Algoritmos Genéticos

Barros. K.W.C , Lima R.N, *Member, IEEE*

Resumo

Este projeto é uma análise do artigo Simulating nature 's methods of evolving the best design solution. Cezary Janikow e Daniel Clair. IEEE. 1995, uma descrição sobre o vídeo da explicação do algoritmo genético e implementação dele para o jogo do dinossauro do Google Chrome, é por fim a descrição da implementação de um algoritmo genético no jogo do Tetris.Essas diferentes abordagens demonstram a aplicação dos algoritmos genéticos em diferentes contextos. O projeto destaca os resultados e conclusões obtidos, bem como o potencial impacto dessas implementações. Detalhes adicionais sobre cada aspecto serão fornecidos no corpo do projeto.

Palavras-chaves: Algoritmo, Algoritmo genético, Jogo

Abstract:

This project provides an analysis of the article "Simulating nature's methods of evolving the best design solution" by Cezary Janikow and Daniel Clair, published in IEEE in 1995. It also includes a description of the video explaining the genetic algorithm and its implementation in the Google Chrome dinosaur game, as well as a description of the genetic algorithm implementation in the Tetris game. These diverse approaches demonstrate the application of genetic algorithms in various contexts. The project highlights the obtained results, conclusions, and the potential impact of these implementations. Further details on each aspect will be provided in the main body of the project.

Keywords: Algorithm, Genetic algorithm, Game.

I. Introdução

Com o avanço da computação e a crescente necessidade de resolver problemas complexos em diferentes áreas, os algoritmos genéticos têm se destacado como uma poderosa abordagem para otimização e solução de desafios. Inspirados no processo de evolução natural, esses algoritmos são especialmente adequados para lidar com problemas de otimização, onde a busca por uma solução ideal pode ser complexa e demandar um grande espaço de busca.
Essa abordagem revolucionária baseia-se em princípios fundamentais, como a codificação de soluções em forma de cromossomos, a utilização de operadores genéticos, como a reprodução, a seleção, o cruzamento e a mutação, e a avaliação da aptidão de cada solução em relação ao problema em questão.

Os algoritmos genéticos possuem uma estrutura flexível e adaptativa, o que os torna aplicáveis a uma ampla gama de domínios. Eles têm sido utilizados com sucesso em áreas como otimização de funções matemáticas, planejamento de rotas, alocação de recursos, aprendizado de máquina e até mesmo na concepção de sistemas complexos.

Ao longo deste artigo/relatório, exploraremos em detalhes os fundamentos dos algoritmos genéticos, suas principais etapas e operadores genéticos. Além disso, serão analisados dois trabalhos. O primeiro é “Simulating nature's methods of evolving the best design solution” de Cezary Janikow e Daniel Clair, que descreve algoritmos genéticos, abordando as estratégias para utilização desse método, incluindo como representar os dados, as operações que podem ser feitas para buscar resultados melhores e as desvantagens dessa abordagem. O segundo é um resumo do vídeo “Inteligência Artificial com Dinossauro da Google” do canal do YouTube do Ivan Seidel que aborda o assunto e fala sobre sua implementação de algoritmo genético e redes neurais para que ele jogue como o dinossauro no jogo disponível no navegador Chrome.
Por fim será descrito uma implementação modificada do jogo Tetris que utiliza algoritmo genético para conseguir o melhor recorde do jogo.

II. Algoritmos Genéticos

Algoritmos Genéticos (AGs) são métodos meta-heurísticos baseados na teoria de seleção natural de Charles Darwin e foram inicialmente propostos por J. H. Holland em 1975. Essa abordagem computacional busca simular o processo evolutivo para resolver problemas de otimização e busca, inspirando-se nos princípios biológicos de hereditariedade, mutação, recombinação e seleção natural.

Pois de acordo com a teoria de C. Darwin, a seleção favorece os indivíduos mais adaptados, que têm maior probabilidade de reprodução e, assim, de transmitir seus códigos genéticos para as gerações futuras. Esses princípios

são incorporados na construção de algoritmos computacionais, como os algoritmos genéticos, que visam encontrar soluções ótimas para problemas específicos, por meio da evolução de populações de soluções representadas por cromossomos artificiais.

Nesse contexto, os cromossomos são estruturas de dados que representam as possíveis soluções em um espaço de busca. Ao longo de um processo evolutivo, os cromossomos são submetidos a avaliação, seleção, recombinação (crossover) e mutação. Após várias iterações, espera-se que a população contenha indivíduos mais aptos.

Estruturas básicas AGs

A Individuo

A representação do indivíduo em um algoritmo genético é a forma como a informação genética é codificada e estruturada para o algoritmo. É comumente utilizada a codificação binária, em que um indivíduo é uma sequência de bits (0 ou 1). Outras representações como codificação real, vetores de recursos ou árvores de expressão também podem ser usadas. A escolha depende do problema e das características dos genes. Uma representação adequada facilita a busca por soluções de qualidade, enquanto uma inadequada pode levar a soluções de baixa qualidade. É fundamental escolher uma representação que capture as características importantes do problema e permita a manipulação e avaliação dos indivíduos da população.

B População

A população em um algoritmo genético é um conjunto de indivíduos representando possíveis soluções para o problema. Ela evolui através de operações genéticas, como seleção, recombinação e mutação. Os indivíduos têm uma representação genética e são avaliados com base em uma função de fitness. Os mais aptos são selecionados para reprodução, combinando suas informações genéticas para gerar novos indivíduos. A mutação introduz pequenas alterações. O objetivo é melhorar a qualidade da população ao longo das gerações até atingir uma condição de parada, como convergência ou um número máximo de gerações.

C Função de avaliação(Fitness)

Uma função de avaliação ou função fitness deve ser capaz de representar os requisitos necessários a que uma população deve se adaptar a fim de avançar para a geração seguinte. Todos os indivíduos da população de cada geração do AG são avaliados por essa função.

É importante que a função fitness seja representativa e possa diferenciar com precisão os indivíduos (soluções) bons dos ruins. Uma função fitness não ajustada na avaliação de indivíduos pode acabar descartando um indivíduo promissor, que poderia ajudar a encontrar soluções melhores para o problema, além do fato de consumir recursos em indivíduos que agregam pouco no desenvolvimento do AG.

Funções fitness multi-objetivas levam em consideração diversos aspectos do problema ao avaliar indivíduos, e podem tratá-los de maneira igualitária ou dando pesos diferentes para cada um.

D Métodos de avaliação

Métodos de seleção são utilizados em algoritmos genéticos para escolher quais indivíduos serão selecionados como pais para gerar descendentes na próxima geração. A seleção é importante para destacar os indivíduos mais aptos e promover a melhoria contínua da população.

Existem diferentes métodos de seleção, incluindo a seleção por roleta, seleção por fitness, seleção por torneio e seleção por ranking. O método mais comum é a seleção por torneio.

Seleção por roleta

Cada indivíduo possui uma área proporcional ao seu valor fitness na roleta. A roleta é girada várias vezes e, a cada giro, o indivíduo selecionado é escolhido como pai para a próxima geração.

Seleção por Fitness

Baseia-se apenas no valor absoluto da função fitness. Indivíduos mais aptos têm maior probabilidade de serem selecionados, o que pode levar à convergência prematura, onde a busca se concentra em uma região específica do espaço de busca.

Seleção por torneio

Envolve a seleção de um número aleatório de indivíduos e a comparação de seus valores fitness. O indivíduo mais apto é selecionado com uma probabilidade determinada por um parâmetro K. Quanto maior o valor de K, maior a pressão seletiva sobre a população.

Seleção por ranking

Classifica os indivíduos com base em sua fitness e atribui probabilidades de seleção de acordo com o ranking, em vez dos valores de fitness. Essa abordagem reduz a pressão seletiva exercida pelos indivíduos mais aptos e é uma estratégia para evitar a convergência prematura.

E Operadores genéticos

Após a seleção de indivíduos, são utilizados dois operadores genéticos, cruzamento e mutação, para gerar a próxima geração do algoritmo genético. O objetivo desses operadores é refinar a busca por meio da combinação de características genéticas dos indivíduos selecionados (cruzamento) e introduzir variabilidade genética por meio de alterações aleatórias nos genes dos indivíduos (mutação).

O operador de cruzamento combina dois indivíduos para gerar indivíduos descendentes, chamados filhos. Existem três principais formas de cruzamento: ponto simples, multiponto e uniforme.

No cruzamento de ponto simples

Um ponto de corte é escolhido aleatoriamente nos indivíduos pais, e as partes antes e depois desse ponto são trocadas, formando os filhos.

Figura 1 – Funcionamento do cruzamento do tipo um ponto. Fonte: Rodrigo Kato

B No cruzamento multiponto

Vários pontos de corte são selecionados aleatoriamente nos indivíduos pais, e os segmentos entre esses pontos são trocados, gerando maior diversidade nos filhos.

Figura 2 – Funcionamento do cruzamento multiponto com K = 2. Fonte:Rodrigo Kato

C No cruzamento uniforme

Cada gene dos filhos é escolhido aleatoriamente de um dos pais, com base em uma máscara aleatória, resultando em uma mistura mais uniforme das informações genéticas.

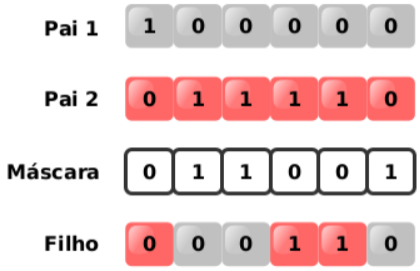

Figura 3 – Funcionamento do cruzamento uniforme. Fonte: Adaptado de Chaudhry e Usman (2017)

Esses operadores genéticos permitem a combinação de características dos pais e a introdução de novas informações genéticas na população, promovendo a exploração de diferentes soluções e aumentando a diversidade genética ao longo das gerações.

Mutação

A função de mutação em um algoritmo genético introduz alterações aleatórias nos genes dos indivíduos para evitar a estagnação da busca e promover diversidade. Ela modifica um ou mais genes, podendo convertê-los ou adicionar perturbações numéricas. A taxa de mutação determina a probabilidade de ocorrência da mutação, equilibrando a exploração e evitando mutações excessivas. A mutação complementa o cruzamento, permitindo a evolução do algoritmo genético e a descoberta de soluções melhores. Diversos tipos de mutação são descritos por Soni e Kumar (2014), como: a **mutação de inserção**, **de inversão** e **uniforme**.

Mutação de inserção

É um tipo de mutação em que um novo gene é inserido aleatoriamente em uma determinada posição do indivíduo. Isso pode ser feito deslocando os genes existentes para abrir espaço para o novo gene ou expandindo a representação genética do indivíduo para acomodar o novo gene. Essa mutação introduz uma nova informação genética na população, possibilitando a exploração de novas soluções.

Mutação de inversão

Envolve a inversão da ordem dos genes em um segmento específico do indivíduo. Um segmento contínuo de genes é selecionado aleatoriamente e a ordem dos genes nesse segmento é revertida. Isso pode levar a uma reorganização significativa da informação genética, possibilitando a busca por soluções diferentes na vizinhança do indivíduo original.

Mutação uniforme

É um tipo de mutação em que cada gene do indivíduo tem uma determinada probabilidade de ser alterado aleatoriamente. Para cada gene, um valor aleatório é gerado e comparado com a taxa de mutação. Se o valor aleatório for menor que a taxa de mutação, o gene é modificado. Essa mutação pode envolver a alteração de um bit em uma codificação binária ou a adição de uma pequena perturbação em um valor numérico.

Parametros

Os algoritmos genéticos possuem parâmetros que afetam seu funcionamento. Alguns desses parâmetros incluem:

- *Número de gerações:* critério de parada que determina quantas iterações o algoritmo realizará. Um número muito pequeno pode levar a resultados insatisfatórios, enquanto um número muito grande pode aumentar o tempo de execução.
- *Tamanho da população:* quantidade de indivíduos em cada geração. Pode ser fixo ou variar ao longo do tempo. Populações maiores aumentam o tempo de execução, enquanto populações menores podem não explorar todo o espaço de busca.
- *Probabilidade de cruzamento:* percentual que indica a chance de troca de material genético entre indivíduos para gerar descendentes. Taxas altas aceleram a introdução de novas características, mas podem levar à perda de indivíduos promissores.
- *Probabilidade de mutação:* chance de ocorrerem alterações nas características dos indivíduos. Evita a estagnação e introduz diversidade. Geralmente, taxas baixas são utilizadas.
- *Tamanho do torneio:* Parâmetro da seleção por torneio que controla a pressão seletiva. Seleciona

aleatoriamente N indivíduos e escolhe o melhor para a próxima geração, evitando convergência prematura.

A escolha adequada desses parâmetros depende do contexto do problema e pode impactar significativamente o desempenho do algoritmo genético.

Criterios de Parada

Existem duas principais formas de encerrar a execução de um algoritmo genético. A primeira ocorre quando é possível identificar um padrão ótimo nas características dos indivíduos que compõem a solução do problema. Nesse caso, não há mais necessidade de continuar executando o algoritmo e ele pode ser encerrado.

A segunda forma de término ocorre quando não é possível identificar um padrão ótimo nos indivíduos. Nesse caso, o algoritmo pode ser encerrado com base em diferentes critérios, tais como:

- O tempo máximo de execução do algoritmo ou o número de gerações é excedido.
- O número total de avaliações feitas pela função fitness é atingido.
- As melhorias nos indivíduos, realizadas pelos operadores genéticos e seleção, atingem um limite, ou seja, não ocorrem mais mudanças significativas.

Em geral, um algoritmo genético é encerrado quando uma das formas descritas anteriormente é satisfeita, ou seja, quando um valor ótimo (ou satisfatório) é alcançado pelos indivíduos ou quando uma condição de parada é cumprida.

III. Artigo Simulating nature's methods of evolving the best design solution

O artigo discute a aplicação de algoritmos genéticos (AGs) para resolver problemas de design. Os AGs são métodos adaptativos de busca que simulam processos naturais, como seleção, herança de informações, mutação aleatória e dinâmica populacional. Eles operam como simulações, onde agentes individuais, chamados cromossomos, competem e cooperam para alcançar uma melhor adaptação. A estrutura do cromossomo representa uma solução potencial para um problema específico, e os AGs utilizam mecanismos de seleção, reprodução e mutação para evoluir essa solução ao longo do tempo.

A ideia geral do artigo é que os AGs são capazes de resolver problemas de design simulando os processos naturais de seleção e herança presentes na evolução biológica. Eles podem ser aplicados em uma ampla variedade de áreas, desde a otimização de parâmetros numéricos até problemas mais gerais, graças a avanços na compreensão das propriedades necessárias da correspondência e no processamento de restrições do problema.

Os AGs são descritos como um modelo iterativo, onde uma população inicial é gerada e, em cada iteração, ocorrem as etapas de avaliação, seleção e reprodução. A pressão seletiva promove a sobrevivência dos indivíduos mais adaptados, enquanto a reprodução introduz variabilidade por meio do cruzamento e da mutação. Os AGs são robustos devido à capacidade de equilibrar eficiência e eficácia na busca por soluções.

O artigo também discute o uso de esquemas, que são modelos de semelhança entre cromossomos, para explorar regularidades nas informações representadas pelos cromossomos. Os esquemas são hiperplanos no espaço de busca e fornecem blocos de construção para o algoritmo. A seleção iterativa e os operadores reprodutivos permitem a exploração de novos esquemas e a geração de instâncias dos esquemas existentes.

É mencionado que os AGs exibem paralelismo implícito devido à manipulação de esquemas implícitos por meio de cromossomos explícitos. Isso também permite que os AGs sejam paralelizados usando tecnologias de processamento paralelo.Ao final, é apresentado um exemplo simplificado de um AG aplicado a um problema de maximização de função, onde exemplo ilustra a dinâmica da população, mostrando como a distribuição dos cromossomos se concentra em regiões com maior retorno esperado ao longo das iterações.

Em resumo, o texto discute a aplicação dos algoritmos genéticos na resolução de problemas de design, descrevendo seus princípios básicos, mecanismos de operação e como eles exploram regularidades nas informações representadas pelos cromossomos para encontrar soluções ótimas.

IV. jogo dinossauro

No vídeo Inteligência Artificial com Dinossauro da Google feito pelo Ivan Seidel é apresentado um solução para o jogo que utiliza redes neurais e algoritmos genéticos para treinar um agente virtual a jogar o jogo do dinossauro no navegador Google Chrome. O agente aprende a tomar decisões (pular ou abaixar) com base em informações dos sensores (distância, tamanho e velocidade dos cactos e pássaros). Através do uso de redes neurais, o programa mapeia as entradas dos sensores para as saídas correspondentes (ações do dinossauro).

A rede neural é composta por camadas de neurônios, onde cada neurônio recebe a entrada multiplicada por um peso. No caso do dinossauro, a camada final da rede decide se o dinossauro deve pular, abaixar-se ou não fazer nada, com base nos valores que chegam até ela. Um valor acima de 0.55 indica que o dinossauro deve pular, um valor abaixo de 0.45 indica que o dinossauro deve abaixar-se e valores intermediários indicam que o dinossauro não deve fazer nada.

Para utilizar algoritmos genéticos, os pesos da rede neural são tratados como um DNA, uma sequência linear de informações. Um conjunto inicial de DNAs (genomas) é gerado com valores aleatórios em cada gene. A cada entrada dos sensores, os genomas geram uma saída correspondente. Com base no desempenho das saídas, os melhores genomas são selecionados e combinados através do crossover para

criar novos genomas. Além disso, mutações aleatórias podem ocorrer nos genes dos genomas com uma probabilidade pré-definida.

Ao longo das iterações do programa, ocorre uma "seleção natural" dos melhores genomas, o que leva a uma melhoria contínua do desempenho do agente virtual. Com o tempo e algumas modificações na rede neural, é possível obter resultados ainda melhores do que os alcançados pelo exemplo mencionado no vídeo, onde o agente virtual conseguiu marcar 2532 pontos no jogo.

Esse tipo de abordagem combina elementos de aprendizado de máquina (redes neurais) com técnicas de algoritmos genéticos, permitindo que o agente aprenda a jogar o jogo do dinossauro sem informações prévias sobre o que é certo ou errado, adaptando-se e melhorando ao longo do tempo.

V. Tetris

O Tetris é um jogo de quebra-cabeça em que o jogador deve empilhar peças chamadas de tetriminos de forma organizada para eliminar linhas horizontais. O objetivo é marcar pontos eliminando linhas completas, enquanto as peças descem do topo da tela. O jogo termina quando as peças se acumulam até o topo. O Tetris é conhecido por sua jogabilidade viciante e desafiadora, exigindo decisões rápidas e estratégicas dos jogadores. É considerado um dos jogos mais influentes da história dos videogames.

Neste trabalho, descreve o funcionamento do algoritmo genético utilizado para jogar Tetris de forma automatizada. O algoritmo foi implementado em C++ e visa encontrar a melhor combinação de pesos para avaliar as características do jogo e tomar decisões de movimento que levem à remoção do maior número possível de linhas.

O algoritmo inicia com a criação de um objeto `CGenetic`, onde são especificadas a largura e altura do campo de jogo. Os pesos iniciais são gerados aleatoriamente para cada indivíduo da população.

A evolução do algoritmo ocorre através do método `evolution()`, que executa uma iteração do processo genético. Nessa iteração, a função `fitness()` avalia o desempenho de cada indivíduo em um conjunto de jogos, utilizando sua aptidão. Os indivíduos são ordenados por aptidão e selecionados para cruzamento.

O cruzamento entre os indivíduos selecionados é realizado através da função `crossover()`, gerando novos indivíduos com combinações de pesos herdadas dos pais. Além disso, uma pequena porcentagem dos indivíduos sofre mutação através da função `mutation()`, onde ocorrem pequenas alterações em seus pesos.

Durante a avaliação, cada indivíduo joga o Tetris várias vezes e sua aptidão é calculada com base no desempenho. Para encontrar o melhor movimento para cada indivíduo em cada jogo, é utilizada a função `searchMove()`. Os pais para o cruzamento são selecionados com base na aptidão.

Os pesos dos indivíduos são normalizados para garantir que a soma dos quadrados dos pesos seja igual a 1, evitando que pesos dominantes influenciam demais nas decisões de movimento.

A cada geração, a população é atualizada com os novos indivíduos gerados após o cruzamento e mutação. O processo de evolução se repete por várias gerações, buscando encontrar DNA candidatos com parâmetros de peso ótimos para estratégias de jogo eficientes no Tetris.

O código também inclui a implementação da classe `CComputer` em `IA.cpp`, que utiliza o algoritmo genético para jogar Tetris. Essa classe controla o estado inicial do jogo, realiza movimentos das peças com base nos resultados obtidos pelo algoritmo genético e trata das condições de fim de jogo.

De maneira geral, o algoritmo genético implementado busca encontrar a melhor combinação de pesos para avaliar e tomar decisões de movimento no jogo Tetris. O algoritmo evolui uma população de indivíduos através de seleção, cruzamento e mutação, avaliando seu desempenho em jogos e buscando melhorar a aptidão ao longo das gerações. O objetivo final é encontrar indivíduos com pesos otimizados que resultem em estratégias eficientes para o Tetris.

VI. Complexidade do algoritmo genetico

Se considerarmos "g" como o número de gerações, "n" como o tamanho da população e "m" como o tamanho do cromossomo, a complexidade para realizar essa tarefa seria proporcional a O(g * n * m). Isso significa que o tempo de execução do algoritmo aumentará de forma linear à medida que o número de gerações, o tamanho da população e o tamanho do cromossomo aumentarem.

References

[1] Link do repositorio: <https://github.com/KvWIlY/Kevin_Rafel_FinalProject_AA_RR_2023>.

[2] Algoritmos Genéticos, BIOINFO. Disponível em: <https://bioinfo.com.br/algoritmos-geneticos/>.

[3] JANIKOW, Cezary Z.; CLAIR, Daniel St.. Simulating nature's methods of evolving the best design solution.

[4] Tetris. Disponível em: <https://pt.wikipedia.org/wiki/Tetris>. Acesso em: 25 jun. 2023.

[5] BELOUSOV, Y. Project_Tetris. Disponível em: <https://github.com/TheTOXIN/Project_Tetris>. Acesso em: 25 jun. 2023.

[6] Artificial Intelligence in Google's Dinosaur (English Sub). Disponívelem: <https://www.youtube.com/watch?v=P7XHzqZjXQs>. Acesso em: 25 jun. 2023.