A picture containing building, drawing

Description automatically generatedUNIVERSIDADE FEDERAL DO PARANÁ

WANDERSON GIACOMIN JR – GRR20193923

MARCOS ANTONIO P. KERCHNER – GRR20184576

ALGORITMOS GENÉTICOS

UMA APLICAÇÃO PRÁTICA

CURITIBA

2021

WANDERSON GIACOMIN JR – GRR20193923

MARCOS ANTONIO P. KERCHNER – GRR20184576

**ALGORITMOS GENÉTICOS**

**UMA APLICAÇÃO PRÁTICA**

Trabalho apresentado como requisito parcial à aprovação da disciplina de Inteligência Artificial II, Setor de Educação Profissional e Tecnológica da Universidade Federal do Paraná.

curitiba

2021

Sumário

[1 introdução 4](#_Toc76298090)

[2 OBJETIVO 6](#_Toc76298091)

[2.1 O trabalho 6](#_Toc76298092)

[3 SOLUÇÃO DESENVOLVIDA 8](#_Toc76298093)

[3.1 definição da função objetivo 8](#_Toc76298094)

[3.2 Demais aspectos da solução 10](#_Toc76298095)

[3.3 RESPOSTA AO USUÁRIO 10](#_Toc76298096)

[4 conclusão 11](#_Toc76298097)

# introdução

Algoritmos genéticos, mais conhecidos como GA, são algoritmos que utilizam da otimização e são baseados nos princípios da genética e seleção natural. Esse tipo de algoritmo é frequentemente usado para se encontrar solução ótimas ou próximas das ótimas em problemas complexos que, de outra maneira, poderiam levar um tempo extremamente longo para serem resolvidos. Além disso, o seu uso se dá principalmente na otimização de problemas relacionados a pesquisa e aprendizado de máquina.

Antes de se adentrar no mundo do GA, é importante se ter bem trabalhada a ideia do objetivo de uma otimização. Em poucas palavras, otimizar é tornar alguma melhor, levar um processo a um estágio superior. O termo em si, se refere ao processo de se encontrar os parâmetros que retornaram um resultado ótimo para uma determinada função objetivo. Normalmente, esse processo se dá através da variação dos parâmetros de entrada com o intuito de se maximizar ou minimizar alguma saída.

Os algoritmos genéticos, então, pertencem ao grupo dos algoritmos estocásticos de otimização, mais especificamente pertence a um subgrupo conhecido como computação evolucionária.

Desenvolvido inicialmente por John Holland e seus colegas na Universidade de Michigan, esse tipo de algoritmo reserva uma população de solução possíveis para um determinado problema e, através da recombinação e mutação, produzem novas entradas e assim, novos resultados. Esse processo, assim como na genética, é repetido diversas vezes. Cada indivíduo por sua vez, ou candidato à solução, tem a sua função objetivo calculada para que seja feita “a seleção” dos melhores, semelhante ao termo utilizado na biologia “quem sobrevive é o mais forte”. Essa evolução perdura até que o critério de parada seja alcançado.

Pode-se citar algumas vantagens do GA como sendo:

* Não é necessário nenhum processo derivativo,
* Pode ser mais rápido e mais eficiente que alguns métodos tradicionais,
* Tem boa capacidade de paralelização,
* Consegue otimizar tanto funções contínuas como discretas, além de problemas multiobjetivos,
* Retorna uma lista de ‘boas’ solução e não apenas um resultado único,
* Sempre retorna uma resposta para o problema, mesmo não sendo o ótimo global,
* Muito útil quando o espaço de busca é muito grande e muitos parâmetros estão envolvidos.

Porém, também existem algumas limitações quando ao seu uso:

* Não é muito bom para problemas muito simples e possui um desempenho menor para problemas que possuem informações de suas derivadas,
* A função objetivo é calculada inúmeras vezes, o que pode custar muito computacionalmente para alguns problemas,
* Por ser estocástica, não há garantia da qualidade do resultado ou da descoberta do ponto ótimo,
* Se mal implementada, GA pode não convergir bem.

É interessante ainda apontar que a execução de algoritmos como o GA pode não gerar a mesma solução a cada execução. Na verdade, é bem mais provável que a solução encontrada seja diferente da anterior devido ao conjunto de fatores aleatórios implantados nesse algoritmo.

Em suma, algoritmos genéticos possuem habilidade de entregar de forma “rápida o suficiente” uma otimização “boa o suficiente”, tornando-o atraente para resolução de muitos problemas difíceis ou eficiente para encontrar solução quase-ótimas em períodos curtos. Um bom exemplo de sua aplicabilidade é o problema do “Entregador Viajante” (TSP). Algoritmos tradicionais levam muito tempo para encontrar soluções para esse tipo de problema cuja derivação não está disponível e, da mesma forma, algoritmos de força bruta são impraticáveis para grandes conjuntos de pontos. Dessa forma, GA consegue entregar relativamente rápido uma solução boa o suficiente para o problema.

# OBJETIVO

Um sistema para rastreamento de origem de sinais sonoros (SRSS) dispõe de três torres com sensores sonoros instalados nas seguintes respectivas coordenadas em um sistema de referência, previamente estabelecido em metros:

1. ​Sensor 1 (0, 0, 0);
2. ​Sensor 2 (2300, 1100, 20);
3. ​Sensor 3 (-170, 915, 11);

O SRSS também conta com algoritmo para a identificação de sinais de interesse, que registra os tempos em que um mesmo sinal chegou a cada um dos sensores, (t1, t2, t3), em cronômetros sincronizados. Considerando a velocidade do som como 340 m/s:

*construir uma solução, em um algoritmo genético (AG), para facilitar a identificação da origem de um estampido no sistema de referência utilizado, a partir dos tempos observados nos sensores.*

## O trabalho

1. A solução deve ser construída na plataforma e linguagem escolhida pela equipe, desde que os conceitos de AG sejam utilizados de forma correta e justificada no relatório.
2. ​O programa deve usar algoritmos genéticos para propor soluções coerentes.
3. ​O Trabalho pode ser realizado em equipes de até três pessoas.
4. ​O procedimento a ser usado para gerar e utilizar o gráfico de apresentação do resultado é parte integrante do trabalho e deve compor a avaliação;
5. ​A equipe deve construir o seu ambiente de teste e um problema análogo, diferente (outras coordenadas para os sensores), pode ser proposto no dia da avaliação;
6. ​Um relatório técnico apresentando a solução ser entregue ao final da disciplina.
7. ​A avaliação levará em conta a qualidade da solução, a programação e a apresentação dos resultados (relatório e apresentação oral).

Esse relatório tem como objetivo apresentar uma possível solução para o problema proposto.

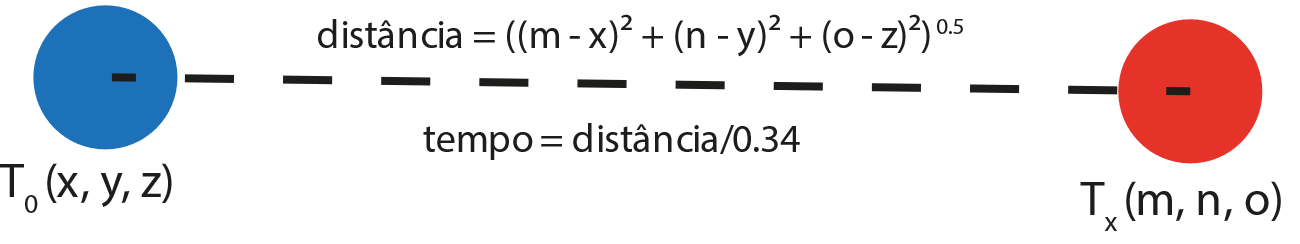
# SOLUÇÃO DESENVOLVIDA

A solução desenvolvida utilizou a linguagem Python juntamente com algumas bibliotecas já existentes:

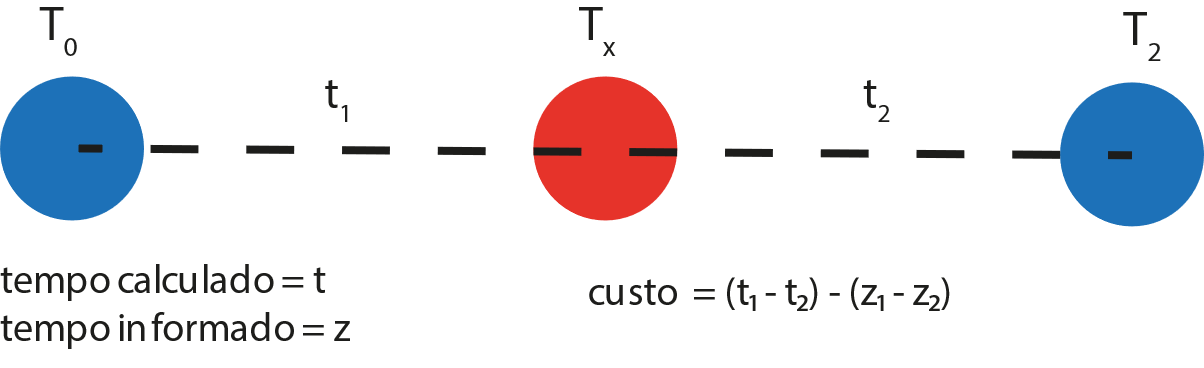
* geneticalgorithm
* Flask
* Plotly
* Numpy
* Pandas

## definição da função objetivo

A função objetivo foi determinada utilizando-se como base a distância euclidiana sendo alimentada por um ponto Tx contendo 3 coordenadas do espaço, conforme a imagem abaixo.



Após calculada a distância entre o ponto alimentado e as três torres, a distância dada em metros é convertida em tempo na unidade de milissegundos. A partir desse dado, é então calculado o custo da função com base em um ponto relativo. Por exemplo, na imagem abaixo, o custo da função é o tempo relativo entre T0 e Tx usando-se como base o ponto T2. Assim tempo que o custo é a diferença entre os tempos calculados dos dois pontos menos a diferença dos tempos informados entre os dois pontos.



Esse raciocínio é generalizado para os demais pontos e então penalizado para se promover uma curva mais acentuada e melhorar a conversão de pontos mais distantes. Em notações matemáticas, a distância entre 2 pontos se dará como:

o cálculo da função do desvio é então dado como:

sendo,

k, o tempo informado

t, o tempo calculado

x, o ponto informado como argumento na função

T, uma coordenada de 3 dimensões

Como dito anteriormente, a elevação ao quadrado serve como penalização para pontos muito distantes do objetivo. Os dois algoritmos estão exibidos abaixo:

**def** **calc\_dist**(p1, p2):  
 **return** np.sqrt(np.power(p1[0] - p2[0], 2)

+ np.power(p1[1] - p2[1], 2)

+ np.power(p1[2] - p2[2], 2))  
  
**def** **f**(t3):  
 **global** t0, t1, t2  
 t3\_t1 = calc\_dist(t3, T1) / .34  
 t3\_t2 = calc\_dist(t3, T2) / .34  
 t3\_t0 = calc\_dist(t3, T0) / .34  
  
 delta = np.power(t3\_t1 - t3\_t0 - t1 + t0, 2)

+ np.power(t3\_t2 - t3\_t0 - t2 + t0, 2)  
 **return** delta

## Demais aspectos da solução

Definida a função objetivo, é apenas necessário desenvolver a interface e se definir os parâmetros. O código de interface foge ao objetivo desse relatório, então se fará apenas presente a definição dos parâmetros utilizados na biblioteca da GA.

Definiu-se que o algoritmo se executará duas vezes, caso a primeira não atinja um custo zero, dessa forma, podemos pressupor que o resultado será melhor que com uma solução única obtida pelo código, caso o ponto ótimo não seja encontrado na primeira tentativa.

Quanto aos parâmetros escolhidos, foram os seguintes:

|  |  |
| --- | --- |
| **Parâmetro** | **Valor** |
| Número máximo de iterações | - |
| Tamanho da população | 300 |
| Probabilidade de mutação | 0.15 |
| Porcentagem de elite | 0.01 |
| Probabilidade de crossover | 0.5 |
| Porção parental | 01 |
| Tipo de crossover | Uniforme |
| Número máximo de iteração sem melhora | 120 |
| Tipo de abordagem | Números inteiros |

A definição se deu a partir de testes e análise dos resultados com conjuntos de dados conhecidos.

## RESPOSTA AO USUÁRIO

Para o usuário, a aplicação é transparente, havendo apenas um campo para preenchimentos dos dados necessários, um gráfico interativo e um pequeno espaço para a devolução do resultado.

Chart, scatter chart

Description automatically generated

# conclusão

Diante do apresentado, foi constatado que o algoritmo escolhido possui um ótimo desempenho para o problema proposto. Percebe-se, porém, que existe uma necessidade de otimização que se dá principalmente pelo desempenho da biblioteca escolhida.

Outro ponto é quanto aos parâmetros. É sabido que existem hoje ferramentas que ajudam na definição dos mesmos e em seus testes. Isso deve ser visto como pontos futuros de melhoria.