 de setembro de 2020

Saeidi, Shahram. (2018). Solving the Rubik’s Cube using Simulated Annealing and Genetic Algorithm. International Journal of Education and Management Engineering. 8. 10.5815/ijeme.2018.01.01. dsponível em: http://www.mecs-press.org/ijeme/ijeme-v8-n1/IJEME-V8-N1-1.pdf Acesso em: 8 out. 2020.

Shine, Ian;Wrobel, Sylvia (2009) “Thomas Hunt Morgan: Pioneer of Genetics”. Paperback

Silva, Hamon Veloso da; Ferlin, Edson Pedro; Brawerman, Alessandro; Perreto, Mauricio. Interface para Arquitetura Paralela Reconfigurável aplicada a Algoritmos Genéticos usando NIOS II. (2006). Trabalho de Conclusão de Curso (Graduação em Engenharia da Computação) - Centro Universitário Positivo.

Tanenbaum Andrew S. (2006) Structured Computer Organization-5 Ed. - Upper Saddle River, New Jersey 07458 disponível em: http://fuuu.be/polytech/infof 201/bouquin%20 syst%c3%a8mes%20d\_exploitations.pdf Acesso em 07 de agosto de 2020

TELKAPALLI, Murali Krishna. Mutual Interdependency between Compiler and Computer Architecture: An Overview. Indian Journal of Scientific Research, 303, Maharana Pratap Complex, Opp. Kapadia Guest House, B/H V.S.Hospital, Paldi, Ahmedabad, Gujarat 360006, Índia, p. 1, 4 jan. 2015. Disponível em: https://www.researchgate.net/profile/Murali\_krishna\_Telkapalli/publication/283299461\_Mutual\_Interdependency\_between\_Compiler\_and\_Computer\_Architecture\_An\_Overview/links/571dbe7708aee3ddc56acf08/Mutual-Interdependency-between-Compiler-and-Computer-Architecture-An-Overview.pdf?origin=publication\_detail. Acesso em: 25 set. 2020.

Toshniwal E. S. e Golhar Y., (2019) "Rubik’s Cube Solver: A Review," 9th International Conference on Emerging Trends in Engineering and Technology - Signal and Information Processing (ICETET-SIP-19), Nagpur, India, 2019, pp. 1-5, doi:

W3C. WebAssembly Web API: W3C First Public Working Draft, 15 fevereiro 2018. In: WebAssembly Web API. [S. l.], 15 fev. 2019. Disponível em: https://www.w3.org/TR/2018/WD-wasm-web-api-1-20180215/. Acesso em: 25 set. 2020.