UNIVERSIDADE FEDERAL DO PARANÁ

MARÍA LUCÍA RODRÍGUEZ

ESTUDO DOS RESÍDUOS DE UMA SÉRIE TEMPORAL DA USINA HIDRELÉTRICA DE ITAIPU USANDO ALGORITMOS GENÉTICOS

CURITIBA
2018

MARÍA LUCÍA RODRÍGUEZ

ESTUDO DOS RESÍDUOS DE UMA SÉRIE TEMPORAL DA USINA HIDRELÉTRICA DE ITAIPU USANDO ALGORITMOS GENÉTICOS

Monografia apresentada como requisito parcial à obtenção do título de Especialista, Curso de Especialização em Métodos Numéricos em Engenharia, Setor de Ciências Exatas e Setor de Tecnologia, Universidade Federal do Paraná.

Orientador: Prof[a]. Dr[a]. Isabella Andreczevski Chaves.
Coorientador: Prof. Dr. Jairo Marlon Corrêa

CURITIBA
2018

FOLHA/TERMO DE APROVAÇÃO

MARÍA LUCÍA RODRÍGUEZ

ESTUDO DOS RESÍDUOS DE UMA SÉRIE TEMPORAL DA USINA HIDRELÉTRICA DE ITAIPU USANDO ALGORITMOS GENÉTICOS

Monografia aprovada como requisito parcial à obtenção do título de Especialista, Curso de Especialização em Métodos Numéricos em Engenharia, Setor de Ciências Exatas e Setor de Tecnologia, Universidade Federal do Paraná, pela seguinte banca examinadora:

Dra. Isabella Andreczevski Chaves
Orientadora - Departamento de Construção Civil - UFPR

Dr. Anselmo Chaves Neto
Programa de Pós-Graduação em Métodos Numéricos em Engenharia - UFPR

Dr. Emerson Lazzarotto
Centro de Engenharia e Ciências Exatas - UNIOESTE

CURITIBA
2018

Você pode atravessar montanhas,
oceanos, tragédias, dificuldades
com apenas uma coisa,
confiança em si mesmo
Yogi Bhajan

Dedicatória

Dedico este trabalho a meu marido Victor Obrist, que também é meu companheiro de estudo, de trabalho e de vida, quem me inspira e impulsiona através do seu amor e apoio incondicional para que eu seja cada vez melhor.

Agradecimentos

A Deus, pois está acima de tudo e é tudo.

A minha avó, Jagan Nath K. K. Você é diretamente responsável por tudo o que eu sou hoje.

Quero agradecer imensamente a Dra. Isabella Andreczevski Chaves, minha Orientadora, por todo o apoio e contenção durante o desenvolvimento do projeto, bem como a excelente guia, as forças, todo o tempo brindado sem importar o horário, dia, feriado, ferias, no todo o desenvolvimento do trabalho. Sem o apoio de você professora eu não iria saber a direção para onde ir e muito menos iria atingir os objetivos no tempo e prazo.

Ao Dr. Jairo Marlon Corrêa, por toda a ajuda brindada, a boa disposição e gentileza em todo momento.

Ao CEASB do PTI, pela oportunidade de fazer o programa e por brindar todo o apoio logístico e direção durante este trabalho.

Aos professores do PPGMNE UFPR, por todas as aulas e ensinamentos.

A UFPR por ter me proporcionado uma excelente formação.

Resumo

Neste trabalho foi realizada a previsão dos resíduos de uma série temporal que representam os Deslocamentos Horizontais de um Bloco de concreto da Barragem principal da Usina Hidrelétrica de Itaipu, por meio da aplicação híbrida de um Modelo ARIMA, da metodologia Box & Jenkins no modelo linear da série e aplicando Algoritmos Genéticos ao modelo não linear da série. Logo apos foram comparados os resultados do Modelo Híbrido com os resultados do Modelo ARIMA puro, sendo que o Modelo Híbrido reduziu o erro de previsão em um 30%. Destaca-se a natureza inovadora da aplicação desta técnica em séries temporais.

Palavras chaves: Séries Temporais, Algoritmos Genéticos, Box & Jenkins, Resíduos, Barragens.

Abstract

In this work, the residuals of a time series representing the horizontal displacements of a concrete block of the main dam of the Itaipu Hydroelectric Power Plant were predicted, through the hybrid application of an ARIMA Model, of the Box & Jenkins methodology in the linear model of the series and applying Genetic Algorithms to the nonlinear model of the series. Soon, the results of the Hybrid Model were compared with the results of the pure ARIMA Model, and the Hybrid Model reduced the prediction error by 30%. We highlight the innovative nature of the application of this technique in Time Series.

Key words: Time Series, Genetic Algorithms, Box & Jenkins, Residues, Dams.

Lista de Figuras

Lista de Gráficos

Lista de Tabelas

Sumário

Capítulo 1

Introdução

A técnica de Algoritmos Genéticos, dentro da área de Inteligência Artificial, é uma poderosa técnica para a busca de soluções ótimas para determinados problemas. Essa técnica tem seus fundamentos na teoria genética e na teoria da evolução das espécies, onde emula a maneira de reprodução dos cromossomos humanos, onde sobre cada gene se aplicasse diversos operadores genéticos para a obtenção de resultados.

O conceito de algoritmos genéticos é baseado na ideia de que a computação, especialmente o campo da inteligência computacional, tem se beneficiado muito da observação da natureza. Através deste observação surgiram, entre outros, as redes neurais (simulação do comportamento do cérebro humano), o resfriamento simulado (simulação do processo de resfriamento de metais) e, finalmente, os algoritmos genéticos (simulação da evolução natural) (LINDEN, 2012).

Eles usam processos biológicos em software para encontrar respostas a problemas que têm espaços de pesquisa realmente grandes gerando soluções possíveis continuamente, avaliando o quão bem as soluções se encaixam no resultado desejado e refinando as melhores soluções.

Têm-se a disposição vários exemplos para a adoção da técnica de algoritmos genéticos, tais como: problemas de otimização em diversas áreas, análise e previsão de modelos econômicos, aprendizagem de máquinas, *data mining*, entre outros.

Algoritmos genéticos são uma das ferramentas que podemos usar para aplicar aprendizado de máquina para encontrar boas soluções; às vezes até ótimo, para problemas que têm bilhões de soluções potenciais, processos biológicos em software

usados para encontrar respostas para os problemas que têm realmente grandes espaços de busca através da geração contínua de soluções possíveis, avaliando a forma como as soluções são ajustadas para o resultado desejado e refinando as melhores soluções (MASSSAGO, 2013).

O processo de Algoritmos Genéticos (AG) começa com a criação da primeira geração, de acordo com certos parâmetros, tais como: número de indivíduos por geração, número de gerações, valor da função *Fitness* que será a aprovar. Em seguida, aos indivíduos são aplicados Operadores Genéticos de Cruzamento, Mutação e Seleção, sendo que no processo de Seleção é definido se o indivíduo avança para a próxima geração de acordo com sua aptidão *Fitness*. Os Operadores Genéticos são aplicados aos novos indivíduos da próxima geração e assim por diante até obter a melhor solução com o algoritmo.

Com base nas soluções fornecidas pelos Algoritmos Genéticos acima mencionados, este trabalho nasceu da necessidade de encontrar uma melhor previsão das soluções para o problema de segurança das barragens, de modo a proporcionar uma melhor previsão dos valores futuros, a partir dos lançados diariamente pelo pêndulo direto do bloco principal da represa de Itaipu.

A segurança da barragem é um ponto crucial que deve ser monitorada e assim encontrar várias maneiras que ajudem a manter o funcionamento correto da infraestrutura dentro das margens estabelecidas. Nesse contexto, a previsão de valores o mais perto possível beneficia a engenheiros e técnicos para adotar medidas corretivas, sendo aplicada uma metodologia híbrida ARIMA/Algoritmos Genéticos para o melhor ajuste das previsões de valores dos deslocamentos de pêndulo direto.

1.1 Objetivos

1.1.1 Objetivo Geral

Este estudo tem como objetivo geral ajustar aos resíduos de uma série temporal de valores usando Algoritmos Genéticos para a obtenção de prognósticos mais precisos.

1.1.2 Objetivos Específicos

- Ajustar o modelo Box & Jenkins à série temporal.
- Aplicação de Algoritmos Genéticos aos resíduos dado o ajuste do modelo ARIMA à série temporal.
- Codificar na linguagem de programação *Python* usando o as bibliotecas *DEAP* e *Scikit learn.*
- Comparar os resultados obtidos através da aplicação da metodologia proposta com os obtidos através da aplicação da metodologia Box & Jenkins.

1.2 Justificativa

A questão abordada do paradigma de AG deste trabalho são as previsões das séries temporais para a segurança de barragens utilizando os pêndulos do tipo: diretos e invertidos, especificamente, dos deslocamentos horizontais de blocos-chave da barragem principal da Usina Hidrelétrica de Itaipu, local onde estão instaladas as unidades geradoras de energia.

As informações sobre os valores de tais deslocamentos da estrutura de uma barragem são uma das questões mais preocupantes para técnicos e engenheiros. Porém, ter os valores de previsão de séries temporais permitem uma projeção que melhor se aproxima da realidade ao longo do tempo, possibilitando melhores tomadas de decisões em quanto a prevenções de possíveis riscos da barragem.

1.3 Estrutura da Monografia

O presente trabalho está composto por seis capítulos: (1) Introdução; (2) Revisão Bibliográfica; (3) Referencial Teórico; (4) Materiais e Métodos (5) Resultados e Discussão; (6) Conclusão e Considerações Finais, finalizando com as Referências Bibliográficas.

No capítulo 1 apresenta-se um breve resumo do trabalho, a justificativa e os objetivos propostos, tanto o geral como os específicos. No capítulo 2 é apresentado resumos e conclusões de artigos relacionados ao trabalho. No capítulo 3 são

explanados os conceitos e descrições de técnicas usadas no trabalho, tais como: Modelos de Box & Jenkins, Algoritmos Genéticos. A seguir, no capítulo 4, são apresentados o material (séries temporais utilizadas) e a descrição da metodologia proposta. Já no capítulo 5 são apresentados e discutidos os resultados obtidos utilizando algumas das técnicas presentes no método descrito no capítulo 3. No capítulo 6 estão as Considerações Finais e a Conclusão. Finalmente, as Referências na última parte do trabalho.

Capítulo 2

Revisão Bibliográfica

Foi utilizado o método de Algoritmos Genéticos na resolução do problema de comparação entre os métodos: clássico Modelo de Predição *Classic* UCM, Redes Neurais Artificiais RNAs e Algoritmos Genéticos baseado no Índice de Produção Industrial do Estado do Rio Grande do Sul fornecido pelo IBGE. De acordo com o que foi apresentado no trabalho, a técnica de Algoritmos Genéticos pode ser considerada como ferramenta na predição de séries temporais, pois são capazes de realizar prognósticos com o mesmo nível de predição que os outros métodos citados anteriormente (MARQUES, 2012).

O artigo apresenta uma nova metodologia de parametrização do indicador de análise técnica do mercado financeiro chamado *Moving Average Convergence-Divergence* (MACD) utilizando algoritmos genéticos e lógica nebulosa quando aplicado ao preço de valores mobiliários, para indicar o melhor momento de compra e venda das ações. Os resultados obtidos no trabalho revelam que essa metodologia pode ser usada com sucesso, proporcionando lucros superiores a 70%, principalmente quando comparados as duas configurações usuais que apresentaram lucros de 57,94% e 29,46% (MARQUES; GOMES., 2009).

Foi utilizado o método de Algoritmos Genéticos para estimação dos hiperparâmetros do modelo de amortecimento exponencial *Holt-Winters* com Múltiplos Ciclos tomando como base a série histórica de carga elétrica horária de uma das concessionárias de energia elétrica do sudeste brasileiro, a ESCELSA. De acordo com o que foi apresentado no trabalho, foi concluído que o método de otimização é muito bom para minimizar o MAPE (*Mean Absolute Percent Error*), e com isso ter uma boa previsão da série em questão (carga de energia elétrica) (CORRÊA; PACHECO; SOUZA, 2001).

O objetivo do estudo foi desenvolver um modelo de algoritmo genético (método heurístico), que permita aos tomadores de decisões determinar o período de intervenção das equipes de corte nos pontos de produção, para minimizar os custos com as atividades relacionadas à colheita e ao transporte principal de madeira. De acordo com o que foi apresentado no trabalho, as soluções dos algorítimos genéticos apresentaram custos de colheita, transporte e estoque excedente de madeira entre 2,0% e 5,2% superiores aos do modelo de programação linear (SOUZA, 2004).

Foi estudado a possibilidade da aplicação de algoritmos evolutivos ao problema da estimação de um parâmetro que modela a degradação por efeito da força cortante em elementos de concreto armado.Os resultados obtidos do trabalho demonstram que as estratégias de evolução podem melhorar notavelmente os resultados obtidos por meio de métodos de regressão, e tem uma aplicação clara em problemas aonde o alcance de busca não se conhece ou é impossível de calcular (GUTIÉRREZ D. J. M. C. CANALES, 2017).

O artigo apresenta uma revisão bibliográfica dos métodos que foram utilizados nas últimas duas décadas a fim de prever índices bursáteis. Existem muitos modelos e métodos. Os primeiros modelos foram os auto regressivos, conhecidos como ARIMA, que de acordo com o que foi apresentado no trabalho, foi concluído que o ARIMA simplesmente capturava as características lineares da bolsa de valores, e não podem capturar as características não lineares presentes nas séries temporais financeiras, como as séries bursáteis, sendo necessário combinar o ARIMA com outros métodos, como os algoritmos genéticos e redes neurais, que possam modelar as características não lineares e melhorar a previsão (GARCÍA et al., 2017).

Capítulo 3

Referencial Teórico

Neste capítulo, será feita uma breve revisão de alguns conceitos importantes utilizados neste trabalho.

3.1 Série Temporal

"Série Temporal é um conjunto de observações sobre uma variável, ordenado no tempo", e registrado em períodos regulares. Pode-se enumerar os seguintes exemplos de séries temporais: temperaturas máximas e mínimas diárias em uma cidade, vendas mensais de uma empresa, valores mensais do IPC-A, valores de fechamento diários do IBOVESPA, resultado de um eletroencefalograma, gráfico de controle de um processo produtivo.

A suposição básica que norteia a análise de séries temporais é que há um sistema causal mais ou menos constante, relacionado com o tempo, que exerceu influência sobre os dados no passado e pode continuar a fazê-lo no futuro. Este sistema causal costuma atuar criando padrões não aleatórios que podem ser detectados em um gráfico da série temporal, ou mediante algum outro processo estatístico (REIS, 2018).

3.2 Modelos Box & Jenkins

A metodologia Box & Jenkins é, sem dúvida, o mais importante trabalho na área de Previsão de Séries Temporais. Foi esse estudo o responsável pelo grande desenvolvimento e a correspondente formalização da área de estudo de Séries Temporais. O trabalho dos pesquisadores Box & Jenkins foi baseado no importante resultado "qualquer série temporal pode ser representada por uma estrutura de médias móveis infinita", ou melhor, "qualquer processo estocástico estacionário Yt pode ser representado como a soma de dois processos mutuamente inter-relacionados, Yt = Dt + At, onde Dt é linearmente determinístico (sistemático) e At é um processo Médias Móveis infinito (MA(∞))" (NETO, 2017).

3.3 ARIMA

Segundo Box e Jenkins (1970 apud (TEIXEIRA; ET.AL, 2017)) um modelo ARMA plausível para a série temporal $\{y_t\}_{t=1}^{T}$, com cardinalidade T, de ordem p (autorregressivo - AR) e q (médias móveis - MA) é descrito pela equação 3.1.

$$y_t = \delta + \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} - \theta_1 a_{t-1} - \theta_2 a_{t-2} - ... - \theta_q a_{t-q} + a_t \quad (3.1)$$

O modelo dado pela equação 3.1 combina valores passados das entradas y_t e choques aleatórios (a_t) descorrelacionados, de média zero e variância constante. Neste modelo $\phi_i \theta_j \in$ R, com $i = 1, ..., p$ e $j = 1, ..., q$, denotam os parâmetros do modelo e δ uma constante. Na hipótese da série temporal ser não estacionária, esta deve ser diferenciada e o modelo ARMA(p,q) substituído pelo ARIMA(p,d,q), sendo d a ordem de diferenciação da série. Para a identificação do modelo, as ordens p e q podem ser determinadas por meio da análise do perfil dos gráficos das funções de autocorrelação (FAC) e autocorrelação parcial (FACP) (HAMILTON, 1994 apud (TEIXEIRA; ET.AL, 2017)).

Identificado o modelo, passa-se ao estágio seguinte que é a estimação dos parâmetros. Para tanto, é necessário utilizar métodos iterativos não lineares de mínimos quadrados. Maiores detalhes destas aplicações podem ser encontrados em Box e Jenkins (1970 apud (TEIXEIRA; ET.AL, 2017)) e Morettin e Toloi (2006 apud (TEIXEIRA; ET.AL, 2017)). Para a validação do modelo já com os parâmetros estimados pode-se usar testes estatísticos, tais como: teste de Box-Pierce, teste

do periodograma acumulado, teste da autocorrelação residual, entre outros. No caso da série temporal $\{y_t\}_{t=1}^{T}$ apresentar componente sazonal, o modelo de Box & Jenkins plausível é o ARIMA multiplicativo, dado genericamente na equação 3.2.

$$\phi(B)(1-\Phi_1 B-...-\Phi_p B^{PS})\bigtriangledown^d(1-B^S)^D y_t = \theta(B)(1-\Theta_1 B-...-\Theta_Q B^{QS})a_t \quad (3.2)$$

onde: $\phi(B) = (1 - \phi_1 B - ... - \phi_p B^p)$,$\theta(B) = (1 - \theta_1 B - ... - \theta_q B^q)$,$d$ é a ordem das diferenças simples, D é a ordem das diferenças sazonais, S é o período sazonal, $\phi_k \in$ R e $\theta_j \in$ R são os coeficientes dos polinômios não sazonais e $\Phi_m \in$ R e $\Theta_n \in$ R são os coeficientes dos polinômios sazonais.

A fim de validar o modelo ajustado foram analisados os resíduos com a construção dos gráficos das funções FAC e FACP, buscando mostrar que os mesmos são não autocorrelacionados.

Para testar a hipótese que todos os coeficientes de autocorrelação p_k sejam iguais a zero, foi utilizada a estatística $Q^* = n\sum_{k=1}^{m}\hat{p}_k^2$, desenvolvida por Box e Pierce, em que n é o tamanho da amostra e m a duração da defasagem. A estatística Q^* tem distribuição qui-quadrado com m graus de liberdade. Quando Q^* excede o valor crítico, rejeita-se a hipótese nula de que todos os p_k são iguais a zero (GUJARATI, 2000 apud (TEIXEIRA; ET.AL, 2017)).

3.4 Análise dos Resíduos

Após um modelo ter sido ajustado a uma série temporal deve-se verificar se o modelo fornece uma descrição adequada dos dados. Assim como em outros modelos estatísticos a ideia é verificar o comportamento dos resíduos onde

$$\text{resíduo} = \text{observação - valor ajustado.}$$

Para os modelos vistos aqui o valor ajustado é a previsão 1 passo à frente de modo que o resíduo fica definido como o erro de previsão 1 passo à frente (EHLERS, 2005).

3.5 Algoritmo Genético (AG)

Algoritmo Genético (AG) consiste em uma técnica de Inteligência Artificial (IA) que se fundamenta em teorias e conceitos da genética e da evolução de populações de seres vivos. A implementação de tal técnica visa possibilitar que soluções ótimas sejam encontradas para resolver problemas do mundo real, detectados em diversas áreas de conhecimento.

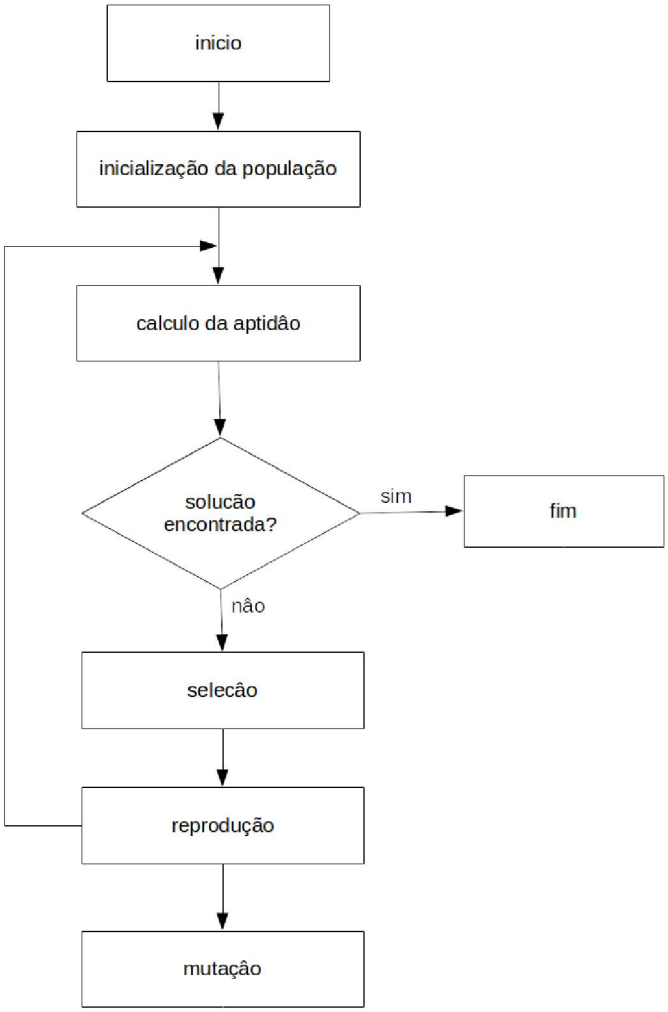
Algoritmo Genético é uma técnica de IA que foi criada com o intuito de imitar determinados processos observados na evolução natural das espécies. Deste modo, tal técnica fundamenta-se nas explicações oferecidas por Charles Darwin a respeito da seleção e evolução dos indivíduos na natureza, como também, em outras teorias de genética formuladas, posteriormente, por estudiosos tais como Gregor Mendel. Assim, o objetivo da técnica de AG consiste em solucionar problemas do mundo real de forma otimizada, sendo aplicável a diversas áreas.

Para compreender o funcionamento dos AGs faz-se necessário realizar uma analogia à explicação sobre a evolução das espécies. Assim, o AG trabalha da seguinte forma:

- Inicialmente é gerada uma população formada por um conjunto aleatório de indivíduos, que podem ser vistos como possíveis soluções do problema.
- Durante o processo evolutivo, esta população é avaliada, sendo que para cada indivíduo é atribuída uma nota, ou índice, que reflete sua habilidade de adaptação a determinado ambiente.
- Uma porcentagem dos indivíduos mais adaptados é mantida, enquanto os outros são descartados.
- Os membros mantidos pela seleção podem sofrer modificações em suas características fundamentais por meio de cruzamentos (*crossover*), mutações ou recombinação genética gerando descendentes para a próxima geração.
- Este processo, chamado de reprodução, é repetido até que uma solução satisfatória seja encontrada. Embora possam parecer simplistas do ponto de vista biológico, estes algoritmos são suficientemente complexos para fornecer mecanismos de busca adaptativos poderosos e robustos (ROSA; LUZ, 2009).

A estrutura básica do Algoritmo Genético é mostrada na Figura 3.1 a seguir.

Figura 3.1 –Estrutura Básica de um Algoritmo Genético Simples.



FONTE: (MIRANDA, 2017)

3.5.1 Codificação do problema

Inicialmente, o problema precisa ser codificada. O problema apropriada para aplicar o algoritmo genético é o problema de otimização, isto é, obter o valor mínimo (ou máximo) sujeito a alguma condição específica.

Analisando o problema, determina-se os parâmetros numéricos e suas condições que caracteriza uma solução factível (solução que satisfaz a condição do problema). Esta sequência de números é denominado de cromossomos. Inicialmente, somente os cromossomos formados pelos números inteiros foram usados, mas com o tempo, os números reais também começam a ser usados com frequência.

Para servir de exemplo, vamos supor que queremos obter o mínimo da função $f(x) = \sum x_i^2$. Em geral, dado o problema de encontrar o mínimo de $f : X \to Y$, $x \in X$ é o cromossomo. O cromossomo costuma ser sequências de números, sendo armazenados no vetor dos números. Um individuo codificado armazena o cromossomo e as informações relevantes obtidos a partir dele. No caso de $f : X \to Y$, o individuo codifica $x \in X$ e $z = f(x)$.

Por exemplo, no nosso exemplo, um individuo pode armazenar o ponto x e o valor $z = f(x)$. Vamos codificar o cromossomo no vetor x e seu valor na variável z (MASSSAGO, 2013).

```
individuo
        x : vetor de tamanho N dos reais
        z : real
fim individuo
```

FONTE: (ROSA; LUZ, 2009)

3.5.2 População Inicial

O AG inicia gerando um número pré-definido de soluções iniciais, aleatoriamente, formando a população inicial. Computacionalmente, a implementação deste procedimento é muito simples, em função de existirem boas funções geradoras de números aleatórios, na maioria das ferramentas de programação. Este procedimento torna-se adequado à codificação dos *strings* segundo os blocos do circuito, conforme mencionado no item anterior (BENTO; KAGAN, 2008).

Uma população é o conjunto dos indivíduos. Cada individuo é representado pelos seus cromossomos. Em geral, podemos representar uma população como sendo vetor de indivíduos.

```
populacao : vetor de tamanho POP_SIZE de individuo
```

FONTE: (ROSA; LUZ, 2009)

Em cada iteração do algoritmo genético, uma nova população será gerada a partir do atual, através do chamado operador genético. A forma como gera a nova população varia de caso por caso, devendo escolher a forma mais apropriada para cada problema. No entanto, existem duas formas opostas importantes: *steady-state* e generacional.

No *steady-state*, uma nova população é obtido pela população antiga, trocando com o novo individuo gerado somente quando ele for melhor que pior da população. Em termo operacional, gera os novos indivíduos e troca com o pior da população, caso ele for melhor. Caso não supere o pior, novo individuo será descartado. A ideia é juntar os novos indivíduos na população, ordenar pela aptidão e truncar pelo tamanho da população. No generacional, exceto uma seleta de indivíduos denominados elites, todos serão trocados pelo novo individuo. Se não tiver os elites, seria próximo ao que acontece na natureza.

3.5.3 Inicialização, evaluação e ordenação

Inicialmente, gera os indivíduos iniciais da população aleatoriamente. Para nosso exemplo, vamos supor que o tamanho da população é POP_SIZE. Então gera POP_SIZE de n-uplas de números sorteados entre A e B. Para cada par de pontos, calcula o valor de f.

A população é inicializado pela função chamado novo_individuo.

```
novo_individuo
        para i =1..N
                individuo.x[i] = A+(B-A)* drand
        fim para
        individuo.z = f(individuo . x )
        retorna individuo
fim novo_individuo
```

FONTE: (ROSA; LUZ, 2009)

Agora ordenamos o individuo em termos dos valores de f. Como queremos o menor valor de f, o melhor individuo fica no começo da lista.

```
inicializa
        para k =1..POP\_SIZE
                populacao [ k ] = novo_individuo
        fim para ordena populacao
```

fim inicializa

FONTE: (ROSA; LUZ, 2009)

3.5.4 Operadores Genéticos

Os operadores genéticos têm por objetivo realizar transformações em uma população, fazendo com que, a cada nova geração, indivíduos cada vez mais capazes sejam criados, contribuindo assim para que as populações evoluam a cada nova geração. Com isto, os operadores genéticos são classificados em: inicialização, função de aptidão, seleção, cruzamento, mutação, atualização e finalização. Sendo que destes, destacam-se os de seleção, cruzamento e mutação, responsáveis por conduzirem a busca no sentido da detecção da melhor solução.

3.5.5 Operador Genético Seleção

O operador probabilístico *Seleção Proporcional* empregado no AG básico sorteia os indivíduos de uma população para cruzamento, onde cada um tem chances de sorteio proporcionais aos seus respectivos valores de avaliação, isto é, a probabilidade de sorteio de um indivíduo i é dado pela equação 3.3:

$$p_i = \Phi(a_i) / \sum_{j=1}^{u} \Phi(a_j) \tag{3.3}$$

onde μ representa o tamanho da população e $\Phi : B^i \rightarrow \Re^+$, a função de avaliação. Este método de seleção é também conhecido por "Regra da Roleta".

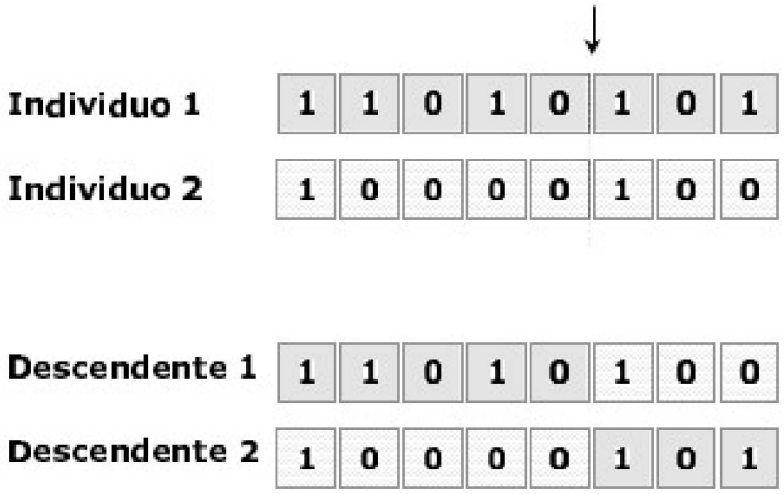
3.5.6 Elitismo

O AG básico descarta a geração anterior e considera para a futura apenas os descendentes obtidos. A técnica Elitista consiste em reintroduzir o indivíduo melhor avaliado de uma geração para a seguinte, evitando a perda de informações importantes presentes em indivíduos de alta avaliação e que podem ser perdidas durante os processos de seleção e cruzamento. Algumas técnicas controlam o número de vezes que o indivíduo pode ser reintroduzindo, o que contribui para evitar convergência a máximos locais.

3.5.7 Cruzamento

O operador Cruzamento troca arbitrariamente *sub-strings* entre dois indivíduos selecionados para reprodução, como se mostra na Figura 3.2 adiante.

Figura 3.2 –Processo de Cruzamento simples



FONTE: O autor (2018)

Cruzamento (recombinação): Escolhe-se dois indivíduos na população, denominado de pai. O novo individuo é obtido, combinando os cromossomo de cada um deles. Existem várias formas de combinar os cromossomos, resultando diversos operadores de cruzamento. Para nosso exemplo, vamos supor que cada gene é a média ponderada dos genes correspondentes dos pais.

```
cruzamento (pai1, pai2)
        para x = 1..N
                t = drand
                individuo.x[i] = t*pai1.x[i] + (1-t)*pai2.x
                    [i]
        fim para
        individuo.z = f(individuo.x)
        retorna individuo
```

```
fim cruzamento
```

FONTE: (ROSA; LUZ, 2009)

Para cruzamento, é desejável que evite que pai1 seja diferente de pai2. Poderá fazer sem muita dificuldade e melhorar a eficiência do cruzamento, mas não será obrigatória (pior dos casos, perderá uma parcela dos individuos obtidos por cruzamentos).

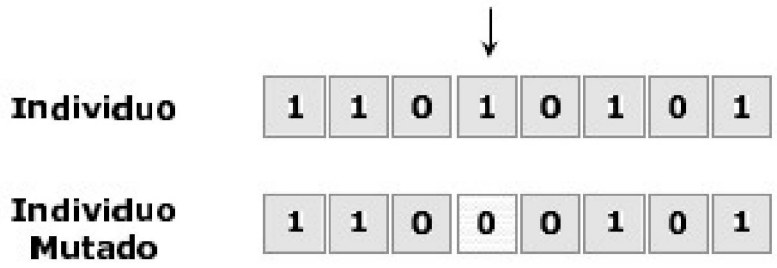
```
seleciona_pais(i1, i2)
        i1 = rand (POP_SIZE)
        i2 = rand (POP_SIZE-1)
        se i2 <= i1
                entao i2 = i2 + 1
        fim se
fim seleciona_pais
```

FONTE: (ROSA; LUZ, 2009)

3.5.8 Mutação

No AG básico, o papel central da pesquisa por novas soluções se baseia na seleção e reprodução dos indivíduos, como se mostra na Figura 3.3 adiante.

Figura 3.3 –Processo de Mutação



FONTE: O autor (2018)

Mutação (clone modificado): copia o individuo da população anterior e modifica um pouco o cromossomo. Como existe várias formas de modificar o cromossomo, haverá diversos operadores de mutação. Para nosso exemplo, vamos supor

que a mutação ocorre em um dos genes do cromossomo, com pequena perturbação, com máximo de L para cima ou para baixo.

```
mutacao (pai)
        individuo = pai
        i = rand (N)
        individuo.x[ i ] = individuo.x[i] + 2*L*(drand -
           0.5)
        individuo.z = f(individuo.x)
        retorna individuo
fim mutacao
```

FONTE: (ROSA; LUZ, 2009)

3.5.9 Gerando uma nova população

No caso de *steady-state* , o individuo gerado entra na população somente quando for melhor que o anterior. Então, gera o individuo, avalia e insere se for melhor que o pior da população.

```
steady_state
para k = 1..NUM_MUTACAO
        i = rand (POP_SIZE)
        item = mutacao ( populacao [ i ] )
        se (item.z < populacao [POP_SIZE ].z )
                entao trocar ultimo por item e reordenar
        fim se
fim para

para k = 1..NUM_CRUZAMENTO
        seleciona_pais(i1 ,i2)
        item = cruzamento ( populacao [ i1 ] , populacao [
           i2 ] )
        se ( item.z < populacao [POP_SIZE].z )
                entao trocar ultimo por item e reordenar
        fim se
fim para
para k = 1..NUM_NOVOS
        item = novo_individuo
        se ( item.z < populacao [POP_SIZE ].z )
                entao trocar ultimo por item e reordenar
```

```
        fim se
fim para
```

FONTE: (ROSA; LUZ, 2009)

3.6 Otimização de Sistemas

Problemas de otimização são frequentemente encontrados em diferentes campos técnicos, como Engenharia, Economia, ou mesmo no dia-a-dia. Frequentemente, os problemas almejam objetivos contraditórios, relacionando custos, prazos, lucros, qualidade, eficiência e outras variáveis.

A formulação de um problema de otimização qualquer envolve a composição de uma função-objetivo, que relacione as diferentes variáveis consideradas, bem como as restrições impostas a cada uma. Otimizar um determinado problema consiste em identificar a solução, ou os respectivos valores para as variáveis consideradas, que maximize ou minimize o valor da função-objetivo, conforme a natureza do problema, de forma que nenhuma outra solução atribua um valor maior ou menor à função, respectivamente, respeitadas as restrições do problema (ROSA; LUZ, 2009).

3.7 Solução de problemas orientados a objetivos

Suponha que se tenha 10 chances de adivinhar um número entre 1 e 1.000 e a única informação que recebe é se sua resposta foi correta ou incorreta. Você poderia adivinhar o número com certeza? Apenas sabendo se sua resposta estava correta ou não, eu não teria como melhorá-la; Por isso, tem, na melhor das hipóteses, uma chance de 1 em 100 de adivinhar o número. Um aspecto fundamental da resolução de problemas usando algoritmos genéticos é que eles devem fornecer feedback que ajude o mecanismo a selecionar a melhor das duas hipóteses. Essa regeneração é chamada de *fitness* (em inglês). *Fitness* é uma palavra usada pela comunidade de programação genética. Seu valor é uma medida de quão perto a suposição se ajusta ao resultado desejado. Mais importante, implica uma progressão geral.

Se, em vez de saber se sua suposição está correta ou não, você for informado de que o número é maior ou menor do que o número da sua estimativa, você pode

sempre adivinhar o número. Isso ocorre porque 10 suposições são suficientes para busca binária para encontrar qualquer número no intervalo de 1 a 1000.

Agora imagine multiplicando o tamanho do problema para que em vez de tentar encontrar um número que são simultaneamente tentar encontrar um conjunto de 100 números, todos no intervalo de 1 a 1000. E você só recebe um valor de adequação que indica o quão próximo esse conjunto de números corresponde ao resultado desejado. Seu objetivo seria maximizar ou minimizar essa capacidade. De 100 números, você poderia encontrar o grupo correto de números? Sua possibilidade seria melhor, do que adivinhar aleatoriamente, se você tem conhecimento específico do problema que ajuda a eliminar certas combinações de números. O uso de conhecimento específico do problema para orientar a criação e modificação de soluções de algoritmos genéticos em potencial pode ajudá-lo a encontrar uma solução de ordem de magnitude mais rápida.

Algoritmos genéticos e programação genética são muito bons em encontrar soluções para problemas muito complexos. Eles fazem isso tirando milhões de amostras do espaço de pesquisa, fazendo pequenas alterações; possivelmente recombinar partes das melhores soluções, comparando a aptidão resultante com a da melhor solução atual e mantendo o melhor das duas soluções. Esse processo é repetido até que uma condição de parada ocorra como um dos seguintes: a solução conhecida é encontrada, é encontrada uma solução que atende a todos os requisitos, um certo número de gerações passou, uma quantidade específica de tempo, etcetera (MASSSAGO, 2013).

Capítulo 4

Materiais e Métodos

A seguir, são apresentados os materiais e métodos utilizados para a realização este trabalho. Entre os materiais, temos os dados diários obtidos a partir da leitura do pêndulo direto que mede os deslocamentos horizontais da barragem de Itaipu, resultando em uma série temporal; o software utilizado para aplicar a metodologia de Box & Jenkins; bem como as tecnologias utilizadas para a execução do algoritmo que permitiram obter previsões da série.

4.1 Materiais

4.1.1 Barragem de Itaipu

A barragem é a estrutura (concreto, enrocamento e terra) que serve para represar a água e obter o desnível de 120 m (queda bruta nominal) que permite a operação das turbinas. Na parte superior da barragem principal, estão situadas as tomadas de água, estruturas com comportas que permitem que a água, passando por elas e pelos condutos forçados, alcance a caixa espiral, onde faz a turbina girar. (ITAIPU, 2018a)

A barragem da Itaipu tem 7.919 metros de extensão e altura máxima de 196 metros, o equivalente a um prédio de 65 andares. Consumiu 12,3 milhões de metros cúbicos de concreto, enquanto o ferro e o aço utilizados permitiriam a construção de 380 Torres Eiffel, dimensões que transformaram a usina em referência nos estudos de concreto e na segurança de barragens

A barragem principal da Itaipu apresenta trechos dos tipos: gravidade maciça, de gravidade aliviada, de enrocamento e em arco, como se mostra na Figura 4.1.

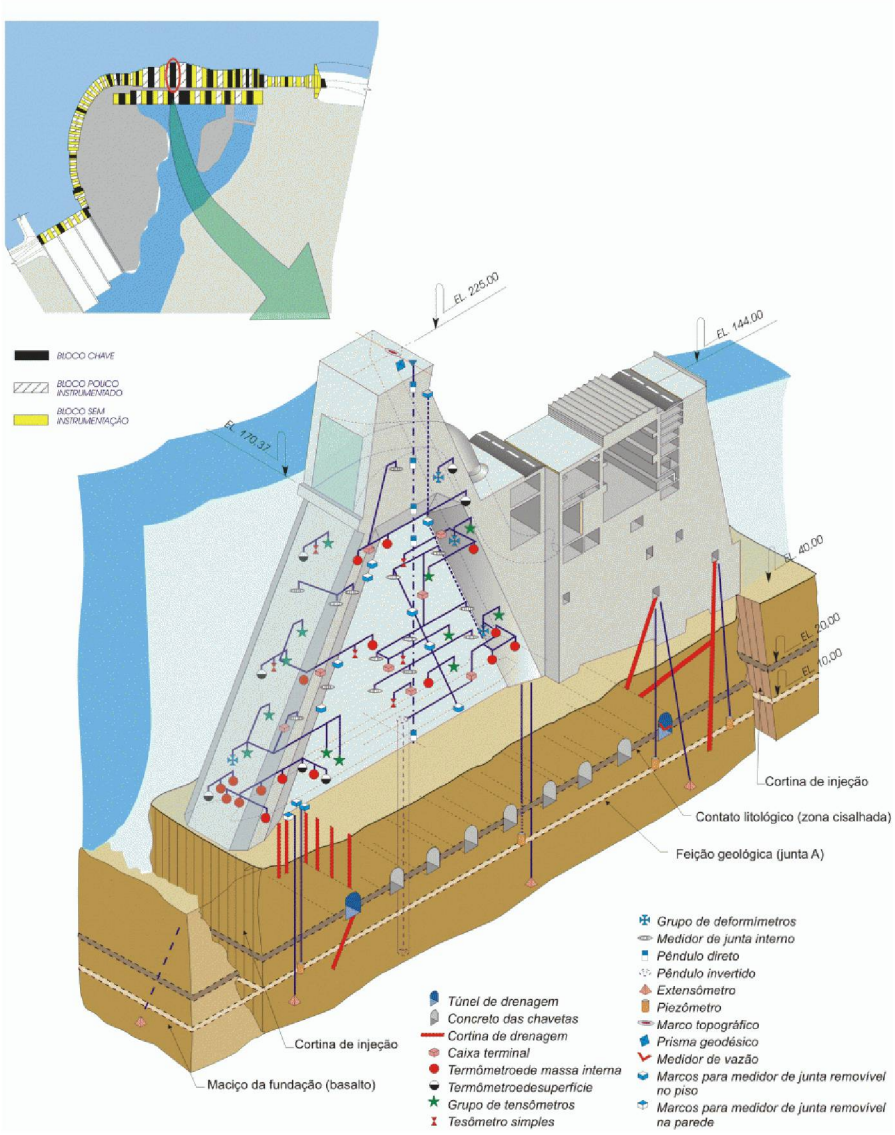
Figura 4.1 –Mapa de Barragens da Usina Hidrelétrica de Itaipu

FONTE: (ITAIPU, 2018a)

A Itaipu produz eletricidade com base na energia hidráulica, ou seja, pelo aproveitamento da energia potencial gravitacional da água contida em uma represa elevada. Esta energia está presente na natureza e pode ser aproveitada em desníveis acentuados ou quedas d'água. (ITAIPU, 2018b)

A usina de Itaipu, uma das maiores geradoras de hidroeletricidade do mundo, com mais de 2.200 instrumentos que monitoram seu comportamento geotécnico e estrutural, os quais possuem leituras armazenadas, como pode-se observar na Figura 4.2. Os técnicos da Itaipu têm o auxílio de 2.400 instrumentos (1.358 no concreto, 881 nas fundações e 161 para geodesia), sendo 270 automatizados, e 5.295 drenos (949 no concreto e 4.346 nas fundações) para acompanhar o desempenho das estruturas de concreto e fundações. (ITAIPU, 2018c)

Figura 4.2 –Instrumentos que monitoram as estruturas de concreto e fundação



FONTE: (ITAIPU, 2018c)

4.1.2 Linguagem de Programação e Bibliotecas

Statgraphics Centurion VI

Statgraphics Centurion XVI é uma ferramenta de análise de dados que combina uma ampla gama de procedimentos analíticos e funções estatísticas avançadas com gráficos interativos para fornecer um ambiente de análise integrado que pode ser aplicado em cada uma das fases de um projeto, desde os protocolos de gerenciamento Seis Sigma até processos de Controle de Qualidade (STATGRAPHICS.NET, 2018).

Python

Python é uma linguagem de programação criada por Guido van Rossum em 1991. Os objetivos do projeto da linguagem eram: produtividade e legibilidade. Em outras palavras, *Python* é uma linguagem que foi criada para produzir código bom e fácil de manter de maneira rápida. Entre as características da linguagem que ressaltam esses objetivos estão:

- baixo uso de caracteres especiais, o que torna a linguagem muito parecida com pseudo-código executável;
- o uso de identação para marcar blocos;
- coletor de lixo para gerenciar automaticamente o uso da memória.

Além disso, *Python* suporta múltiplos paradigmas de programação. A programação procedimental pode ser usada para programas simples e rápidos, mas estruturas de dados complexas, como tuplas, listas e dicionários, estão disponíveis para facilitar o desenvolvimento de algoritmos complexos. Grandes projetos podem ser feitos usando técnicas de orientação a objetos, que é completamente suportada em *Python* (inclusive sobrecarga de operadores e herança múltipla). Um suporte modesto para programação funcional existe, o que torna a linguagem extremamente expressiva: é fácil fazer muita coisa com poucas linhas de comando. Também possui inúmeras capacidades de meta-programação: técnicas simples para alterar o comportamento de comportamentos da linguagem, permitindo a criação de linguagens de domínio específico. (PYSCIENCE, 2017)

Scikit learn

Existem várias bibliotecas *Python* que fornecem implementações sólidas de uma variedade de algoritmos de aprendizado de máquina. Um dos mais conhecidos é o *Scikit-Learn* , um pacote que fornece versões eficientes de um grande número de algoritmos comuns. O *Scikit-Learn* é caracterizado por uma API limpa, uniforme e simplificada, bem como por uma documentação on-line muito útil e completa. Um benefício dessa uniformidade é que, uma vez que entende-se o uso básico e a sintaxe do *Scikit-Learn* para um tipo de modelo, mudar para um novo modelo ou algoritmo é muito simples. (VANDERPLAS, 2018)

DEAP

O *DEAP* é uma nova estrutura de computação evolucionária para prototipagem rápida e teste de ideias. Procura tornar os algoritmos explícitos e estruturas de dados transparentes. O *DEAP* funciona em perfeita harmonia com o mecanismo de paralelização, como o multiprocessamento e o *SCOOP* . A documentação a seguir apresenta os principais conceitos e muitos recursos para construir suas próprias evoluções. (DEAP, 2018)

O *DEAP* (Algoritmos Evolutivos Distribuídos em *Python*) é uma nova estrutura volumétrica de computação para prototipagem rápida e teste de ideias. Seu design se afasta da maioria dos outros frameworks existentes, na medida em que procura tornar os algoritmos explícitos e as estruturas de dados transparentes, ao contrário do tipo mais comum de frameworks de caixa preta. Ele também incorpora um fácil paralelismo em que os usuários não precisam se preocupar com detalhes cruciais da implementação, como sincronização e balanceamento de carga, apenas decomposição funcional. (RAINVILLE et al., 2012)

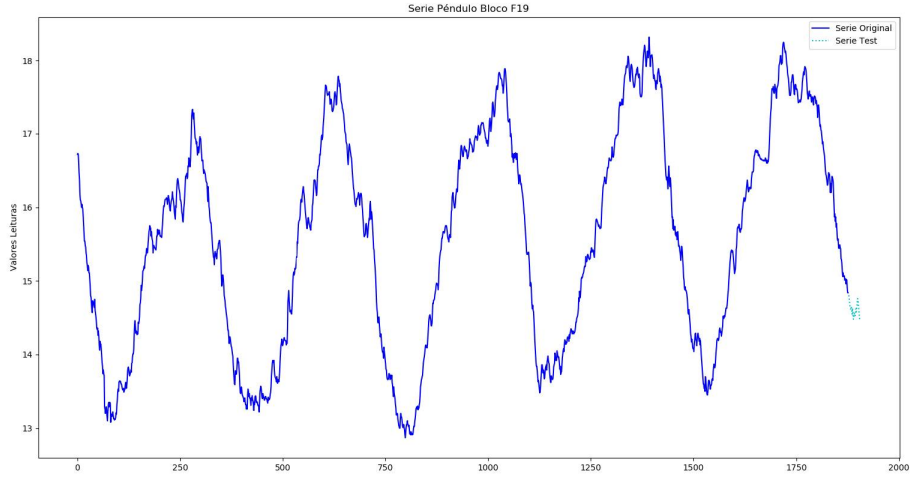
4.2 Metodologia

No trabalho, os dados utilizados foram obtidos do sistema ADAS (Automatic Data Acquisition System) que Itaipu dispõe desde 2005. Para a análise utilizam-se as leituras automatizadas dos sensores do pêndulo direto que representam os deslocamentos horizontais relativos ao bloco F19, na direção do eixo "x". O período de dados para a análise foi das datas 28 de outubro de 2005 ao 21 de fevereiro de 2011, sendo um total de 1909 observações, com uma periodicidade de uma leitura

por dia. Os deslocamentos foram medidos em milímetro.

Os dados foram inseridos no *software* de pacote estatístico *Statgraphics Centurion XVI* e analisados com la metodologia Box & Jenkins aplicando os modelos auto regressivos integrados de média móvel ARIMA, tendo como resultado a Série Temporal que pode-se observar no Gráfico 4.1. Do modelo ARIMA, os parâmetros p=2, d=2 e q=1, ou seja, ARIMA (2, 2, 1) foram os que melhor representam a Série Temporal.

Gráfico 4.1 –Série Temporal dos Deslocamentos Horizontais do bloco F19



FONTE: O autor (2018)

Logo, a partir do modelo ARIMA, a parte não linear (os resíduos) é utiliza como entrada para o Algoritmo Genético, num formato de arquivo Excel. Quando um problema é resolvido com um Algoritmo Genético, ao em vez de solicitar uma solução específica, são fornecidas características que a solução deve ter ou regras que a solução deve passar para ser aceita.

A metodologia proposta trará previsões do deslocamento horizontal apresentando as tendências do movimento que melhor se aproximam aos valores reais.

Ou seja, o trabalho tem duas etapas:

1. Ajustar o modelo Box & Jenkins: Quando se ajusta o modelo linear do B&J, tira-se a parte linear, mas nos resíduos permanece a parte não linear.

2. Dos resíduos da série temporal, serão aplicados algoritmos genéticos para obter o valor de previsão dos resíduos.

O resultado final será a soma dos resultados da aplicação da metodologia de B&J na série temporal e dos resultados da aplicação de algoritmos genéticos nos resíduos da série temporal.

4.3 Implementação

O trabalho foi desenvolvido utilizando o paradigma de Algoritmos Genéticos para predizer resultados dos resíduos da série original obtida aplicando o método de predição de B&J, provando sua viabilidade de uso também nesta área, sendo que o valor previsto deva possuir o menor erro possível em relação ao valor real.

A maneira pela qual o uso do algoritmo genético foi aplicado foi primeiro, começamos com uma sequência inicial de valores gerados aleatoriamente, em seguida, selecionando / cruzando um valor aleatório de cada vez até que a sequência de valores convergisse para o resultado esperado.

A seguir se faz un desglose tendo em conta cada elemento necessário para a programação do código do Algoritmo Genético (ver Apêndice .1).:

4.3.1 Gen

Para começar, o algoritmo genético precisa de um conjunto de genes para criar conjecturas. Para este projeto, foi um conjunto genérico de dados numéricos.

4.3.2 Gerar uma conjectura, Cromossoma

Em seguida, o algoritmo precisa de uma maneira gerar um vetor aleatório a partir do conjunto de genes.

Na técnica de Algoritmos Genéticos, a população inicial, isto é, da primeira geração é constituída por 50 (cinquenta) indivíduos criados de forma aleatória. Cada indivíduo constitui-se como um vetor onde em cada gene tem-se o valor do resíduo num total de 1907 (mil novecentos sete) parâmetros. Os parâmetros relativos a cada um dos resíduos são colocados num vetor de tamanho 1907 (mil novecentos sete elementos), sendo que, em termos biológicos, a cada elemento corresponde um gene e à designação de vetor corresponde a de cromossomo.

O tamanho da população é de cinquenta indivíduos em um total de 1000 gerações, isso devido a maior a geração e menor o erro.

Os valores foram obtidos com a combinação de probabilidades baixas de cruzamentos e probabilidade nula de mutação devido ao valor máximo e mínimo com que os indivíduos são gerados nas gerações em relação ao *fitness*.

4.3.3 Random.sample

Pega os valores numéricos solicitados da entrada sem substituição. Isso significa que não haverá duplicatas no pai gerado, a menos que o conjunto de genes contenha duplicatas ou o tamanho solicitado seja maior que *len* (conjunto de genes).

4.3.4 Seleção

A estratégia de seleção utilizada foir a *selWorst* da livraria *DEAP*, onde é escolhido o individuo com o menor valor de *fitness*, sendo o *fitness* a representação do erro RMSE entre o cromossoma gerado e o vetor de resíduos. O erro representa quanto o individuo e mais parecido com o modelo, neste caso, o vetor de resíduos.

4.3.5 Aptidão

O valor de aptidão fornecido pelo algoritmo genético é o único *feedback* que o algoritmo obtém para guiá-lo em direção a uma solução.

Uma boa escolha da função de aptidão pode ser responsável pelo bom funcionamento do AG, especificamente no caso de Séries Temporais, pode-se utilizar como função de aptidão a função que mede o erro calculado entre o valor previsto e o valor real.

Para comparar o desempenho obtido pelo algoritmo proposto com respeito ao ARIMA, foi utilizada a medidas estatística do Erro Quadrático Médio.

Então, como exemplo, no erro quadrático médio quanto menor for o erro obtido, melhor será o ajuste do modelo de previsão. O que se deseja, portanto, é minimizar a função de aptidão ou função objetivo.

Do total de 1907 valores dos resíduos resultantes da aplicação do B&J, foram divididos em dois segmentos, um para o treinamento, sendo constituído por 1877 valores, e outro segmento para teste, constituídos pelos 30 valores restantes. Os 30 valores guardados para teste foram comparados com os últimos 30 valores dos 1907 genes do melhor individuo da última geração.

A previsão é efetuada 30 passos à frente.

Portanto, tem-se a determinação de um máximo de 50 soluções em cada geração e cada qual é expressa como um indivíduo, resultante da evolução de uma população inicial e constante de 50 indivíduos ao longo de 1000 gerações. Nas séries de evolução, os melhores indivíduos de cada execução do Algoritmo Genético obtiveram sempre resultados com erros menores.

Capítulo 5

Resultados e Discussões

Apos geradas 1.000 generações, os 30 valores dos últimos genes dos 1907 valores gerados na última iteração são comparados com os resultados da metodología B&J.

A seguir, na Tabela 5.1 pode-se observar os 30 valores da última geração, comparados com os resultados da metodologia B&J.

Tabela 5.1 –Valores gerados do Bloco F13.

Valor Nro.	Valor Real	Metodologia B & J	Modelo Híbrido
1	14,37	14,4073	14,38235
2	14,38	14,3778	14,40274
3	14,34	14,3484	14,36285
4	14,28	14,3189	14,29465
5	14,22	14,2895	14,24058
6	14,23	14,2601	14,27584
7	14,21	14,2306	14,22958
8	14,24	14,2012	14,2185
9	14,20	14,1717	14,1698
10	14,14	14,1423	14,1502
11	14,21	14,1129	14,1684
12	14,13	14,0834	14,10835
13	14,06	14,0540	14,0885
14	14,08	14,0245	14,0613
15	14,10	13,9951	14,0823
16	14,10	13,9657	14,0737

17	14,09	13,9362	14,0825
18	14,13	13,9068	13,9535
19	14,18	13,8773	14,0138
20	14,16	13,8479	14,0835
21	14,22	13,8185	13,9571
22	14,24	13,7890	14,0235
23	14,33	13,7596	14,2514
24	14,33	13,7301	14,0291
25	14,28	13,7007	14,0823
26	14,22	13,6712	13,9834
27	14,14	13,6418	14,0255
28	14,08	13,6124	13,0823
29	14,05	13,5829	13,8523
30	14,06	13,5535	14,0035

FONTE: O autor (2018)

Na Tabela 5.2 pode-se observar os erros para cada grupo de dados, os de treinamento e os de teste, tanto da metodologia B&J como do modelo hibrido, dando como resultado uma mínima diferença de erro. Note-se que na fase de treinamento a Metodología B&J apresenta menor erro, mas na fase de teste o Modelo Híbrido apresenta uma melhora significativa.

Tabela 5.2 –Erros RMSE de treinamento e teste.

Grupo	Metodologia B & J	Modelo Híbrido
Treinamento	0,064290	0,08235
Teste	0,313363	0,218736

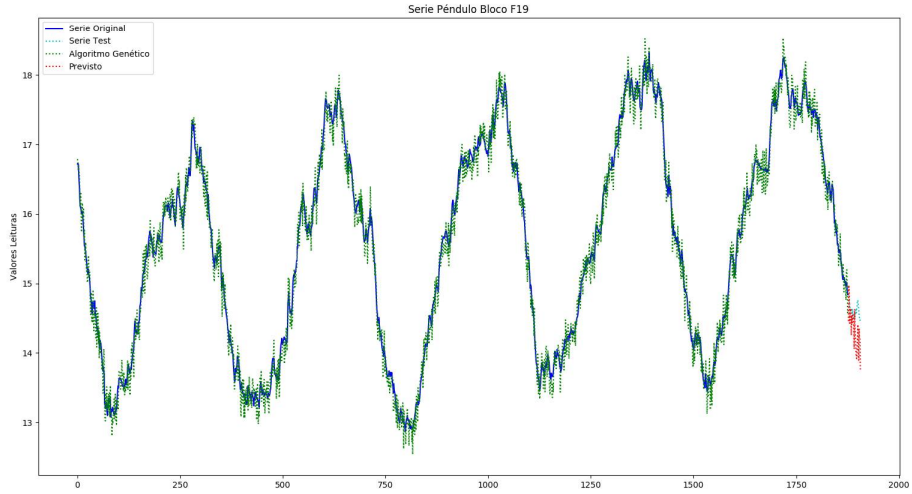
FONTE: O autor (2018)

Nos Gráficos 5.1 e 5.2 a seguir foram feitas as comparações dos resultados obtidos pelo método Algoritmos Genéticos para as previsões realizadas com o método de previsão utilizado metodologia Box & Jenkins.

Como se observa no Gráfico 5.1, os valores da melhor geração que representa a série (em verde), se aproximam da série original (em azul) na fase de Treinamento.

Os valores gerados pelo Modelo Híbrido oscilam em torno aos valores da Série Original, tendo um erro próximo do erro gerado pelo Modelo ARIMA.

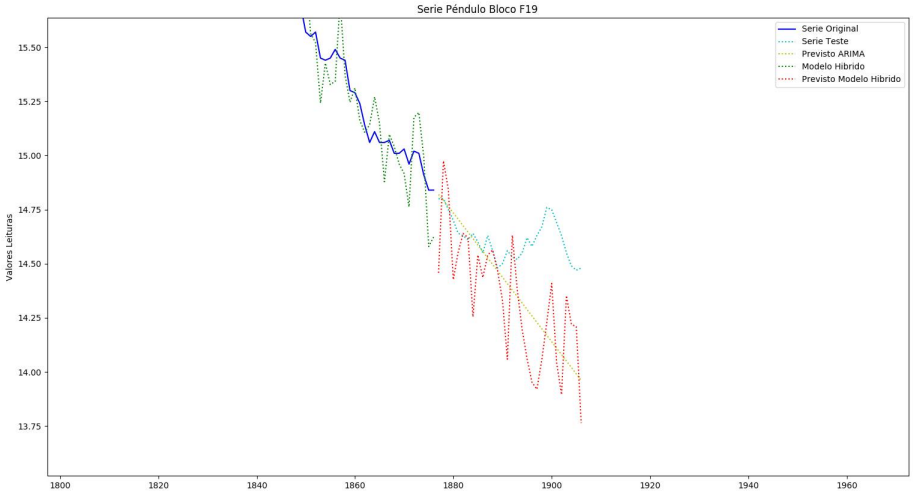
Gráfico 5.1 –Comparação do Modelo Híbrido com a Série Temporal



FONTE: O autor (2018)

No Gráfico 5.2 pode-se observar que os valores previstos pelo Modelo Híbrido (em vermelho) se aproximam dos valores reais da série (em azul claro) em comparação dos valores previstos pelo Modelo ARIMA (em amarelo), sendo a melhora verificada na Tabela 5.2, aonde se observa que o Modelo Híbrido teve um resultado com diminuição do erro em 30%.

Gráfico 5.2 –Comparação das previsões dos valores estimados.



FONTE: O autor (2018)

A partir da realização destes procedimentos, conclui-se que o Modelo Híbrido aplicando a metodologia de Algoritmos Genéticos proposta mostrou-se bastante eficiente na tarefa de previsão devido a que os arranjos dos erros foram pequenos.

Os resultados obtidos foram comparados a outro método de previsão utilizado metodologia Box & Jenkins. Além de terem sido comparados os erros de previsão obtidos (RMSE).

Capítulo 6

Conclusão

Foram ajustado os resíduos de uma série temporal usando Algoritmos Genéticos para a obtenção de previsões mais precisas.

Logo após, foi realizada a aplicação de Algoritmos Genéticos aos resíduos dado o ajuste do modelo ARIMA à série temporal.

Foi codificado um algoritmo na linguagem de programação *Python* usando o as bibliotecas *DEAP* e *Scikit learn*.

Por último, foram comparados os resultados obtidos através da aplicação da metodologia proposta com os obtidos através da aplicação da metodologia Box & Jenkins.

Referências Bibliográficas

BENTO, E. P.; KAGAN, N. *Algoritmos genéticos e variantes na solução de problemas de configuração de redes de distribuição*. 2008.

CORRÊA, M. C. F.; PACHECO, M. A. C.; SOUZA, R. C. Estimação de hiperparâmetros para um modelo de previsão holt-winters com múltiplos ciclos por algoritmos genéticos. *Revista de Inteligência Computacional Aplicada (RICA)*, Rio de Janeiro, 2001.

DEAP. *Documentação do DEAP*. 2018. Disponível em: <https://deap.readthedocs.io/en/master/>. Acesso em: 01 out. 2018.

EHLERS, R. *Análise de Séries Temporais. Departamento de Estatística, UFPR*. 2005. Disponível em: <http://www.each.usp.br/rvicente/AnaliseDeSeriesTemporais.pdf>. Acesso em: 29 set. 2018.

GARCÍA, M. et al. Métodos para predecir índices bursátiles. *Ecos de Economía*, Medellin, n. 37, p. 51–82, 2017.

GUTIÉRREZ D. J. M. C. CANALES, D. A. H. D. J. G. *Análisis e Implementación de Algortimos Evolutivos para la Optimización de Simulaciones en Ingeniería Civil. Monografia em Informática*. Murcia: [s.n.], 2017.

ITAIPU. *Itaipu*. 2018. Disponível em: <https://www.itaipu.gov.br/energia/barragem>. Acesso em: 26 set. 2018.

ITAIPU. *Itaipu*. 2018. Disponível em: <https://www.itaipu.gov.br/energia/barragem>. Acesso em: 26 set. 2018.

ITAIPU. *Itaipu*. 2018. Disponível em: <https://www.itaipu.gov.br/energia/barragem>. Acesso em: 26 set. 2018.

LINDEN, R. *Algoritmos Genéticos*. 3. ed. Rio de Janeiro: Editora Ciência Moderna, 2012. ISBN 978-85-399-205-7.

MARQUES, F. C. R.; GOMES., R. M. *Análise de Séries "Temporais Aplicadas ao Mercado Financeiro com o uso de Algoritmos Genéticos e Lógica Nebulosa". Laboratório de Bancos de Dados*. 2009. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/enia/2009/013.pdf/>. Acesso em: 24 jan. 2018.

MARQUES, I. S. *Predição de Séries Temporais utilizando Algoritmos Genéticos. Tese. Universidade Federal do Rio Grande do Sul, Porto Alegre, 2012.* Dissertação (Mestrado) — Universidade Federal do Rio Grande do Sul, Porto Alegre, 2012. Disponível em: <https://lume.ufrgs.br/bitstream/handle/10183/71088/000877909.pdf?sequence=1&isAllowed=y>. Acesso em: 22 ago. 2018.

MASSSAGO, S. *Introdução ao Algoritmo Genético*. 2013. Disponível em: <https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=2ahUKEwj2wM6jhfLeAhWMIpAKHfpRAzgQFjAAegQICRAC&url=https%3A%2F%2Fwww.dm.ufscar.br%2F~sadao%2Fdownload%2F%3Ffile%3Darticle%2Falgoritmos-geneticos.pdf&usg=AOvVaw2CaKvS1eoDeS1qQs2wI4Tj>. Acesso em: 10 nov. 2018.

MIRANDA, M. N. *Notas de aula do GINAPE do projeto em Desenvolvimento: Vida Artificial - Utilização dos conceitos de vida artificial no ensino exploratório de Ciências. Algoritmos Genéticos: Fundamentos e Aplicações. Universidade Federal do Rio de Janeiro*. Rio de Janeiro: [s.n.], 2017. Disponível em: <http://www.nce.ufrj.br/GINAPE/VIDA/alggenet.htm>. Acesso em: 02 fev. 2018.

NETO, A. C. *Notas de aula da disciplina MNE700 – Análise de Séries Temporais. Universidade Federal do Paraná. Curitiba, 2017*. 2017.

PYSCIENCE, B. *Python: O que é? Por que usar?* 2017. Disponível em: <http://pyscience-brasil.wikidot.com/python:python-oq-e-pq>. Acesso em: 01 out. 2018.

RAINVILLE, F.-M. D. et al. *DEAP: A Python framework for Evolutionary Algorithms*. 2012. 85-92 p. Disponível em: <https://www.researchgate.net/publication/235707002_DEAP_A_Python_framework_for_Evolutionary_Algorithms>. Acesso em: 01 out. 2018.

REIS, M. M. *Notas de aula da disciplina INE 7001 - Análise de Séries Temporais*. 2018. Disponível em: <http://www.inf.ufsc.br/~marcelo.menezes.reis/Cap4.pdf>. Acesso em: 10 fev. 2018.

ROSA, T. O.; LUZ, H. S. Conceitos básicos de algoritmos genéticos: Teoria e prática. *Anais ULBRA*, Canoas, 2009.

SOUZA, D. O. *Algoritmos Genéticos aplicados ao Planejamento do Transporte Principal de Madeira*. Dissertação (Mestrado) — Universidade Federal do Paraná, Curitiba, 2004.

STATGRAPHICS.NET. *Características do Statgraphics*. 2018. Disponível em: <https://statgraphics.net/caracteristicas/>. Acesso em: 01 out. 2018.

TEIXEIRA, L. L.; ET.AL. *Revisão de Séries Temporais por meio de um Método Híbrido Wavelet-Neural Integrado com Bootstrap" em Métodos Numéricos Aplicados a Análise de Segurança de Barragens, Foz do Iguaçu, Ed. Parque Itaipu*. 2017.

VANDERPLAS, J. *Apresentando o Scikit-Learn*. 2018. Disponível em: <https://jakevdp.github.io/PythonDataScienceHandbook/05.02-introducing-scikit-learn.html>. Acesso em: 01 out. 2018.

Apêndice

.1 Código Fonte

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Jul 31 18:40:02 2018

@author: lucia
"""
import pandas
import itertools
import random
import numpy
from deap import base, creator, algorithms, tools
from sklearn.metrics import mean_squared_error
from math import sqrt
import matplotlib.pyplot as plt
from operator import itemgetter
from openpyxl import load_workbook

xlsx_file = 'datos_completo.xlsx'
xlsx = pandas.ExcelFile(xlsx_file)
# folhas = xlsx.sheet_names

folhas = ['F13']
CANT_DADOS = 1907
CANT_PREV = 30

# CANT_DADOS = 190
```

```
# CANT_PREV = 10

CANT_TREINO = CANT_DADOS - CANT_PREV
dataframe = []
indvs = []

def avaliacao(individual):
    # print('Modelo')
    # print(modelo)
    # print('Individuo')
    # print(individual)

    rmse = 0.0 # para el error cuadratico medio
    rmse = sqrt(mean_squared_error(modelo, individual[:
        CANT_TREINO]))
    indvs.append([rmse,individual])

    # print(rmse)

    # for i in range(len(individual)):
    #     print(modelo[i])
    #     print(individual[i])

    return rmse,

def varAnd2(population, toolbox, cxpb, mutpb):
    """Part of an evolutionary algorithm applying only the
        variation part
    (crossover **and** mutation). The modified individuals
        have their
    fitness invalidated. The individuals are cloned so
        returned population is
    independent of the input population.
    :param population: A list of individuals to vary.
    :param toolbox: A :class:`~deap.base.Toolbox` that
        contains the evolution
                    operators.
    :param cxpb: The probability of mating two individuals.
    :param mutpb: The probability of mutating an individual
        .
```

```
:returns: A list of varied individuals that are
    independent of their
          parents.
The variation goes as follow. First, the parental
    population
:math:`P_\mathrm{p}` is duplicated using the :meth:`
    toolbox.clone` method
and the result is put into the offspring population :
    math:`P_\mathrm{o}`. A
first loop over :math:`P_\mathrm{o}` is executed to
    mate pairs of
consecutive individuals. According to the crossover
    probability *cxpb*, the
individuals :math:`\mathbf{x}_i` and :math:`\mathbf{x}_
    {i+1}` are mated
using the :meth:`toolbox.mate` method. The resulting
    children
:math:`\mathbf{y}_i` and :math:`\mathbf{y}_{i+1}`
    replace their respective
parents in :math:`P_\mathrm{o}`. A second loop over the
     resulting
:math:`P_\mathrm{o}` is executed to mutate every
    individual with a
probability *mutpb*. When an individual is mutated it
    replaces its not
mutated version in :math:`P_\mathrm{o}`. The resulting
    :math:`P_\mathrm{o}`
is returned.
This variation is named *And* beceause of its
    propention to apply both
crossover and mutation on the individuals. Note that
    both operators are
not applied systematicaly, the resulting individuals
    can be generated from
crossover only, mutation only, crossover and mutation,
    and reproduction
according to the given probabilities. Both
    probabilities should be in
:math:`[0, 1]`.
"""
```

```
offspring = [toolbox.clone(ind) for ind in population]

p1 = offspring

# print("Offspring Antes")
# for p in offspring:
#     print(p.fitness.values,p)
    # cross_arch.write(str(p.fitness.values)+"\n")

corte = int(len(offspring)*.50) # Elegir el mejor 30%
# print("Corte: ",corte)
# print(type(corte))

selected = [toolbox.clone(ind) for ind in population[:
    corte]]
# print("\nSeleccion")
# for p in selected:
#     print(p.fitness.values, p)

# Se mezcla el material genetico para crear nuevos
    individuos
# for i in range(corte, len(offspring)):
i = corte
while (i < len(offspring)):
    # print('i: '+str(i))
    # punto2 = random.randint(1, len(offspring) - 1)  #
        Se elige un punto para hacer el intercambio

    cp = 99
    while True:
        padre = random.sample(selected, 2)  # Se eligen
            dos padres
        cp = cmp(padre[0], padre[1])

        if(cp != 0 ):
            break

    # print('Padres')
    #
```

```
        # print(cp)
        # if(cp==0):
        #     for x in padre:
        #         print(x)
        #     break

        # for j in range(corte, len(offspring)-1):
        while True:
            punto = random.randint(1, len(offspring) - 1)
                # Se elige un punto para hacer el
                intercambio
            # print(punto)

            offspring[i][:punto] = padre[0][:punto]  # Se
                mezcla el material genetico de los padres en
                 cada nuevo individuo
            offspring[i][punto:] = padre[1][punto:]

            # print(offspring[i])

            del offspring[i].fitness.values
            sw = 99
            for j in range(corte, i):
                # print('i:'+str(i)+' j: ' + str(j))
                sw = cmp(offspring[j], offspring[i])

                if (sw == 0):
                    break

            if(sw != 0):
                break
            # else:
            #     print('repetido')

        i += 1

    for i in range(len(offspring)):
        if random.random() < mutpb:
            # print("muto")
            offspring[i], = toolbox.mutate(offspring[i])
```

```
            del offspring[i].fitness.values

    # print(len(offspring))
    # print("Offspring Despues")
    # for p in offspring:
    #     print(p.fitness.values, p)

    p2 = offspring


    # for i in range(len(offspring)):
    #     d = list(set(p1[i]).symmetric_difference(set(p2[i
        ])))
    #     print(d)


    return offspring

def eaSimple2(population, toolbox, cxpb, mutpb, ngen, stats
    =None,
              halloffame=None, verbose=__debug__):
    """This algorithm reproduce the simplest evolutionary
       algorithm as
    presented in chapter 7 of [Back2000]_.
    :param population: A list of individuals.
    :param toolbox: A :class:`~deap.base.Toolbox` that
       contains the evolution
                    operators.
    :param cxpb: The probability of mating two individuals.
    :param mutpb: The probability of mutating an individual
       .
    :param ngen: The number of generation.
    :param stats: A :class:`~deap.tools.Statistics` object
       that is updated
                  inplace, optional.
    :param halloffame: A :class:`~deap.tools.HallOfFame`
       object that will
                       contain the best individuals,
                          optional.
    :param verbose: Whether or not to log the statistics.
```

```
:returns: The final population
:returns: A class:‘~deap.tools.Logbook‘ with the
    statistics of the
          evolution
The algorithm takes in a population and evolves it in
    place using the
:meth:‘varAnd‘ method. It returns the optimized
    population and a
:class:‘~deap.tools.Logbook‘ with the statistics of the
     evolution. The
logbook will contain the generation number, the number
    of evalutions for
each generation and the statistics if a :class:‘~deap.
    tools.Statistics‘ is
given as argument. The *cxpb* and *mutpb* arguments are
     passed to the
:func:‘varAnd‘ function. The pseudocode goes as follow
    ::
     evaluate(population)
     for g in range(ngen):
          population = select(population, len(population)
              )
          offspring = varAnd(population, toolbox, cxpb,
              mutpb)
          evaluate(offspring)
          population = o(0.16631157298322366,)ffspring
As stated in the pseudocode above, the algorithm goes
    as follow. First, it
evaluates the individuals with an invalid fitness.
    Second, it enters the
generational loop where the selection procedure is
    applied to entirely
replace the parental population. The 1:1 replacement
    ratio of this
algorithm **requires** the selection procedure to be
    stochastic and to
select multiple times the same individual, for example,
:func:‘~deap.tools.selTournament‘ and :func:‘~deap.
    tools.selRoulette‘.
```

```
    Third, it applies the :func:`varAnd` function to
       produce the next
    generation population. Fourth, it evaluates the new
       individuals and
    compute the statistics on this population. Finally,
       when *ngen*
    generations are done, the algorithm returns a tuple
       with the final
    population and a :class:`~deap.tools.Logbook` of the
       evolution.
    .. note::
        Using a non-stochastic selection method will result
            in no selection as
        the operator selects *n* individuals from a pool of
            *n*.
    This function expects the :meth:`toolbox.mate`, :meth:`
       toolbox.mutate`,
    :meth:`toolbox.select` and :meth:`toolbox.evaluate`
       aliases to be
    registered in the toolbox.
    .. [Back2000] Back, Fogel and Michalewicz, "
       Evolutionary Computation 1 :
       Basic Algorithms and Operators", 2000.
    """
    logbook = tools.Logbook()
    logbook.header = ['gen', 'nevals'] + (stats.fields if
       stats else [])

    # Evaluate the individuals with an invalid fitness
    invalid_ind = [ind for ind in population if not ind.
       fitness.valid]
    fitnesses = toolbox.map(toolbox.evaluate, invalid_ind)
    for ind, fit in zip(invalid_ind, fitnesses):
        ind.fitness.values = fit

    if halloffame is not None:
        halloffame.update(population)

    record = stats.compile(population) if stats else {}
```

```
    logbook.record(gen=0, nevals=len(invalid_ind), **record
        )
    if verbose:
        print(logbook.stream)

    # Begin the generational process
    for gen in range(1, ngen + 1):
        # print("Poblacion Antes")
        #
        # for p in population:
        #     print(p.fitness.values,p)

        # Select the next generation individuals
        offspring = toolbox.select(population, len(
            population))


        # Vary the pool of individuals

        # **********    Trocada a funcao nesta parte
            **********
        offspring = varAnd2(offspring, toolbox, cxpb, mutpb
            )


        # Evaluate the individuals with an invalid fitness
        invalid_ind = [ind for ind in offspring if not ind.
            fitness.valid]
        fitnesses = toolbox.map(toolbox.evaluate,
            invalid_ind)
        for ind, fit in zip(invalid_ind, fitnesses):
            ind.fitness.values = fit

        # Update the hall of fame with the generated
            individuals
        if halloffame is not None:
            halloffame.update(offspring)

        # Replace the current population by the offspring
        population[:] = offspring
```

```
        # print("Siguiente Poblacion")
        # for p in population:
        #     print(p.fitness.values, p)

        # Append the current generation statistics to the
            logbook
        record = stats.compile(population) if stats else {}
        logbook.record(gen=gen, nevals=len(invalid_ind), **
            record)
        if verbose:
            print(logbook.stream)

    # cross_arch.close()
    # cross_arch2.close()
    # cross_arch3.close()
    return population, logbook


if __name__ == "__main__":
    for f in folhas:
        # print(f)
        # sheet_to_df_map[sheet_name] = xls.parse(
            sheet_name)
        dataframe = pandas.read_excel(xlsx_file, sheet_name
            =f, header=0, usecols=[0, 1, 2], nrows=
            CANT_DADOS)

        # print(dataframe.head())

        serie = dataframe['Real'][:CANT_TREINO].values.
            tolist()
        serie = pandas.to_numeric(serie).astype('float32')

        serie_test = dataframe['Real'][CANT_TREINO:].values
            .tolist()
        serie_test = pandas.to_numeric(serie_test).astype('
            float32')
```

```
arima = dataframe['ARIMA'][:CANT_TREINO].values.
    tolist()
arima = pandas.to_numeric(arima).astype('float32')

arima_test = dataframe['ARIMA'][CANT_TREINO:].
    values.tolist()
arima_test = pandas.to_numeric(arima_test).astype('
    float32')

residuos_arima = dataframe['Residuo'][:CANT_TREINO
    ].values.tolist()
residuos_arima = pandas.to_numeric(residuos_arima).
    astype('float32')

# CANT_DADOS = len(serie)

# modelo = residuos_arima[:10]
modelo = residuos_arima
# print('Modelo')
# print(modelo)

max_datos = numpy.amax(modelo)
min_datos = numpy.amin(modelo)

# plt.plot(datos)
# plt.show()

# modelo = [1,1,1,1,1,1,1,1,1,1] #Objetivo a
    alcanzar
#
#
# largo = 10 #La longitud del material genetico de
    cada individuo
# num = 10 #La cantidad de individuos que habra en
    la poblacion
# pressure = 3 #Cuantos individuos se seleccionan
    para reproduccion. Necesariamente mayor que 2
# mutation_chance = 0.2 #La probabilidad de que un
    individuo mute
#
```

```
# print("\n\nModelo: %s\n"%(modelo)) #Mostrar el
    modelo, con un poco de espaciado

toolbox = base.Toolbox()
creator.create("FitnessMax", base.Fitness, weights
    =(1.0,))
creator.create("Individual", list, fitness=creator.
    FitnessMax)
# toolbox.register("leitura", random.uniform
    ,-1.0,1.0)
toolbox.register("leitura", random.uniform,
    min_datos, max_datos)
toolbox.register("individual", tools.initRepeat,
    creator.Individual,
                     toolbox.leitura, n=len(modelo) +
                        CANT_PREV)
toolbox.register("population", tools.initRepeat,
    list, toolbox.individual)

indvs = []
# cross_arch = open("poblacion.txt","w")
# cross_arch2 = open("ordenados.txt","w")
# cross_arch3 = open("cruzados.txt","w")

toolbox.register("evaluate", avaliacao)
toolbox.register("mate", tools.cxOnePoint)
toolbox.register("mutate", tools.mutFlipBit, indpb
    =0.0)
# toolbox.register("select", tools.selRoulette)
toolbox.register("select", tools.selWorst)

# random.seed(1)
populacao = toolbox.population(n=50) # cantidad de
    invididuos por poblacion = 10
probabilidade_crossover = 1.0
probabilidade_mutacao = 0.0
numero_geracoes = 15

logbook = tools.Logbook()
```

```
estatisticas = tools.Statistics(key=lambda
    individuo: individuo.fitness.values)
estatisticas.register("max", numpy.max)
estatisticas.register("min", numpy.min)
estatisticas.register("med", numpy.mean)
estatisticas.register("std", numpy.std)

# populacao, info = algorithms.eaSimple(populacao,
    toolbox,
#
    probabilidade_crossover,
#
    probabilidade_mutacao,
#
    numero_geracoes, estatisticas)

populacao, info = eaSimple2(populacao, toolbox,
                            probabilidade_crossover
                                ,
                            probabilidade_mutacao
                                ,
                            numero_geracoes,
                                estatisticas,
                                verbose=True)

melhores = tools.selWorst(populacao, 1)
# print(len(melhores[0]))

# for individuo in melhores:
#     print('Individuo loop')
#     print(individuo)
#     print('Fitness')
#     print(individuo.fitness)

# valores_grafico = info.select("min")
# plt.plot(valores_grafico)
# plt.title("Acompanhamento dos valores")
# plt.show()

# for i in melhores[0]:
```

```
#       print(i)


testado = numpy.add(arima, melhores[0][:CANT_TREINO
    ])
# print(previsto)

previsto = numpy.add(arima_test, melhores[0][
    CANT_TREINO:])
# print("Previsto")
# for i in previsto:
#       print(i)

plt.title("Serie Pendulo Bloco "+f)
plt.ylabel('Valores Leituras')
plt.plot(serie, 'b-', label='Serie Original')

vacio = numpy.empty(len(serie))
vacio.fill(numpy.nan)

serie_test_plot = numpy.insert(serie_test,0,vacio)
plt.plot(serie_test_plot, 'c:', label='Serie Test')

plt.plot(testado, 'g:', label='Algoritmo Genetico')

previsto_plot = numpy.insert(previsto, 0, vacio)
plt.plot(previsto_plot, 'r:', label='Previsto')
plt.legend()
plt.show()

# print(len(melhores[0]))
# rmse2 = sqrt(mean_squared_error(modelo, melhores
    [0][:100]))

# print('Modelo '+str(len(modelo)))
# for m in modelo:
#       print(m)
# # print(modelo)
#
# print('Individuo '+str(len(melhores[0])))
```

```
        # for me in melhores[0]:
        #     print(me)
        # # print(melhores[0])
        # print(rmse2)

        # for i in indvs:
        #     print(i)

        # gener = info.select("gen")
        # pobl = info.select("nevals")
        #
        # k = 0
        # L = 0
        # for g in gener:
        #     print("Generacion ",g)
        #     L += pobl[g]
        #     while k < L:
        #         print(indvs[k])
        #         k+=1


# print('CANT. DADOS: '+str(CANT_DADOS))

xlsx_file2 = 'datos_completo2.xlsx'
writer = pandas.ExcelWriter(xlsx_file2)
for f in folhas:
    # print(f)
    # sheet_to_df_map[sheet_name] = xls.parse(
        sheet_name)
    dataframe = pandas.read_excel(xlsx_file, sheet_name
        =f, header = 0, usecols=[0,1,2], nrows=
        CANT_DADOS)

    # dataframe['Residuo Hibrido'] = '=B'+str(dataframe
        ['ARIMA'].index)
    dataframe['Residuo_Hibrido'] = melhores[0]
    dataframe['Previsto_Hibrido'] = dataframe['ARIMA']
        + dataframe['Residuo_Hibrido']
```

```
dataframe['Error_Statgraphics'] = numpy.square(
    dataframe['Real'] - dataframe['ARIMA'])
dataframe['Error_Hibrido'] = numpy.square(dataframe
    ['Real'] - dataframe['Previsto_Hibrido'])

RMSE_ARIMA_TREINO = sqrt(numpy.mean(dataframe.loc[:
    CANT_TREINO, 'Error_Statgraphics']))
RMSE_ARIMA_TESTING = sqrt(numpy.mean(dataframe.loc[
    CANT_TREINO:, 'Error_Statgraphics']))
RMSE_HIBRIDO_TREINO = sqrt(numpy.mean(dataframe.loc
    [:CANT_TREINO, 'Error_Hibrido']))
RMSE_HIBRIDO_TESTING = sqrt(numpy.mean(dataframe.
    loc[CANT_TREINO:, 'Error_Hibrido']))

# print(RMSE_ARIMA_TREINO)
# print(RMSE_HIBRIDO_TREINO)
#
# print(RMSE_ARIMA_TESTING)
# print(RMSE_HIBRIDO_TESTING)


dataframe.to_string()

#insertar espacio
espac = pandas.DataFrame(
    {'Real': '', 'ARIMA': '', 'Residuo': '', '
       Residuo_Hibrido': '', 'Previsto_Hibrido': ''
       ,
     'Error_Statgraphics': '', 'Error_Hibrido': ''
       }, index=[1876.5])

dat = pandas.DataFrame(
    {'Real': '', 'ARIMA': '', 'Residuo': '', '
       Residuo_Hibrido': '', 'Previsto_Hibrido': '
       RMSE_Treino',
     'Error_Statgraphics': str(RMSE_ARIMA_TREINO),
       'Error_Hibrido': str(RMSE_HIBRIDO_TREINO)},
       index=[1876.5])
```

```
        dataframe = dataframe.append(dat, ignore_index=
            False, sort=False)
        dataframe = dataframe.append(espac, ignore_index=
            False, sort=False)
        # print(dataframe.head())
        dataframe = dataframe.sort_index().reset_index(drop
            =True)
        # print(dataframe.head())

        dat = {'Real': '', 'ARIMA': '', 'Residuo': '', '
            Residuo_Hibrido': '', 'Previsto_Hibrido': 'RMSE_
            Previsto',
                'Error_Statgraphics': str(RMSE_ARIMA_TESTING
                    ), 'Error_Hibrido': str(
                    RMSE_HIBRIDO_TESTING)}

        dataframe = dataframe.append(dat, ignore_index=True
            )

        dataframe.to_excel(writer, sheet_name=f, index=
            False)
        # dataframe.to_excel(writer, sheet_name=f)
        # print(dataframe.head())
        # print(dataframe.tail())

    writer.save()
```