UNIVERSIDADE FEDERAL DE ALAGOAS - UFAL
INSTITUTO DE COMPUTAÇÃO - IC
CIÊNCIA DA COMPUTAÇÃO

JOÃO VICTOR RIBEIRO FERRO

Um Algoritmo Genético Auto Adaptável pela Fuzzificação da Taxa de Mutação

Maceió - AL
2021.2

JOÃO VICTOR RIBEIRO FERRO

Um Algoritmo Genético Auto Adaptável pela Fuzzificação da Taxa de Mutação

Trabalho de Conclusão de Curso submetido ao corpo docente do Instituto de Computação da Universidade Federal de Alagoas. Orientado pela Prof(a). Dra. Roberta Vilhena Vieira Lopes

Maceió - AL
2021.2

Catalogação na Fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico

Bibliotecário: Marcelino de Carvalho Freitas Neto – CRB-4 - 1767

F395a Ferro, João Victor Ribeiro.
Um algoritmo genético auto adaptável pela fuzzificação da taxa de mutação / João Victor Ribeiro Ferro. – 2021.
66 f. : il.

Orientadora: Roberta Vilhena Vieira Lopes.
Monografia (Trabalho de conclusão de curso em Ciência da Computação) – Universidade Federal de Alagoas, Instituto de Computação. Maceió.

Bibliografia: f. 57-61.
Apêndices: f. 62-66.

1. Algoritmos genéticos - Taxa de mutação. 2. Algoritmos genéticos - Seleção. 3. Algoritmos genéticos - Substituição. 4. Lógica difusa. I. Título.

CDU: 004.421

Folha de Aprovação

AUTOR: JOÃO VICTOR RIBEIRO FERRO

Um Algoritmo Genético Auto Adaptável pela Fuzzificação da Taxa de Mutação

Trabalho de Conclusão de Curso submetido ao corpo docente do Instituto de Computação da Universidade Federal de Alagoas e aprovado em 14 de Julho de 2022

_____

(Profa. Dra. Roberta Vilhena Vieira Lopes, UFAL) (Orientadora)

Banca Examinadora:

_____

(Prof. Dr. Evandro de Barros Costa, UFAL) (Examinador Interno)

_____

(Profa. Dra. Andrilene Ferreira Maciel - UFRPE) (Examinadora Externo)

Agradecimentos

Primeiramente a Deus por ter me dado forças para continuar e superar as dificuldades.

A minha mãe Antonia Cléria Ribeiro Ferro, ao meu pai João Carlos Ferro da Silva e a minha irmã Victória Camilly Ribeiro Ferro pelo incentivo, conselhos e apoio necessário para alcançar meus objetivos.

A UFAL e ao corpo docente do curso de Ciência da Computação por abrirem oportunidades únicas no meu crescimento profissional e em especial a minha orientadora Roberta Vilhena Vieira Lopes, pelo tempo dedicado, por sempre acreditar e me fazer acreditar na minha capacidade e também pelo suporte dado neste trabalho.

Aos professores Evandro de Barros Costa e Fábio Coutinho pela disponibilidade e empenho no desenvolvimento da minha trajetória acadêmica.

Aos meus companheiros de turma, por tornar essa jornada menos difícil e mais divertida, com as brincadeiras e horas de desestresse. Em especial José Rubens da Silva Brito, Lucas A. Lisboa, Matheus Gomes de Oliveira, Ramon Basto Callado, William Gabriel da Paz Rosendo.

E a todos que, diretamente ou indiretamente, fizeram parte da minha formação.

Resumo

Neste trabalho será apresentado uma variação de um algoritmo genético de Holland para o problema de otimização com o foco no ajuste do parâmetro da taxa de mutação por meio da fuzzificação da diversidade da população e do valor da adaptação do indivíduo e, em paralelo, compreender como é o comportamento dos métodos de seleção e substituição. Uma vez que esses parâmetros interferem diretamente na convergência e na qualidade da solução encontrada pelo algoritmo genético. Para avaliar o desempenho do algoritmo proposto foram realizados experimentos com problema de otimização combinatorial, em que foram analisados a convergência, a qualidade da solução encontrada, a diversidade da população e o número de indivíduos avaliados.

Palavras-chaves: Algoritmo Genético, Taxa de mutação, Lógica Fuzzy, Seleção, Substituição.

Abstract

In this work, a variation of Holland's genetic algorithm will be presented for the optimization problem, focusing on the adjustment of the mutation rate parameter, by means of the population diversity and the adaptation value of the individual and, in parallel, understanding how the selection and substitution methods behave. Since these parameters directly interfere with the convergence and quality of the solution found by the genetic algorithm. To evaluate the performance of the proposed algorithm, experiments were conducted with combinatorial optimization problems, in which the convergence, the quality of the solution found, the diversity of the population and the number of individuals evaluated were analyzed.

Keywords: Genetic Algorithm, Mutation Rate, Fuzzy Logic, Selection, Substitution.

Lista de Figuras

Lista de Gráficos

Lista de Quadros

Lista de Abreviaturas

AG - Algoritmo Genético

MinLT - Tempo de Vida Mínimo Permitido

MaxLT - Tempo de Vida Máximo Permitido

AvgFit - Valor de adaptação médio da População

SUMÁRIO

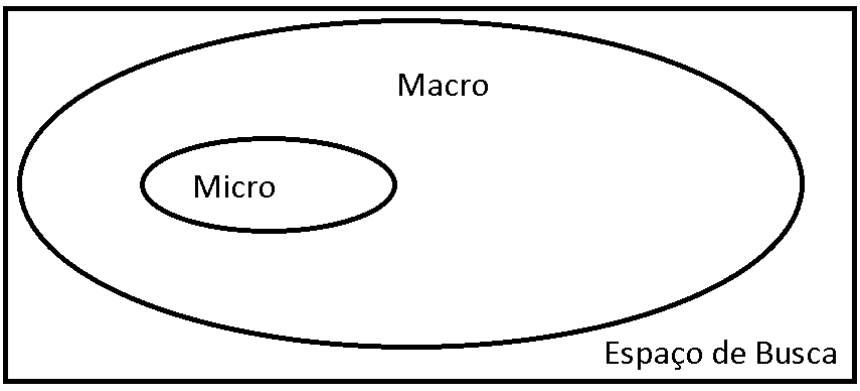
CAPÍTULO 1

Introdução

Os algoritmos genéticos simulam o processo de evolução das espécies descrito na teoria de DARWIN (2004) combinado aos conceitos de hereditariedade, seleção, cruzamento, mutação, da biologia genética (POZO,2005) . Com o objetivo de explorar o espaço de busca de problemas complexos atrás de uma solução ótima global, por meio da realização tanto de uma busca macro como de uma busca micro no espaço de investigação do problema.

A busca macro é realizada pelo operador genético de mutação para identificar os sub-espaços promissores, que são aqueles subespaços do espaço de busca cujas soluções têm melhor qualidade do que as soluções dos sub-espaços vizinhos. Já a busca micro realizada pelo cruzamento irá trabalhar nos sub-espaços promissores com a finalidade de encontrar as soluções óptimas locais de um sub-espaço promissor (Z. Michalewicz, 1996).

Figura 1.1 - Espaço de Busca Macro e Micro



Fonte: Autor

Como mostrado na Figura 1.1, para realizar a busca macro, os algoritmos genéticos fazem uso da taxa de mutação, responsável por determinar a quantidade de novos subespaços de busca a ser considerada na construção na próxima iteração do algoritmo. Encontrar o melhor valor para a taxa de mutação é um desafio, pois diferentes problemas podem exigir diferentes valores. Sendo a escolha da taxa de mutação determinada pelo programador que pode usar um valor padrão, um valor bem sucedido para um problema da mesma classe ou um valor obtido pelo método da tentativa-e-erro. Porém, todas estas formas de escolha da taxa de mutação podem gerar uma busca macro eficiente.

Uma busca macro é considerada eficiente quando a diversidade da população trabalhada pelo algoritmo genético é mantida durante toda a sua execução. Uma forma de manter a diversidade da população é fazer com que o valor da taxa de mutação seja sensível à diversidade da população, como resultado gerou-se os algoritmos auto ajustáveis ou auto adaptáveis.

Outra forma de manter a diversidade da população é através dos métodos de seleção e substituição que direcionam o algoritmo genético no processo de busca por subespaço de soluções mais promissoras. Enquanto o método de seleção identifica os indivíduos aptos a gerar descendentes. O método de substituição indica quais indivíduos devem ser mantidos, inseridos e retirados da população.

1.1 - Motivação e Justificativa do Estudo

Existe alguns estudos que avaliam o comportamento dos algoritmos genéticos variando a taxa de cruzamento e mutação como o de (PINHO *et al* , 2007), (CARVALHO, 2017), além desses estudos tem os que utilizam a Lógica Fuzzy como o de (BURDELIS, 2019), (BRITO,2011) e (FERRARI, 2014), onde é mostrado a importância da utilização da Lógica Fuzzy para manter a diversidade da população trabalhada pelo AG. Contudo, na maioria desses trabalhos não existe comparação entre o algoritmo desenvolvido e o impacto dos métodos de seleção e substituição como é proposto em trabalhos futuros por BARCELLOS (2000).

Deve-se destacar a facilidade de implementação desses procedimentos de auto adaptação da taxa de mutação dentro do pseudocódigo de um Algoritmo Genético e também a ajuda que a adoção desses dá aos programadores iniciantes na área de computação evolucionária. Pois a fuzzificação de variáveis como diversidade

da população, adaptação média da população, etc. consegue determinar de forma iterativa a taxa de mutação para qualquer classe de problema.

Em paralelo, compreender como o comportamento dos métodos de seleção e substituição podem alterar a qualidade do resultado encontrado e a velocidade de convergência do algoritmo genético ajudará um programador iniciante a modelar um algoritmo mais adequado ao problema.

Logo, percebe-se uma necessidade de se investigar o comportamento dos algoritmos genéticos com o valor da taxa de mutação definida antes da execução ou durante a execução, com o intuito de entender o efeito dessa definição no desempenho no algoritmo genético.

1.2 - Delimitação do Estudo

Este trabalho limita-se somente à análise dos elementos que compõem um algoritmo genético capazes de influenciar diretamente na diversidade da população, como a taxa de mutação e os métodos de seleção e substituição, quando aplicado a problemas de otimização, sendo o indivíduo codificado em em vetor binária e tendo uma variação de retorno dentro dos procedimentos implementados no Universo de [0, 100] % .

1.3 - Objetivos

Os objetivos deste trabalho se dividem em objetivo geral e objetivos específicos.

1.3.1 - Objetivo Geral

Avaliar a convergência de um algoritmo genético cuja taxa de mutação é determinada em tempo de execução pela fuzzificação da diversidade da população atual quando combinada a métodos de seleção e da substituição diferentes dos propostos por Holland.

1.3.2 - Objetivos Específicos

I. Implementar um AG com taxa de mutação fixa e analisar o comportamento deste algoritmo de forma empírica.

II. Desenvolver um AG com taxa de mutação auto adaptável em função da fuzzificação da diversidade da população e do valor de adaptação do cromossomo a ser mutado.
III. Escolher um conjunto de métricas para avaliar o comportamento do AG proposto quando com o algoritmo genético de taxa de mutação fixa que obteve os melhores resultados para um conjunto de problemas de teste.
IV. Analisar os resultados obtidos empiricamente para os casos de testes, considerando a qualidade do indivíduo encontrado e o tempo de convergência.

1.4 - Organização deste trabalho

- Capítulo 2 - Algoritmo Genético: Apresenta o modelo de um algoritmo genético simples e como podem ser geradas alguns variações do mesmo, além da definição dos algoritmos genéticos auto adaptáveis;
- Capítulo 3 - Lógica Fuzzy: Faz uma introdução a lógica nebulosa e sua aplicação;
- Capítulo 4 - Algoritmo Proposto: Descreve o algoritmo genético com taxa de mutação auto adaptável por um conjunto de regras fuzzy, e apresenta os resultados dos experimentos realizados;
- Capítulo 5 - Conclusão: Apresenta as conclusões obtidas neste trabalho, considerações finais e os trabalhos futuros.

CAPÍTULO 2

Algoritmos Genéticos

Neste capítulo serão discutidos conceitos relacionados a definição dos algoritmos genéticos como representação, operadores, gerenciamento da população e o ajuste de parâmetros. Por fim, os trabalhos relacionados à pesquisa e a demonstração da sua importância.

2.1 - Introdução

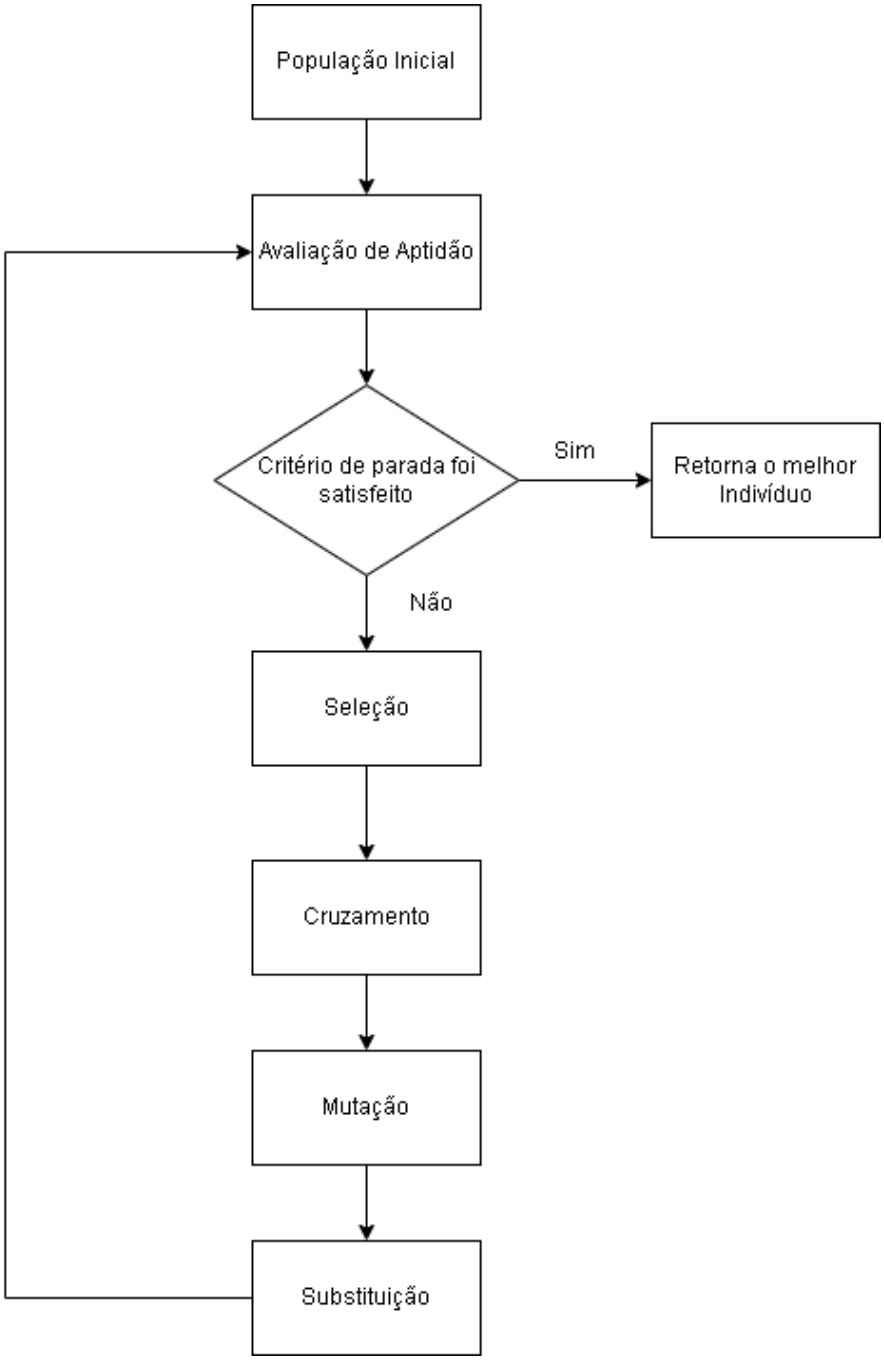
O algoritmo genético (AG) é uma técnica de busca inspirada na evolução natural. Esse algoritmo tem como ideia evoluir um conjunto de soluções de tamanho fixo de um determinado problema, denominado de população. Os elementos de população são vetores binários de tamanho fixo, chamados cromossomos, um cromossomo é uma abstração de um elemento do mundo real, que corresponde ao indivíduo na teoria de evolução das espécies. A cada cromossomo da população é atribuído um valor de adaptação que informa o quão bom são as características desse cromossomo para a sua sobrevivência no habitat atual. Dessa forma, o AG determina através da função de aptidão do indivíduo qual a chance dele sobreviver e ter descendentes nas gerações futuras.

Como os AG trabalham a otimização de um conjunto de soluções, então eles são mais rápidos que um algoritmo normal de busca trabalhando uma solução por vez. Segundo LINDEN (2012), por causa do carácter paralelo e aleatório do AG, não é garantido encontrar a melhor solução em um tempo curto, e sim soluções muito próximas a solução ótima.

O cromossomo é um vetor de caracteres pertencentes a um alfabeto, usado para armazenar as características dos indivíduos da população, em que cada posição é chamada de gene, e os caracteres que podem ocupar uma determinada posição são chamados de alelos (FREITAS, 2019). O alfabeto usado pode ser o binário, real, inteira, letras do alfabeto, entre outras. A interpretação das características presentes em um cromossomo depende da resolução modelada para o problema proposto.

Os módulos de uma algoritmo genético padrão estão ilustrado na figura 2.1:

Figura 2.1 - Exemplo de Estrutura Básica de um AG Simples



Fonte: Autor

Outro ponto importante, é a definição do tamanho da população, pois uma população muito pequena pode prejudicar o desempenho do AG, devido a baixa cobertura do espaço de busca do problema, por causa da baixa diversidade populacional. Já uma população muito grande, possui maior diversidade, mas necessita de mais tempo e recursos computacionais (KIM; HAN, 2000). Por último, a importância de gerar aleatoriamente os indivíduos da primeira população, é defendida por ROSA e LUZ (2009), na qual afirma que, essa aleatoriedade garante uma exploração melhor do espaço de busca em decorrência da variedade genética, caso contrário, o algoritmo poderá gerar vários indivíduos com o material genético parecido em um curto intervalo de tempo, tendo como consequência uma convergência genética prematura em torno de uma solução ótima local. Dessa maneira, dificultando a exploração do espaço de busca.

2.2 - Operadores Genéticos

Como mostrado na seção anterior (2.1), o AG é inspirado no processo evolutivo, logo seus módulos tentam simular o comportamento dos módulos descritos pelo processo de evolução genética que são o cruzamento, mutação, a seleção e a substituição descritas detalhadamente neste capítulo.

2.2.1 - Cruzamento

O cruzamento pode ser chamado de *crossover* ou recombinação, ele é inspirado na reprodução sexuada, que tem o intuito de permitir a propagação de características para as gerações posteriores (ROSA; LUZ, 2009), que para acontecer precisa de um conjunto de progenitores cuja cardinalidade é maior ou igual a 2, para gerar um conjunto de descendentes.

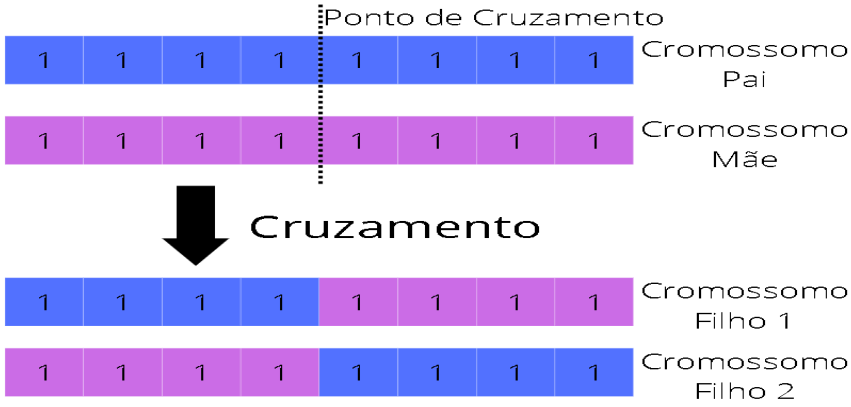
O funcionamento do cruzamento pode ser assim descrito, primeiro é selecionado os progenitores entre os indivíduos da população atual, pelo método elitista e da roleta (explicação na próxima seção 2.2.3), para cada conjunto de progenitores uma taxa de cruzamento é gerada aleatoriamente, chamada de probabilidade de cruzamento, que irá determinar se haverá a troca ou não de características entre eles. Caso aconteça o cruzamento, pode gerar um ou mais novos indivíduos, os quais poderão ou não fazer parte da próxima população dependendo da

sua adaptação e a sua validade dentro do escopo do problema, caso ele seja considerado inválido pode ser eliminado ou pode ter seus genes alterados, no intuito de promover a adaptação desses indivíduos ().

Existem alguns tipos de cruzamentos como o de **um ponto de corte**, cruzamento de vários pontos de corte e cruzamento proporcional (LUCAS, 2002).

No **Cruzamento de Um Ponto de Corte**, é gerado primeiramente um número aleatório dentro do tamanho do cromossomo um ponto de corte, ou seja, no intervalo fechado de [0, t[1]], em seguida é copiado a primeira parte do gene do progenitor 1 da posição inicial até o ponto de corte e depois é copiado a segunda parte do progenitor 2 até o final do cromossomo, é feito então o inverso para gerar o segundo indivíduo (LUCAS, 2002), como mostrado na Figura 2.2, onde a linha tracejada representa o ponto de corte escolhido aleatoriamente, nesse caso foi no bit 5.

Figura 2.2 - Exemplo de Cruzamento Um Ponto de Corte



Fonte: Autor

2.2.2 - Mutação

Para produzir a variabilidade genética que ocorre no surgimento de um novo indivíduo, como no ambiente, por contato com radiação que realiza modificações no indivíduo ou por erros na troca de informações genes durante a reprodução sexuada, o algoritmo genético faz uso da operação de mutação (NUNES, 2018).

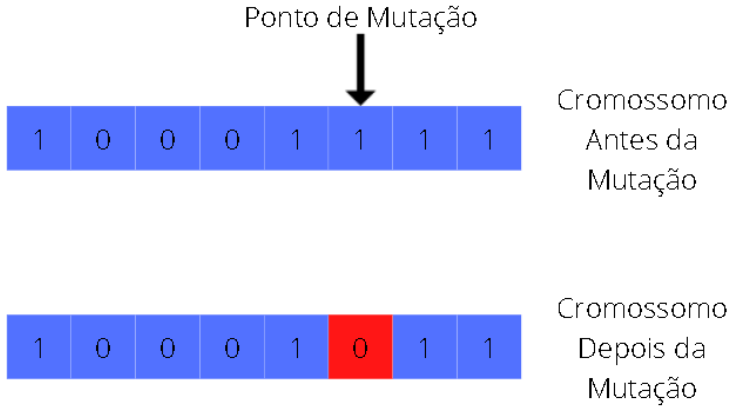
Alguns tipos de mutações são descritos por SONI e KUMAR (2014), por exemplo a mutação baseada na posição, a mutação baseada na ordem e a mutação de mistura e a mutação por complemento. Todas essas mutações recebem um cromossomo, altera seu material genético de acordo com um determinado propósito e

[1] tamanho do cromossomo

retorna um cromossomo diferente do cromossomo fornecido. Neste trabalho será adotada somente a mutação por complemento.

A **Mutação por Complemento** que tem como ideia principal realizar a mudança de um gene aleatório de acordo com o valor específico em que esse gene pode assumir. Funciona da seguinte forma, é escolhido aleatoriamente um gene G, em seguida seleciona-se aleatoriamente um alelo I_2 válido para o gene G que seja diferente do seu alelo I_1 atual, depois troca-se o alelo I_1 por I_2 no gene G. Esse tipo de mutação é bastante utilizado em representações do tipo real, inteiro e binário (KATO, 2021). Sendo representado na Figura 2.3.

Figura 2.3 - Exemplo de Mutação por Complemento



Fonte: (KATO,2021)

2.2.3 - Seleção

A seleção, assim como é observado na Teoria da Evolução de Charles Darwin, afirma que o indivíduo mais bem adaptado ao ambiente têm maior chance de sobreviver, e também de propagar seu material genético pelas próximas gerações através de seus descendentes (POZO, 2005).

Desse modo, existem vários tipos de seleção, neste trabalho será implementado somente a **Seleção por Roleta e a Seleção Elitista** (Eiben e Smith, 2007). A escolha dessas seleções baseia-se na política de gerenciamento da diversidade da população adotada por elas. Enquanto **Seleção Elitista**, a política adotada por ela é diminuir a diversidade da população restringindo a possibilidade de um indivíduo não adaptado sobreviver ou gerar descendentes. Na Seleção Roleta, a política é dar chances a todos os indivíduos de gerar descendentes, contribuindo para o aumento da variabilidade genética da população e consequentemente de sua diversidade.

A **Seleção Elitista** tem o objetivo de escolher indivíduos de maior adaptação da população atual para perpetuar seu material genético para gerações futuras. Sendo assim, utilizar esse tipo de seleção é uma boa forma de perpetuar um bom indivíduo. Funciona da seguinte forma, é ordenada a população e escolhido os *n* melhores indivíduos para efetuar o procedimento de cruzamento (Eiben e Smith, 2007).

Exemplo é na Figura 2.4, em que uma população P de tamanho 8 depois de ordenar e passar pela seleção Elitista, na qual limita a população de ancestrais para 4 membros como mostrado na população Resultante, esses membros têm o dever de efetuar o processo de cruzamento.

Figura 2.4- Exemplo de Seleção Elitista

P	
Cromossomo	*Score*
0101100101	5
0101111101	7
1110100101	6
1100000101	4
0101100111	6
0111111111	9
1110100101	6
1100000111	5

=

Resultante	
Cromossomo	*Score*
0111111111	9
0101111101	7
1110100101	6
1110100101	6

Fonte: Autor

A **Seleção por Roleta** é baseada na proporção da adaptação dos indivíduos de uma população, ou seja, a média esperada para a participação de um indivíduo nas gerações futuras é diretamente proporcional ao valor da adaptação desse indivíduos e inversamente proporcional ao valor da adaptação total da população (MITCHELL, 1999).

Segundo GIGUERE (1989) essa roleta pode ser implementada através de um vetor de tamanho *m*, onde cada indivíduo i de uma população p ocupa $(adaptação(i) \times m) \div (adaptaçãoTotal(p))$ células do vetor roleta. Depois de criada a roleta, esta é girada tantas vezes, até que o número de indivíduos ancestrais seja igual ao esperado.

No quadro 2.1, encontra-se uma população hipotética, com o ID, o seu genótipo, o *score* e o respectivo percentual de adaptabilidade.

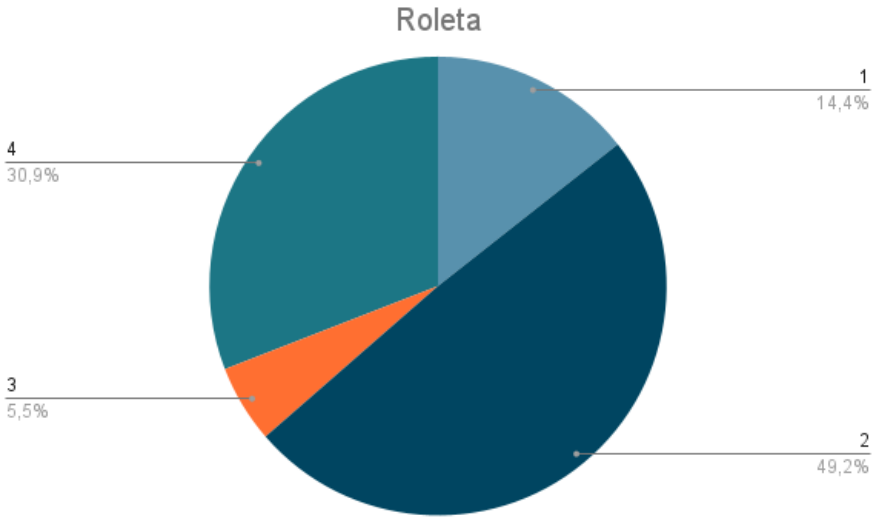
Quadro 2.1- População Hipotética

ID	Indivíduos	(*Fitness*)	% do Total
1	01101	169	14,4
2	11000	579	49,2
3	01000	64	5,5
4	10011	364	30,9
Total		1170	100,0

Fonte: (TEIXEIRA, 2005)

A Figura 2.5, mostra como fica a roleta construída de acordo com o valor de *score* da população hipotética.

Figura 2.5 - Roleta População Hipotética



Fonte: (TEIXEIRA, 2005)

2.2.4 - Substituição

A substituição tem como objetivo definir quais serão os indivíduos da população que irão permanecer nela ou que serão retirados. Desse modo, é garantido uma evolução dos indivíduos dentro da população (Arabas, Mulawka, 1994).

A substituição **Elitismo**, segundo EBERHART et al. (1996), afirma que o indivíduo de maior adaptação de uma geração pode não sobreviver às operações de seleção, recombinação e mutação. Sendo assim, utilizar esse tipo de substituição é uma boa forma de perpetuar um bom indivíduo nas populações futuras (BRITO, 2011).

Funciona da seguinte forma: ordena uma população formada pela união da população atual, população de descendentes e copia os *n* melhores indivíduos dessa população para fazerem parte da próxima população (MITCHELL, 1999).

Exemplo na Figura 2.6, em que ilustra uma população P de ancestrais e uma população P' de descendentes, ambas de tamanho 4. Depois, é juntado essas populações e ordenados em ordem decrescente mantendo vivos os 4 primeiros indivíduos vivos para a próxima geração, como mostrado na população Resultante.

Figura 2.6 - Exemplo de Substituição Elitista

P	
Cromossomo	*Score*
0101100101	5
0101111101	7
1110100101	6
1100000101	4

P'	
Cromossomo	*Score*
0101100111	6
0111111111	9
1110100101	6
1100000111	5

Resultante	
Cromossomo	*Score*
0111111111	9
0101111101	7
1110100101	6
1110100101	6

Fonte: Autor

A substituição por **Tempo de Vida** como definida por Arabas, Mulawka (1994), que acrescenta ao cromossomo da população dois valores inteiros, um para o parâmetro idade e outro para o parâmetro tempo de vida. Quando o cromossomo é gerado o valor de seu parâmetro idade é inicializado como zero e a cada geração o valor do parâmetro idade desse cromossomo é incrementado uma unidade e o valor tempo de vida inicializado segue essa função:

- Proporcional: $vida(i) = min\,(MinLT + n * f(i) / AvgFit\,,\, MaxLT)$

em que $MinLT$ é o tempo de vida mínimo permitido, $MaxLT$ é o tempo máximo de vida permitido, n é o tempo de vida médio, $f(i)$ é o valor da adaptação do cromossomo i, $AvgFit$ é o valor da adaptação média da população e a função $min()$ retorna o mínimo entre os dois parâmetros.

Quando a idade de um indivíduo for igual ao seu tempo de vida, então esse indivíduo é retirado da população independente do valor de sua adaptação. Deve-se observar que a população trabalhada com este método de substituição deverá ter tamanho ilimitado, podendo aumentar e diminuir durante a execução do algoritmo genético. A única coisa que não pode ocorrer é esta população ficar vazia.

2.2.5 - Ajuste de parâmetros em um AG

Este problema de determinar os parâmetros da taxa de mutação, tamanho da população e a taxa de cruzamento, que irão garantir um melhor desempenho do AG nos problemas aplicados, vem sendo estudado há bastante tempo, mas ainda há falta de um modelo para os algoritmos genéticos, que oriente o programador a fazer a melhor escolha (BARCELLOS, 2000). Assim os parâmetros escolhidos são definidos de forma empírica, ou seja, o programador têm que realizar vários testes variando um parâmetro e mantendo os outros parâmetros fixos, para determinar o melhor valor de cada parâmetro daquele algoritmo genético instanciado para um determinado problema.

Exemplo da importância de ajustar corretamente os parâmetros está na seguinte relação, se a taxa de mutação for aumentar bastante o algoritmo genético torna-se um algoritmo de busca aleatória. Caso contrário a diversidade da população estará prejudicada. A taxa de cruzamento influencia diretamente na convergência prematura da população para um ótimo local, quando ela é muito alta cobre praticamente toda a população com as características dos indivíduos mais adaptados, por outro lado quando ela é muito baixa pode-se perder as características dos melhores indivíduos. Já o tamanho da população influencia na qualidade do resultado encontrado, se o tamanho da população for muito pequeno a busca paralela no espaço de busca do problema pode levar muito tempo ou não consegui chegar em um ótimo global, se é muito grande precisará de muitos recursos computacionais (NUNES, 2018).

Outro ponto importante a ser destacado são os dois tipos de parâmetros chamados de qualitativo e quantitativo. O parâmetro quantitativo são valores numéricos atribuídos ao AG como taxa de mutação, tamanho da população, número de gerações, taxa de cruzamento. Já os parâmetros qualitativos referem-se aos tipos de funções que podem ser utilizadas, exemplo a de cruzamento que pode utilizar um ponto de corte, cruzamento uniforme, entre outros (EIBEN e SMIT, 2011).

Segundo CHEBBI e CHAOUACHI (2015), existem duas formas de atribuir um valor para os parâmetros: as que são realizadas durante a execução do programa e as que se mantêm constantes durante a execução, sendo definidas como ajuste de parâmetros e controle de parâmetros, respectivamente. No ajuste de parâmetros, os valores são definidos antes de iniciar o algoritmo e permanecem constantes durante

toda a execução e, o de controle de parâmetros modifica os parâmetros durante a execução do algoritmo (NUNES , 2018).

Este trabalho tem como ênfase analisar a variedade da taxa de mutação no cromossomo, na qual é responsável diretamente pela a diversidade populacional, que assim como a natureza durante a cópia dos materiais genéticos, ou implicações do ambiente, podem ocorrer modificações. Essas modificações nem sempre são benéficas para a população em um modo geral, pois a sua modificação é feita de forma aleatória em todo o cromossomo. Desse modo, o operador de mutação consegue encontrar várias soluções variadas no espaço de busca, aumentando assim seu poder de exploração (BARCELLOS, 2000).

Esse capítulo será necessário para o entendimento do tópico 4, assim como o próximo capítulo que irá tratar como funciona a Lógica Fuzzy.

CAPÍTULO 3

Lógica Fuzzy

Neste capítulo serão apresentados a representação e os operadores da Lógica Fuzzy, e como ela pode ser aplicada a sistemas de computação através da fuzzificação e defuzzificação das variáveis de controle desses sistemas.

3.1 - Introdução

Com o aumento da complexidade dos problemas do mundo real por situações ambíguas, foi necessário a construção de uma nova lógica que se aproximasse das decisões humanas, pois a lógica clássica não estava se encaixando bem nesses novos problemas, por que o mundo não poderia mais ser abstraído em dois valores lógicos, os que possuem e os que não possuem uma determinada propriedade (verdadeiro ou falso) (TARIG, 2001). Desse modo, surgiu a lógica Fuzzy ou Nebulosa com o intuito de adicionar valores lógicos intermediários entre os valores lógicos trabalhados pela lógica Booleana (BURDELIS, 2009). Por exemplo, se você perguntar a uma pessoa “O tempo hoje em sua cidade está quente?”, as respostas podem ser “está quente”, “está muito quente”, “está pouco quente”, “está morno”, “está ameno”, etc.

A Lógica Fuzzy foi introduzida em 1965 por Lofti A. Zadeh no seu trabalho *Fuzzy Sets*, na qual apresentou as noções de conjunto nebulosos como uma extensão da teoria dos conjuntos clássica, em que uma afirmação só podia ser verdadeira ou falsa (0 ou 1). Já a lógica Fuzzy tem como base a teoria dos conjuntos nebulosos, que permite atribuir a uma relação lógica valores verdades variando continuamente entre o

intervalo [0, 1], onde os extremos desse intervalo corresponde ao falso e verdadeiro da lógica clássica (BRITO, 2011).

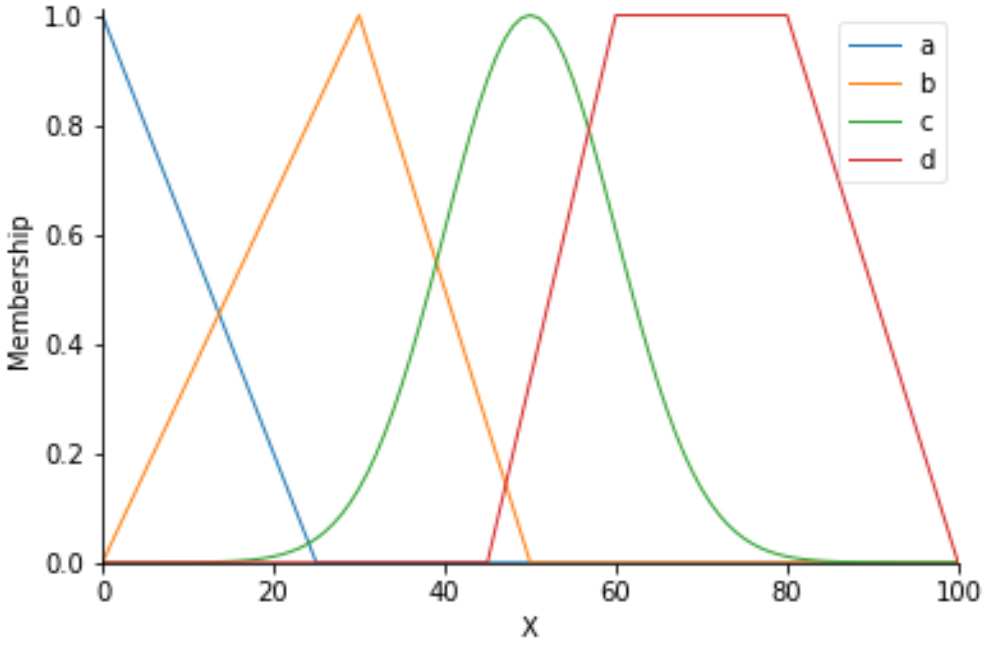
3.2 - Conjuntos Nebulosos

O Conjunto Fuzzy permite facilmente a modelagem de problemas do mundo real, pois imita o comportamento do pensamento humano no processo de tomada de decisão. Exemplo disso, é em relação ao clima, onde uma pessoa analisa o ambiente e suas variáveis, tomando no final uma decisão se está “muito calor”, “frio”, “muito frio” (BRITO, 2011).

Desse modo, é analisado o Conjunto Nebuloso, com o valor 1 representando pertencimento total e o valor zero(0) indicando a exclusão total. Portanto, essa generalização é extremamente importante para a função de pertencimento , ou seja, o conjunto nebuloso A no universo de discurso U é representado da seguinte forma $\mu_A(X)$, onde $A \subseteq B$, recebendo assim valores contínuos do intervalo [0,1] (BARG, 2002).

Essas funções de pertinência são frequentemente representadas na resolução de problemas, na Figura 3.1 tem o *membership* que representa a função de pertencimento, o **a** e o **b** são funções triangulares, o **c** uma função gaussiana e o **d** é uma função Trapezoidal (BRITO, 2011).

Figura 3.1 - Principais Representações das Funções de Pertencimento



Fonte: Adaptado de BRITO (2011)

3.3 - Operações dos Conjuntos Nebulosos

Segundo COX (1994) o conjunto nebuloso possui operações definidas para combinar e modificar, assim como os conjuntos convencionais. Como a ideia do conjunto Fuzzy foi fundamentada seguindo lógica booleana que tem as seguintes operações (*negação, ou, e*), é encontrado de forma similar como as operações de (*complemento, união e interseção*) (BARG, 2002).

3.3.1 - Complemento

O complemento tem como objetivo definir a função de pertinência oposta ao seu subconjunto, isto é, o complemento do subconjunto A, definido como Ā, sendo assim formado pelos pontos opostos de A. Tem como representação formal a equação $\mu\bar{A}(x) = 1 - \mu A(x)$.A representação gráfica está na Figura 3.2.

Figura 3.2 - Representação Gráfica do Complemento

Fonte: (BEDREGAL, 2013)

3.3.2 - União

A união tem como ideia principal unir dois subconjuntos, isto é, a união do subconjunto A com o B, na qual resulta em um subconjunto que abrange os pontos máximos dos dois subconjuntos unidos. Sua representação formal é $A \cup B \rightarrow \mu A(x) \cup \mu B(x) = \max(\mu A(x), \mu B(x))$. A representação gráfica está na figura 3.3.

Figura 3.3 - Representação Gráfica da União

Fonte: (BEDREGAL, 2013)

3.3.3 - Interseção

Na operação de interseção tem como objetivo obter a região comum entre dois subconjuntos, isto é, a interseção do subconjunto A com o B resultando em um subconjunto que tem os pontos que pertencem tanto ao subconjunto A e ao B. Sua

representação formal é A∩B → μA(x) ∩ μB(x) = min(μA(x), μB(x)) e sua representação gráfica é a Figura 3.4.

Figura 3.4 - Representação Gráfica da Interseção

Fonte: (BEDREGAL, 2013)

3.3 - Variáveis Linguísticas

Variáveis linguísticas tem como objetivo obter o raciocínio qualitativo humano, exemplo “mais”, “menos”, entre outros. Dessa modo, se uma variável permite palavras natural relacionadas com seus valores, então ela pode ser considerada uma variável linguística. Com isso, para se atribuir um significado aos termos linguísticos, é associado a cada um deles um conjunto Fuzzy definido sobre um universo de discurso, na qual foi definido (JANG 1997).

Logo, segundo Pacheco (1991), o raciocínio humano na sua essência é “aproximado”, e qualquer técnica de modelá-lo diferente está desprezando a principal vantagem humana, a de tratar diretamente com conceitos inexatos. Com isso, as variáveis linguísticas fornecem de maneira sistemática uma característica aproximada de fenômenos complexos ou mal definidos. Desse modo, permite o tratamento de sistemas que são muito complexos para serem analisados por meio de termos matemáticos convencionais (BARG, 2002).

3.4 - Expressão Fuzzy do Conhecimento

No Sistema Fuzzy é baseado em regras do tipo condição-ação, essas regras têm o seguinte formato de “Se-então”, com isso as palavras são caracterizadas por funções de pertinência contínua (JANG, 1997). Um exemplo dessa regra é: Se está com febre, então tome um remédio.

Logo, para construir um sistema nebuloso é necessário primeiro representar o conhecimento de um especialista em regras “Se-então”, em seguida acoplar todas as regras em um único sistema e por último escolher o modelo de sistema nebuloso, na qual irá ser trabalhado (JANG, 1997).

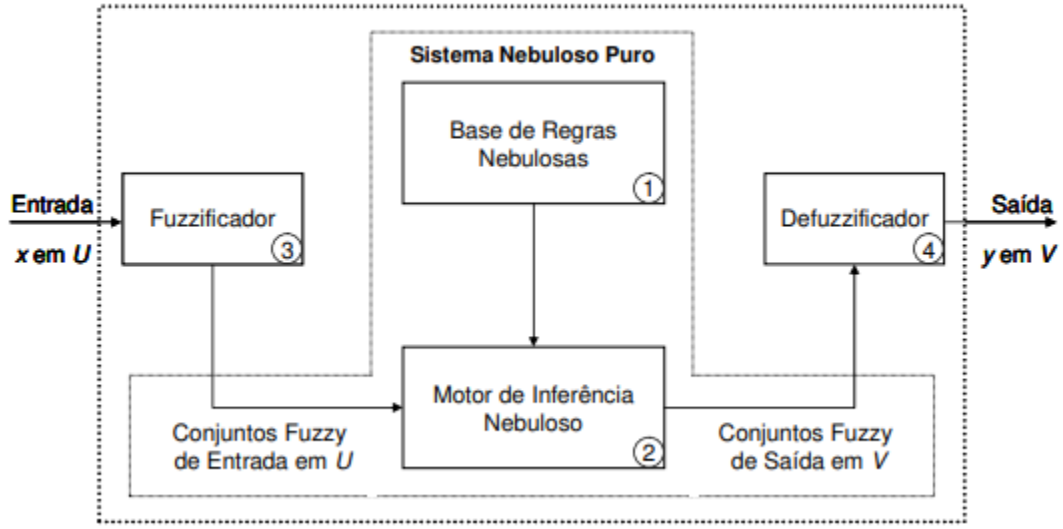
Segundo WANG (1997), existe três tipos de sistemas nebulosos que são Sistema Nebuloso Puro, Sistema Nebuloso de Takagi-Sugeno-Kang, utilizado para a entrada e saída de variáveis reais, e o Sistema Nebuloso de Mamdani com Fuzzificador e Defuzzificador, para mais informações sobre esses sistemas em JANG (1999).

O sistema mais utilizado é o de *Mamdani,* pois permite que as variáveis de entrada e saída sejam reais. Esse sistema foi o escolhido para elaboração deste trabalho, por ser um sistema que representa de forma prática o pensamento do especialista e também o seu Universo de entrada e saída encaixa na modelagem do problema. Desse modo, os outros sistemas não têm essas características.

3.5 - Sistema Nebuloso de Mamdani

O sistema Fuzzy é bastante diferente dos métodos convencionais de processos, já que eles são desenvolvidos via modelagem matemática dos processos de modo a derivar as ações como função do estado do processo (BARG, 2002). A Figura 3.5 tem a representação do sistema nebuloso, na qual está dividido em 4 partes.

Figura 3.5 - Sistema Nebuloso de Mamdani



Fonte: (BRITO, 2011)

Primeiramente, tem que construir uma Base de Regras Nebulosa, que é feita por meio da derivação do conhecimento de um especialista humano, essas regras são do tipo “Se-então”, sendo assim bastante importante implementá-las de forma correta. As regras tem o seguinte formato:

$$\text{Regra}^{(l)}\text{: Se } x_1 \text{ é } A^l_1 \text{ e } \ldots \text{ e } x_n \text{ é } A^l_n\text{, então y é } B^1 \qquad (1)$$

onde A^l_1 e B^l são conjuntos nebulosos em U_1, $V \subset R$, respectivamente; $x_1,\ldots, x_n \in U$ são as entradas; e $y \in V$ é a saída (BRITO, 2011). Por isso existe algumas propriedades, segundo BRITO (2011):

- **Integridade:** a regra tem que está completa, ou seja, cada $x \in U$, existe pelo menos uma regra nebulosa presente na base, na qual o grau de pertinência seja diferente de zero.
- **Consistência:** para ser consistente não pode ter duas ou mais regras com partes iguais de “se” e “então”, sendo necessário o grau de pertinência diferente de zero.
- **Continuidade:** o conjunto de regras nebulosas, não pode ter a parte do “então” um valor vazio.

Em seguida tem o motor de inferência nebuloso, na qual são combinadas às regras, com o intuito de mapear o conjunto nebuloso $A \in U$ em um conjunto nebuloso $B \in V$. Desse modo, encontra-se alguns métodos de inferência utilizados (BRITO, 2011):

- **Inferência Baseada em Composição:** o método tem como objetivo combinar em uma única relação U x V, na qual ficará como uma única regra “se-então”. Desse modo, para executar essa combinação é preciso utilizar os operadores de união e interseção para combiná-las. Com isso, na união as regras são vistas como independentes, e na interseção as regras são agrupadas e no final todas as condições têm que ser satisfeitas para que haja uma consequência, para mais detalhes em WANG (1997).
- **Inferência Baseada em Regra Individual:** cada regra da base determina a saída de um conjunto nebuloso e a saída final do sistema de inferência é a combinação de todas as saídas individuais, essa saída é obtida pela união ou interseção. Tem uma ideia similar ao item anterior, o que muda é que no final é utilizado Modus Ponens Generalizado[2], ou seja, a saída do motor de inferência nebuloso é a combinação de todas as saídas individuais(BRITO, 2011), para mais detalhes em WANG (1997).

Existem também outras regras como a de Intuição, Eficiência Computacional e a de Propriedades Especiais (BRITO,2011). Além disso, o motor de inferência nebuloso pode ser construído para um sistema específico ou utilizar motores já propostos.

[2] Quando existe apenas uma regra é utilizado *Modus Ponens Generalizado* (WANG, 1997)

Em terceiro é o Fuzzificador, que segundo BRITO (2011) são os mapeadores de valores reais x* ∈ U em conjuntos nebulosos A’ em U. Desse jeito, são considerados três fatores para a escolha do fuzzificador mais adequado para um determinado problema.

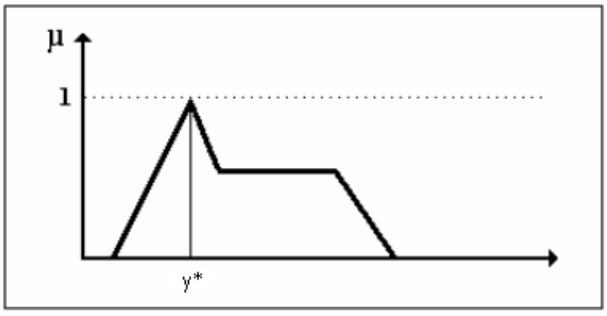
1. Deve-se considerar a entrada um valor discreto no ponto x*, isto é, tem que ter uma função de pertinência que abranja esse ponto no conjunto nebuloso.
2. Caso tenha ruído na entrada do sistema, é dever do fuzzificador minimizar.
3. É tarefa do fuzzificador simplificar a computação realizada no motor de inferência nebuloso.

Na figura 3.1, é observado os tipos de fuzzificadores mais utilizados na construção de sistema e controladores nebulosos.

Por último é o defuzzificador, na qual são os mapeadores do conjunto nebuloso B’ em V, em valores discretos y* ∈ V (BRITO,2011). Segundo ROSS (1995), existem sete métodos que têm sido pesquisado e popularizado, exemplo desses métodos é o de maior pertinência, o método do centróide, e a média de pertinência máxima.

O primeiro método é o princípio de maior pertinência, onde limita-se ao pico da função, como representado na Figura 3.6.

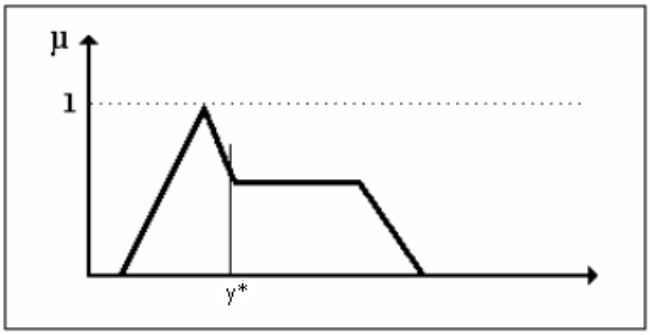
Figura 3.6 - Método de Defuzzificação da Maior Pertinência



Fonte: (BARG, 2002)

O segundo método é o da centróide, mais comum na defuzificação, ele tem como saída o ponto que divide a área da função de pertinência em duas partes iguais (BARG, 2002). Como ilustrado na Figura 3.7.

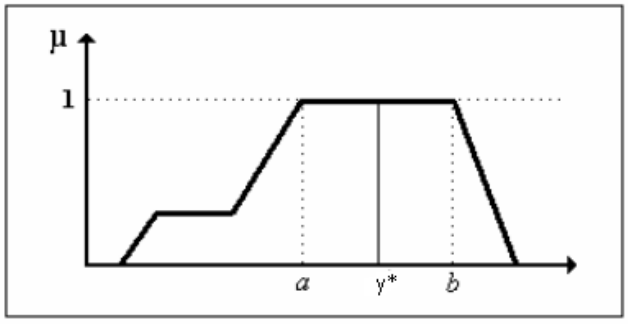
Figura 3.7 - Método da Centróide



Fonte: (BARG, 2002)

O terceiro é o método da pertinência máxima. Tem como ideia principal a localização da maior pertinência, na qual pode se limitar a um único ponto ou a diversos pontos. Logo, o valor de defuzzificação é dado pela média do ponto a e b, como representado na Figura 3.8.

Figura 3.8 - Método da Pertinência Máxima



Fonte: (BARG, 2002)

Além disso, para a escolha de cada um desses desfuzzificadores existem três critérios para determinar qual será utilizado no sistema (BRITO, 2011):

1. **Continuidade:** mudanças no conjunto nebuloso B' não pode gerar em grandes mudanças em y*.
2. **Simplicidade Computacional:** é importante para operar controladores de sistemas em tempo real.
3. **Plausibilidade:** o ponto y* deve representar o B' da melhor forma possível.

No próximo capítulo será apresentado o desenvolvimento do algoritmo genético combinado com a Lógica Fuzzy e também mostrar os resultados obtidos nos problemas OneMax e o Combinatorial '01'.

CAPÍTULO 4

Algoritmo Proposto

Neste capítulo serão apresentados como os conceitos de algoritmo genético auto adaptável e lógica fuzzy foram combinados para gerar um algoritmo genético auto adaptável que converge em menor tempo e encontra resultado de alta qualidade.

4.1 - Introdução

Na internet existem milhares de trabalhos relacionados a algoritmo genético auto adaptável, os que fazem o ajuste com e sem a representação dos parâmetros no cromossomo, os que fazem ajuste somente de um parâmetro ou de mais parâmetros, etc. Desse modo, primeiramente, foram definidas as bases para limitar a busca de artigos e publicações relacionados a esta monografia, sendo elas: ACM, IEEE Xplore, Periódicos CAPES, Scopus, com um filtro no Periódico Capes que foi somente tópicos sobre algoritmos genéticos. A *string* selecionada: ("*Genetic Algorithm*") AND ("*fuzzy logic*") AND ("*parameter adjustmen*t").

O resultado da busca foi na ACM 11 resultados, 38 resultados no IEEE Xplore, 50 resultados no Periódicos CAPES e 13 resultados na Scopus, totalizando 112 estudos, sendo 7 estudos duplicados. Desse modo, com o intuito de simplificar, foram selecionados 4 estudos relacionados a este trabalho para mostrar a sua importância, já que a grande maioria deles está relacionada à utilização da lógica Fuzzy para a resolução de problemas de otimização.

No primeiro artigo selecionado o de FERRARI (2014), foi utilizado Lógica Fuzzy para determinar os parâmetros de taxa de cruzamento e mutação de forma iterativa na Evolução Diferencial. Tendo chegado à conclusão o Algoritmo Diferencial auto

adaptável que utiliza a Lógica Fuzzy converge de forma mais rápida do que o Algoritmo Diferencial convencional.

Já na dissertação de mestrado do BURDELIS (2019) também foi utilizado a Lógica Fuzzy para diminuir o número de gerações e o tempo necessário para o Algoritmo Genético convergir em diferentes problemas como o do Caixeiro Viajante, Minimização de Funções, mostrando assim uma ótima abordagem a utilização dessa lógica para resolução de problemas.

Em terceiro tem o trabalho do CARVALHO (2017), na qual analisou a taxa de mutação e a de cruzamento, em relação ao tempo de convergência dos AG elitistas. Essas taxas variaram de 0,11 a 0,9, utilizando a representação binária de 8 *bits*. Sendo aplicado em problema unidimensional e bidimensional. Com isso, o autor concluiu que o operador de mutação foi o que mais influenciou na velocidade de convergência em relação ao operador de cruzamento.

O estudo de PINHO et al. (2007), teve o foco nos parâmetros: tamanho da população (50 ou 200), número de gerações (50 ou 100), taxa de cruzamento (10% ou 80%) e mutação (1% ou 30%). O algoritmo utilizado tinha como objetivo o ponto máximo de uma função. Logo, segundo o autor os pontos que tiveram maior influência na solução foram tamanho da população, número de gerações e a relação entre o número de gerações, taxa de cruzamento e mutação.

Definir de forma correta os parâmetros que dê o melhor resultado para um Algoritmo Genético (seção 2.6), é bastante complicado pois tem que levar em consideração vários fatores como visto no trabalho de PINHO (2007) e CARVALHO (2017). Uma das vantagens de utilizar a lógica Fuzzy é que ela funciona de forma iterativa , ou seja, muda ao longo da execução do programa, como é visto no trabalho BURDELIS (2019) e FERRARI (2014).

Por fim, percebeu-se uma lacuna de estudos comparativos em Algoritmo Genéticos Simples em relação aos Algoritmo Genéticos Auto Adaptativos que determina de forma iterativa a taxa de mutação e que avaliam também a combinação da seleção e da substituição. Sendo este o foco do presente trabalho analisando o número de gerações, o tempo gasto para convergir, diversidade da população, o *score* do melhor cromossomo no determinado intervalo de geração. Neste capítulo será abordado sobre o ambiente de teste, o desenvolvimento do AG e as modificações realizadas neles, de forma explicada.

4.2 - Configuração do Ambiente de Teste

Para realizar a implementação do Algoritmo Genético foi utilizado o ambiente virtual do Google Colab[3], que tem as seguintes configurações:

- RAM: aproximadamente 12 GB
- HD: aproximadamente 108 GB

Foi utilizado a versão 3.7.13 do Python, também foi utilizado algumas bibliotecas, como:

- ipython-autotime versão 0.3.1 para determina o tempo de execução
- matplotlib versão 3.2.2 para mostrar os gráficos
- scikit fuzzy versão 0.4.2 para montar o sistema Fuzzy

4.3 - Especificação dos Algoritmos

Esta seção está dividida em cada procedimento presente nas formas de implementação dos Algoritmos Genéticos adotados que são **AGEE, AGETV, AGRE, AGRTV, AGFEE, AGFETV, AGFRE, AGFRTV** com o intuito de mostrar as escolhas adotadas.

4.3.1 - Tamanho do Cromossomo e da População e a Representação

O tamanho do cromossomo adotado foi dez (10) e o tamanho da população foi de dez (10) indivíduos, a representação definida foi a binária, ou seja, sua composição será de zeros ou uns.

4.3.2 - Função de Seleção

A função de seleção adotada para definir quais serão os pais selecionados para realizar o cruzamento foi a elitista e o método da roleta. Sendo a Elitista escolhida os 5 melhores indivíduos da população utilizada nos algoritmos **AGEE, AGETV, AGFEE, AGFETV** e a do método da roleta utilizada nos algoritmos **AGRE, AGRTV, AGFRE, AGFRTV**.

[3] https://colab.research.google.com/

4.3.3 - Função de Cruzamento

A taxa de cruzamento adotada foi de 90%, foi adotado um valor bem alto para não ter interferência e também foi definido o cruzamento de um ponto de corte.

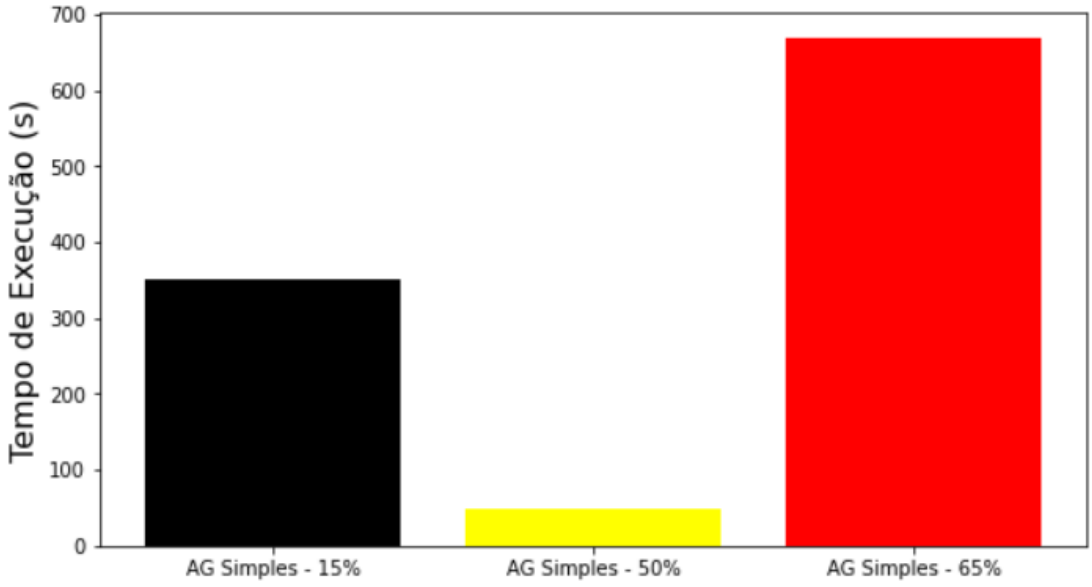
4.3.4 - Função de Mutação

Nesta seção será descrito como foi a implementação da função de mutação que é por complemento e o diferencial de cada uma delas, na qual foi dividido em Algoritmo Genético Simples e aplicação da Lógica Fuzzy.

4.3.4.1 Algoritmo Genético Simples

Para determinar a porcentagem de mutação que irá ser utilizado pelo o Algoritmo Genético Simples, foi realizado um teste no tempo de convergência em segundos de cada algoritmo variando a taxa de mutação em 15% , 50% e 65%, que teve o seguinte resultado determinado no Gráfico 4.1.

Gráfico 4.1 - Tempo de Execução do Algoritmo Genético Simples



Fonte: Autor

Desse modo, a porcentagem escolhida para representar o AG Simples em comparação ao Algoritmo Genético Auto Adaptável foi o AG Simples com a taxa de mutação de 50%. Sendo os Algoritmos Genéticos Simples que utilizaram esse método foram os **AGEE, AGETV, AGRE, AGRTV.**

4.3.4.2 - Aplicação da Lógica Fuzzy

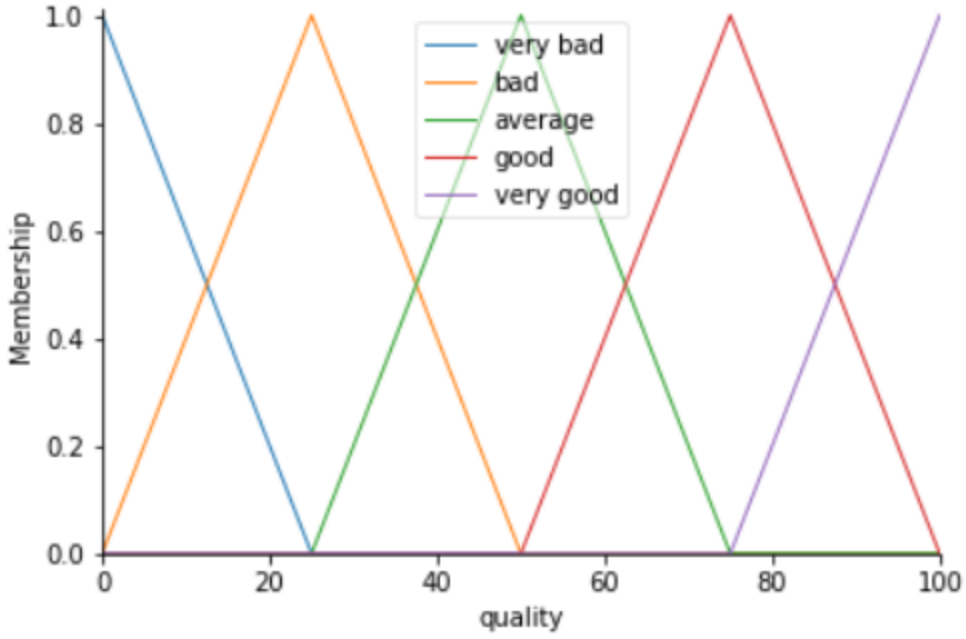
Como visto na seção 2.3, que explica o funcionamento da Lógica Fuzzy. Desse modo, foi adotado a representação da função triangular e testado a quantidade de valores difusos que mais se adequa ao problema do OneMax. Primeiro foram escolhidos os antecedentes e os consequentes, sendo como os antecedentes a diversidade da população (média aritmética do *score* da população) e a qualidade do cromossomo (*score*), resultando no consequente a porcentagem de mutação para determinado indivíduo. Além disso, os antecedentes e o consequente estão no Universo de 0 a 100.

Logo, o teste consiste em 100 execuções, em que cada execução contabiliza um determinado tempo que levou para encontrar o indivíduo ótimo. Os testes realizados no Algoritmo Genético Auto Adaptável foram:

1. Os dois Antecedentes tem o conjunto Fuzzy [“*bad*”, "*average*", “*good*”] e o consequente [“*low*” , "*average*”, “*high*”], como é observado o antecedente da diversidade da população no Apêndice A , qualidade no Apêndice B e o consequente da porcentagem de mutação no Apêndice C e as regras no Apêndice D.
2. A diversidade populacional tem o conjunto Nebuloso [“*very bad*”, “*bad*”, "*average*", “*good*”, “*very good*”], a qualidade do cromossomo [“*bad*”, "*average*", “*good*”] e o consequente em [“*low*” , "*average*”, “*high*”], representado o antecedente da diversidade da população o Gráfico 4.2, qualidade no Apêndice B e o consequente da porcentagem de mutação no Apêndice C e as regras no Apêndice F.
3. A diversidade populacional dividida em [“*very bad*”, “*bad*”, "*average*", “*good*”, “*very good*”], a qualidade do cromossomo em [“*very bad*”, “*bad*”, "*average*", “*good*”, “very *good*”] e o consequente em [“*low*” , "*average*”, “*high*”], representado o antecedente da diversidade da população no Apêndice e a qualidade do cromossomo o Gráfico 4.2 e o consequente da porcentagem de mutação no Apêndice C e as regras no Apêndice G.
4. A diversidade populacional dividida em [“*very bad*”, “*bad*”, "*average*", “*good*”, “*very good*”], a qualidade do cromossomo em [“very *bad*”, “*bad*”, "*average*", “*good*”, “*very good*”] e o consequente em [“ *very low*” , “*low*”, "*average*”, “*high*”, “*very high*”], representado o antecedente da diversidade da população no

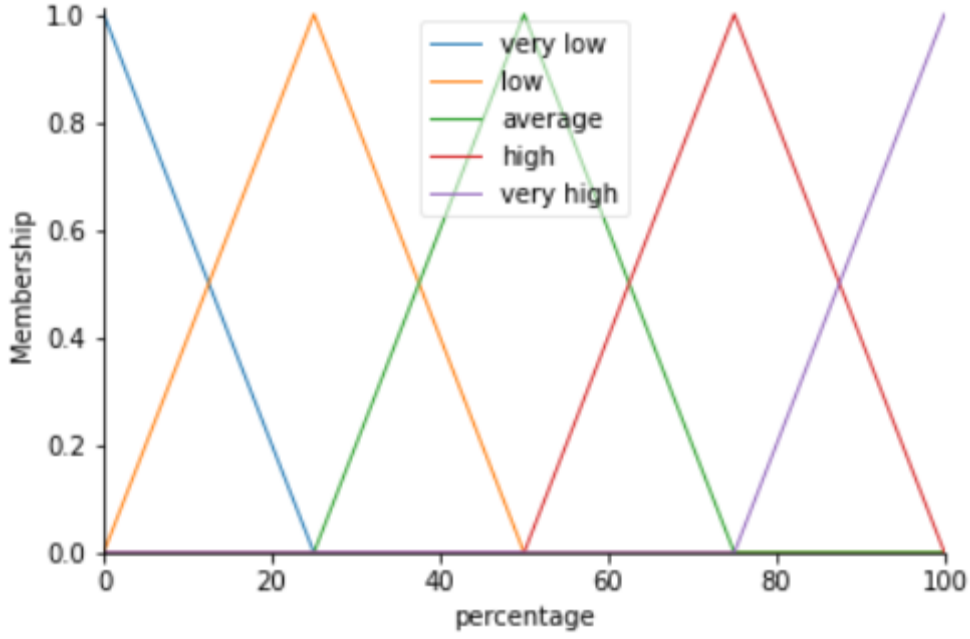
Apêndice e a qualidade no Gráfico 4.2 e o consequente da porcentagem o Gráfico 4.3 e as regras no Quadro 4.1.

Gráfico 4.2 - Função de Pertinência Triangular Antecedentes



Fonte: Autor

Gráfico 4.3- Função Triangular Descendente



Fonte: Autor

Quadro 4.1 - Tabelas de Regras Teste 4

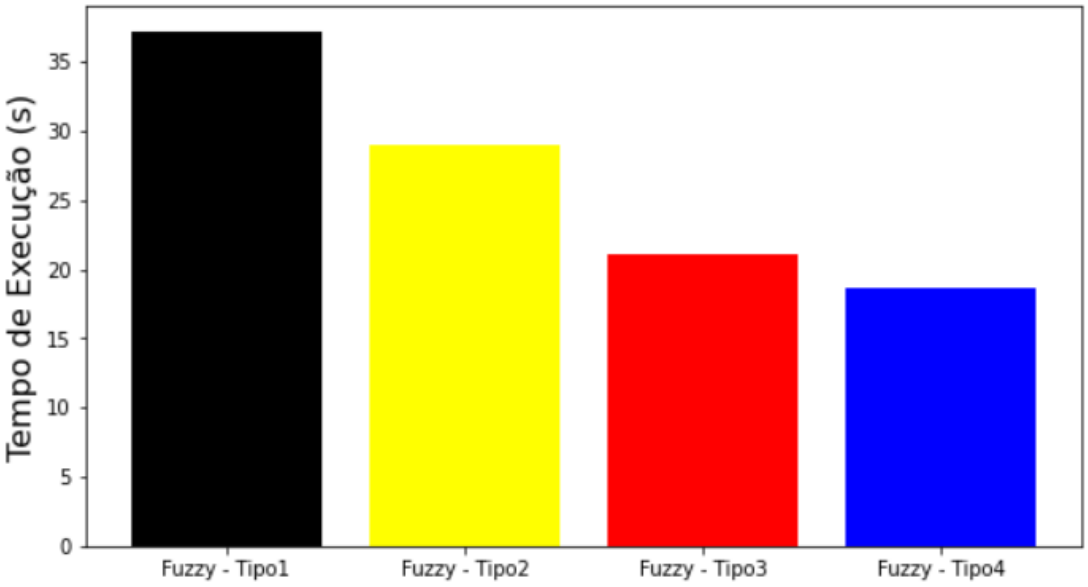
Regra	Se	Qualidade	E	DP	Então	PM

1	Se	*very bad*	E	*very bad*	Então	*very high*
2	Se	*very bad*	E	*bad*	Então	*very high*
3	Se	*very bad*	E	*average*	Então	*very high*
4	Se	*very bad*	E	*good*	Então	*very high*
5	Se	*very bad*	E	*very good*	Então	*very high*
6	Se	*bad*	E	*very bad*	Então	*very high*
7	Se	*bad*	E	*bad*	Então	*high*
8	Se	*bad*	E	*average*	Então	*average*
9	Se	*bad*	E	*good*	Então	*low*
10	Se	*bad*	E	*very good*	Então	*high*
11	Se	*average*	E	*very bad*	Então	*high*
12	Se	*average*	E	*bad*	Então	*average*
13	Se	*average*	E	*average*	Então	*average*
14	Se	*average*	E	*good*	Então	*low*
15	Se	*average*	E	*very good*	Então	*low*
16	Se	*good*	E	*very bad*	Então	*how*
17	Se	*good*	E	*bad*	Então	*low*
18	Se	*good*	E	*average*	Então	*low*
19	Se	*good*	E	*good*	Então	*low*
20	Se	*good*	E	*very good*	Então	*very low*
21	Se	*very good*	E	*very bad*	Então	*very low*
22	Se	*very good*	E	*bad*	Então	*very low*
23	Se	*very good*	E	*average*	Então	*very low*
24	Se	*very good*	E	*good*	Então	*very low*
25	Se	*very good*	E	*very good*	Então	*very low*

Fonte: Autor

O resultado no Gráfico 4.4 é que quanto menor o tempo de convergência em segundos do AG combinado com a Lógica Fuzzy mais fielmente é representado o pensamento de um especialista da área. Dessa forma, foi escolhida a representação do **teste 4** para realizar a comparação com o Algoritmo Genético Simples. Além disso, os algoritmos genéticos combinados com a Lógica Fuzzy foram **AGFEE, AGFETV, AGFRE, AGFRTV.**

Gráfico 4.4 - Tempo de Execução do Algoritmo Genético Proposto



Fonte: Autor

4.3.5 - Substituição

A função de substituição adotada foi a elitista e a do tempo de vida. Dessa forma, os algoritmos ficaram divididos da seguinte maneira:

- Elitista: **AGSEE, AGSRE, AGFEE e AGFRE**
- Tempo de Vida: **AGSETV, AGSRTV, AGFETV e AGFRTV**, sendo o tempo mínimo de vida (MinLtT) defino 1 geração e o tempo máximo de vida (MaxLT) foi de 3 gerações.

4.4 - Métricas de Avaliação

As métricas escolhidas neste trabalho para a realização da comparação foram a quantidade de gerações, a diversidade da população, o tempo de execução necessário para o algoritmo genético encontrar a solução do problema, o *score* do cromossomo mais adaptado por intervalo de geração.

4.5 - Resultados e Discussão

Cada algoritmo foi executado 100 vezes, ou seja, cada execução significa que o algoritmo foi iniciado e somente é parado ao encontrar a solução do problema. Quando encontrada é armazenado os dados e em seguida o sistema é reiniciado, esse *looping* acontece até que sejam completadas as 100 execuções, para assim, ser realizado a extração dos resultados.

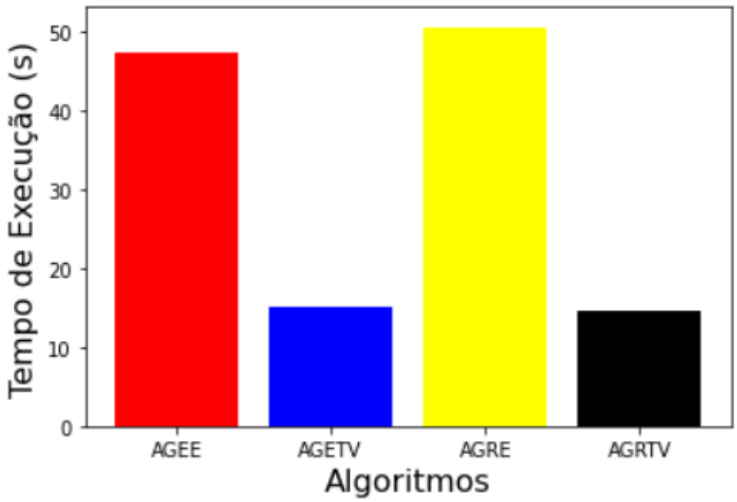
4.5.1 - Problema OneMax

O OneMax é um problema que consiste na contagem de *bits* um (1) que cada cromossomo possui, além dessa quantia, representar também a aptidão do indivíduo. Assim, o binário ótimo é constituído da *string* em que todos os *bits* são uns (1). O espaço de solução ou domínio do problema OneMax depende do comprimento da *string*. Uma característica importante do domínio OneMax é que todos os *bit*s não são correlacionados (GIGUERE, 1998). A simplicidade do problema OneMax torna-o um excelente candidato para estudar a avaliação do desempenho do Algoritmo Genético Simples com o Algoritmo genético auto adaptável em relação a variação da taxa de mutação e mostrar também o impacto no desempenho dos diferentes tipos de combinação de seleção e substituição.

4.5.1.1 Resultados Obtidos no AG Simples

Primeiro foi analisado o tempo de convergência dos algoritmos genéticos simples e suas variações como mostrado no gráfico abaixo:

Gráfico 4.5 - Tempo de Execução AG Simples (OneMax)



Fonte: Autor

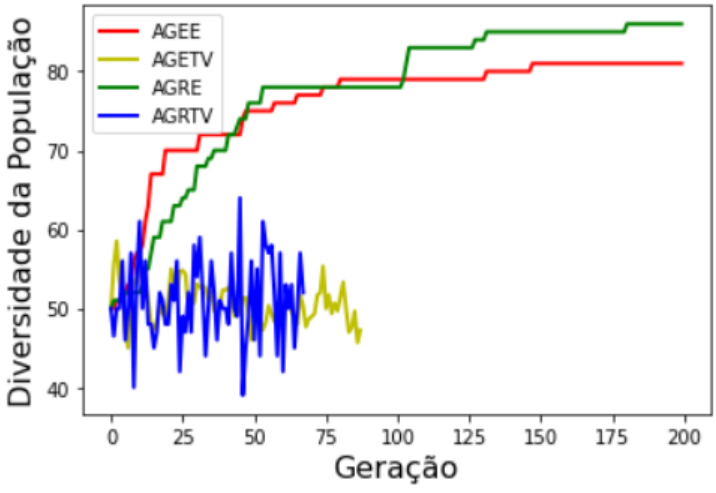
O gráfico 4.5 mostra o tempo total das 100 execuções que cada algoritmo obteve, ou seja, a média aritmética do tempo que cada execução levou para encontrar a melhor solução do problema OneMax. Diante do exposto, é notável a diferença que existe no tempo de execução dos dois algoritmos.

O algoritmo AGRE obteve o pior desempenho, cerca de 51.0s, quando comparado ao desempenho dos outros algoritmos genéticos simples que obtiveram AGEE 47.3s, AGETV 15.0s e o AGRTV 14.5s.

Um dos motivos analisados para essa diferença de tempo do AG simples que escolheu a substituição Elitista em relação aos AG simples com a substituição do Tempo de Vida, foi a demora da convergência da população como visto em (KIM; HAN, 2000) devido ao tamanho da população que impacta diretamente no tempo de convergência dela, isto é, a substituição por Tempo de Vida mantém uma variedade maior de indivíduos vivos por intervalo de tempo, diferente da elitista que a cada iteração mata os indivíduos com menor *scores*.

Além disso, foi verificada a diversidade da população presente em uma execução, em que foram extraídas as 200 primeiras gerações de cada algoritmo como mostrado no Gráfico 4.6. Na qual, observou-se que os Algoritmos Genéticos Simples com Elitismo na substituição apresenta uma diversidade no início igual ao do AG simples com substituição por Tempo de Vida, porém no decorrer das gerações essa diversidade muda, devido ao fato dos algoritmos AGETV e o AGRTV ter uma população maior, porém a diversidade presente nela não é grande, por outro lado os algoritmos AGEE e o AGRE tem uma população reduzida, mas a quantidade de indivíduos diferentes dentro dela é grande.

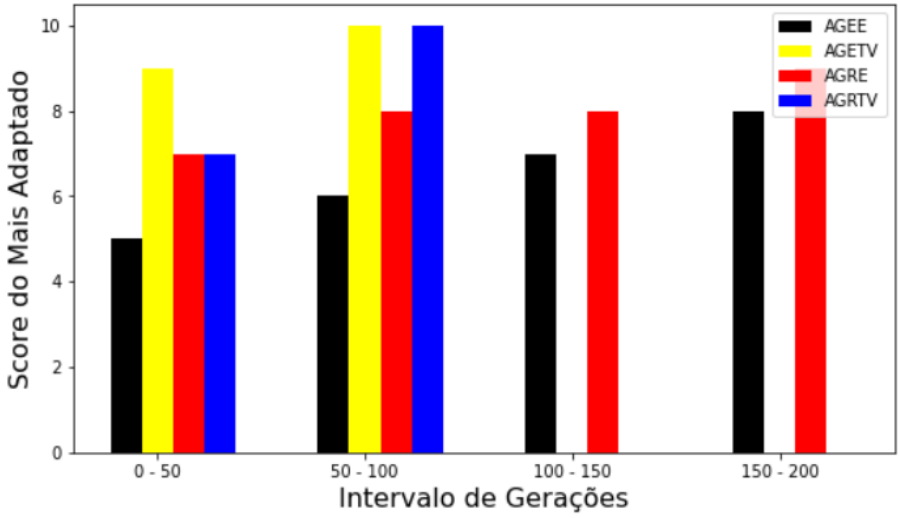
Gráfico 4.6 - Diversidade da População por Geração AG Simples (OneMax)



Fonte: Autor

Por fim, o Gráfico 4.7 mostra o resultado dos cromossomos mais bem adaptados nas primeiras 200 gerações, sendo elas divididas em quatro intervalos que são [0,50[,[50,100[, [100,150[, [150,200] , em que cada intervalo é mostrado o cromossomo de maior *score* presente no determinado algoritmo. Sendo destacado os AG Simples implementados com o Tempo de Vida, na qual consegue encontrar o indivíduo ótimo primeiro.

Gráfico 4.7 - Mais adaptado por intervalo de Geração AG Simples (OneMax)

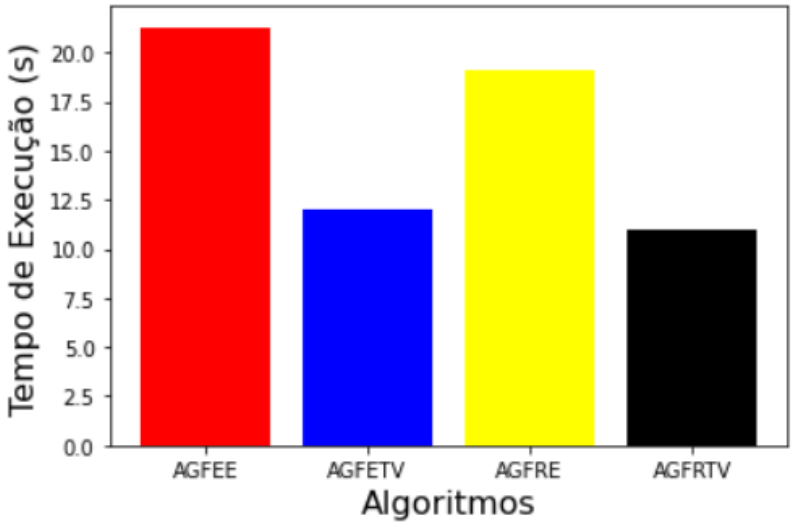


Fonte: Autor

4.5.1.2 Resultados Obtidos no AG Proposto

Os testes realizados no Algoritmo Genético Simples foram também realizados nos Algoritmos Genéticos combinados com a Lógica Fuzzy com suas variações no método da substituição para conseguir realizar as comparações entre eles.

Gráfico 4.8 - Tempo de Execução AG Proposto (OneMax)



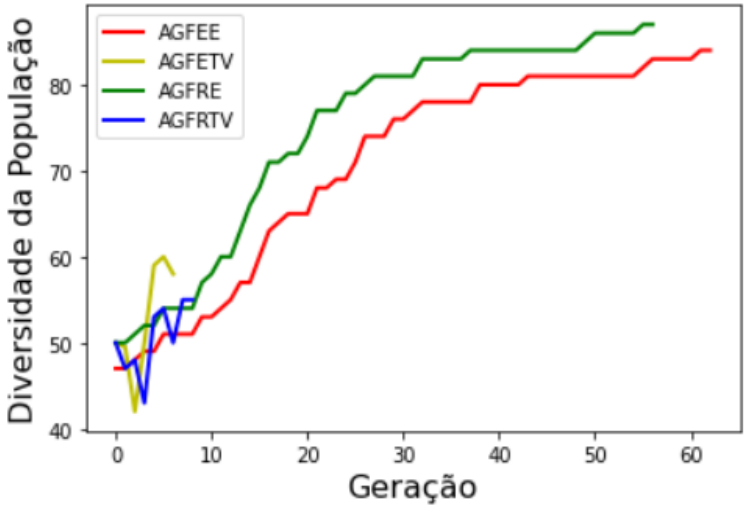
Fonte: Autor

O algoritmo AGFEE obteve o pior desempenho, cerca de 21.3s, quando comparado ao desempenho dos outros algoritmos genéticos combinado com a Lógica Fuzzy que obtiveram AGFETV 12.0s, AGFRE 19.4s e o AGFRTV 11.0s.

Os algoritmos genéticos auto adaptáveis que utilizaram o método da substituição por Tempo de Vida, que tem um aumento da população, na qual impacta na redução do tempo de convergência do AG. Além disso, o AG auto adaptável mantém de forma elevada a pesquisa nas macrorregiões, ou seja, varre de modo efetivo um maior números de espaços de buscas, isso ocorre devido ao fato do mesmo conseguir inferir uma taxa de mutação apropriada a cada cromossomo presente na população, levando em consideração a diversidade da população e a qualidade do cromossomo, a qual interfere numa convergência mais rápida.

Houve uma verificação na diversidade da população presente em uma execução, em que foram extraídas as 200 primeiras gerações de cada algoritmo como mostrado no Gráfico 4.9. Na qual, observou-se que os Algoritmos Genéticos Auto Adaptável com Elitismo na substituição apresenta uma diversidade no início igual ao do AG Auto Adaptável com substituição por Tempo de Vida, porém no decorrer das gerações essa diversidade divergem, devido ao fato dos algoritmos AGFETV e o AGFRTV ter uma população maior, porém a diversidade presente nela não é grande, por outro lado os algoritmos AGFEE e o AGFRE tem uma população reduzida, o que impacta no aumento da sua diversidade, isto é, menor a chance de ter indivíduos com o mesmo material genético.

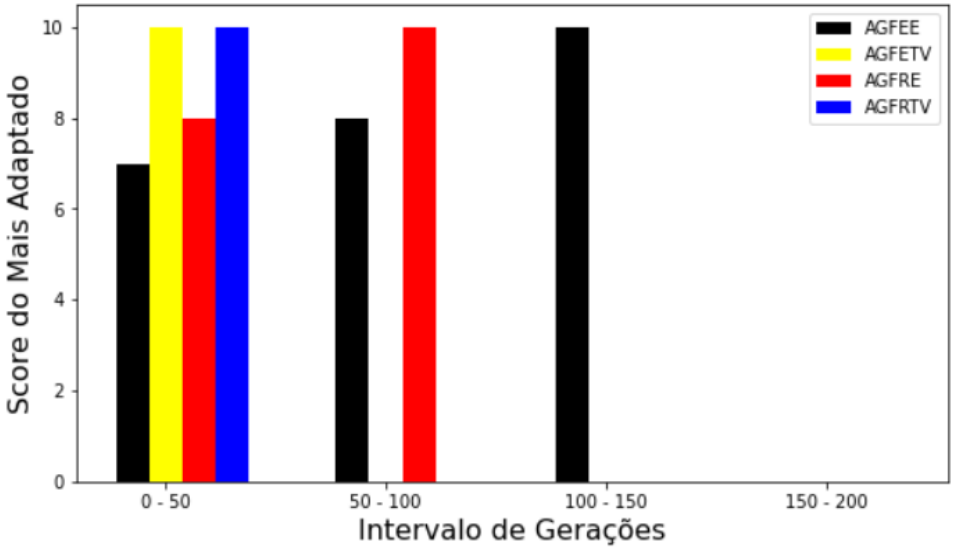
Gráfico 4.9 - Diversidade da População por Geração AG Proposto (OneMax)



Fonte: Autor

No Gráfico 4.10 ilustra o resultado dos cromossomos mais bem adaptados nas primeiras 200 gerações, sendo elas divididas em quatro intervalos que são [0,50[,[50,100[, [100,150[, [150,200] , em que cada intervalo é mostrado o cromossomo de maior *score* presente no determinado algoritmo. Os algoritmos que se destacaram foram os AGFETV e o AGFRTV, na qual encontraram em menos de 50 gerações o indivíduo que satisfaz o problema. Mostrando assim a importância de utilizar os operadores genéticos de forma correta.

Gráfico 4.10 - Mais adaptado por intervalo de Geração AG Proposto (OneMax)



Fonte: Autor

4.5.1.3 - Comparação entre o AG Simples e o AG Proposto

No Quadro 4.2, está a comparação em porcentagem em relação aos AG executados no problema OneMax, sendo o valor positivo indicando que foi *x%* maior que aquele AG avaliado e o valor negativo indica o quanto foi ganho, sendo realizado um truncamento para uma casa decimal.

Quadro 4.2 - Comparação do Tempo de Convergência AG Simples e AG Proposto (OneMax)

	AGRE	AGRTV	AGEE	AGETV	AGFRE	AGFRTV	AGFEE	AGFETV
AGRE	X	71.6%	7.2%	76.5%	62.0%	78.4%	58.2%	76.4%
AGRTV	-251.7%	X	-226.2%	-3.5%	-38.8%	24.1%	-46.9%	17.2%
AGEE	-7.8%	69.3%	X	68.2%	59.0%	76.7%	55.0%	74.6%
AGETV	-240.0%	3.3%	-215.3%	X	-29.3%	26.6%	-42.0%	20.0%

AGFRE	-162.9%	25.2%	-143.8%	22.6%	X	43.3%	-9.8%	38.1%
AGFRTV	-363.6%	-31.8	-330.0%	-36.3%	-76.3%	X	-93.6	-9.0%
AGFEE	-139.4%	31.9%	-122.0%	29.3%	8.9%	48.3%	X	43.6%
AGFETV	-325%	-20.8%	-294.1%	-25.0%	-61.6%	8.3%	-77.5%	X

Fonte: Autor

Como ilustrado no Quadro 4.2, quando é combinado o AG simples com a substituição por Tempo de Vida acontece uma redução no tempo de convergência, exemplo é o algoritmo AGEE comparado com o AGETV que obteve uma redução de 74.6%. Já nos algoritmos genéticos combinado com a Lógica Fuzzy, como o AGFEE comparado com o AGFETV, tem uma redução de 43.6%. Além disso, o AG combinado com a Lógica Fuzzy há uma redução no tempo de convergência comparado aos que não utilizam, exemplo o AGEE comparado com o AGFEE que tem uma redução no tempo de aproximadamente 55.0%, pois como dito na seção 4.1.2 há uma procura mais eficiente nas macrorregiões, o que torna mais rápido a pesquisa da solução do problema. Outro ponto importante, é a comparação com a seleção, sendo essa diferença não muito distante, tanto nos AG que aplicaram a Lógica Fuzzy ou nos AG Simples, como é observado a diferença no AGEE com o AGRE -7.8% e no AGFEE com o AGFRE 8.9%.

Portanto, o AG simples em relação ao AG auto adaptável foi a demora da convergência da população, como visto em (LINDEN, 2012), a taxa de mutação fixa tem como consequência a destruição dos indivíduos com bons *scores* e também a perda do material genético dos indivíduos que irão gerar os bons descendentes.

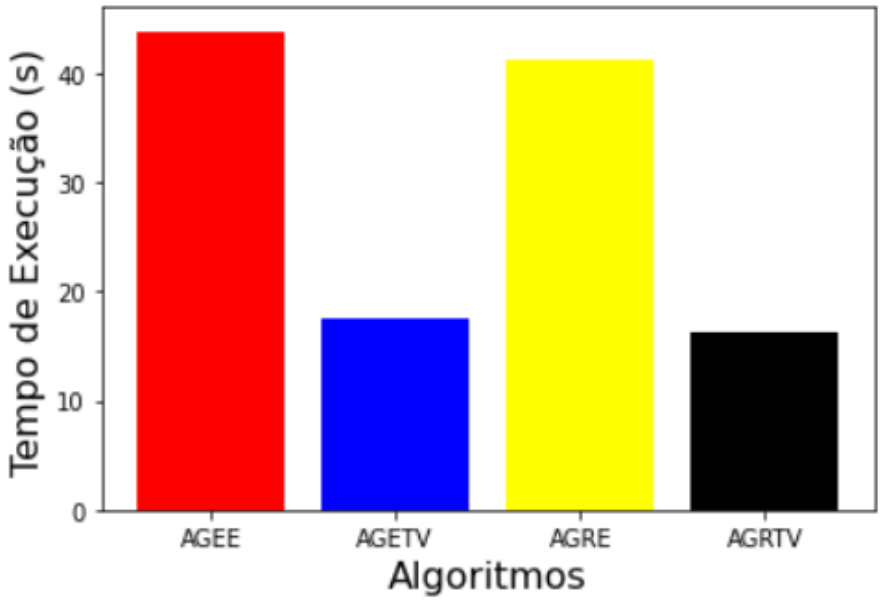
4.5.2 - Problema Combinatorial ‘01’

O problema combinatorial ‘01’ utiliza a representação binário, com o objetivo de encontrar a combinação com a maior quantidade de ‘01’ dentro da *string*, essa contagem também é utilizada como aptidão do indivíduo que funciona da seguinte maneira, uma cadeia de caracteres de tamanho par terá a combinação máxima de ‘01’ com a divisão do seu tamanho por dois. Desse modo, o problema torna-se um bom candidato para analisar o AG Simples e suas variações com a substituição e seleção, como também o AG combinado com a Lógica Fuzzy com suas devidas variações.

4.5.2.1 - Resultados Obtidos no AG Simples

Primeiro foi analisado o tempo de convergência dos algoritmos genéticos simples e suas variações como mostrado no gráfico abaixo:

Gráfico 4.11 - Tempo de Execução AG Simples (Combinatorial ‘01’)



Fonte: Autor

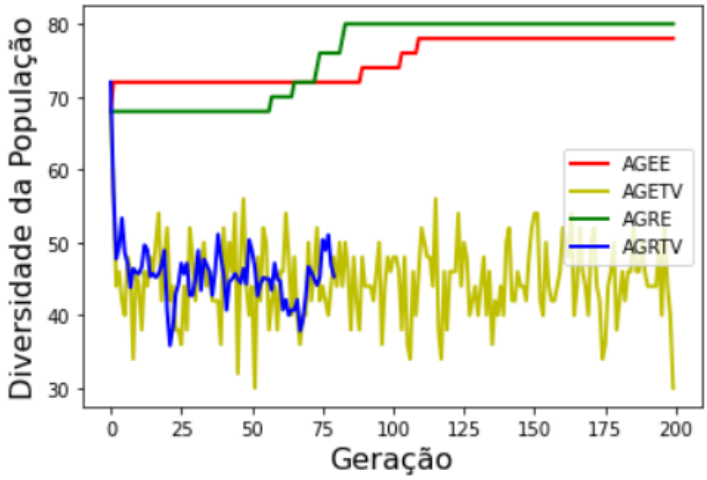
Como é exposto na Figura 4.11, o algoritmo AGEE obteve o pior desempenho, com cerca de 43.9s, quando comparado ao desempenho dos outros algoritmos genéticos simples que obtiveram AGRE 41.4s, AGETV 17.0s e o AGRTV 16.3s.

Na seção 4.1.1, mostra que os algoritmos genéticos simples com o método da substituição por Tempo de Vida que aumenta a população, ou seja, mantém os indivíduos bons e ruins por um intervalo de tempo, a qual aumentando as chances de surgir indivíduos melhores, tendo assim um desempenho melhor, o mesmo acontece no problema combinatorial ‘01’.

A diversidade da população presente em uma execução, em que foram extraídas as 200 primeiras gerações de cada algoritmo como mostrado no Gráfico 4.12. Na qual, observou-se que os Algoritmos Genéticos Simples com Elitismo na substituição apresenta uma diversidade no início igual ao do AG simples com substituição por Tempo de Vida, porém no decorrer das gerações essa diversidade muda, devido ao fato dos algoritmos AGETV e o AGRTV ter uma população maior, o que diminui a diversidade, pois aumenta a chance de um determinado indivíduo que surgir já esteja presente na população, desse modo, a diversidade presente nela não é

grande, por outro lado os algoritmos AGEE e o AGRE tem uma população reduzida, por esse motivo a quantidade de indivíduos diferentes dentro dela é grande.

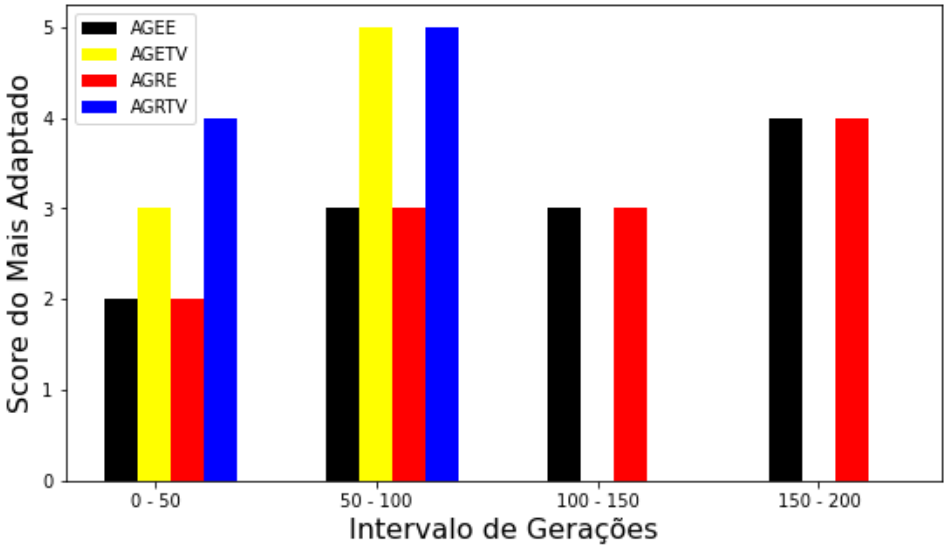
Gráfico 4.12 - Diversidade da População por Geração AG Simples (Combinatorial ‘01’)



Fonte: Autor

O Gráfico 4.13 mostra o resultado dos cromossomos mais bem adaptados nas primeiras 200 gerações, sendo elas divididas em quatro intervalos que são [0,50[,[50,100[, [100,150[, [150,200] , em que cada intervalo é mostrado o cromossomo de maior *score* presente no determinado algoritmo. Sendo destacado os AG Simples implementados com o Tempo de Vida, na qual se consegue encontrar o indivíduo ótimo mais rápido.

Gráfico 4.13 - Mais adaptado por intervalo de Geração AG Simples (Combinatorial ‘01’)

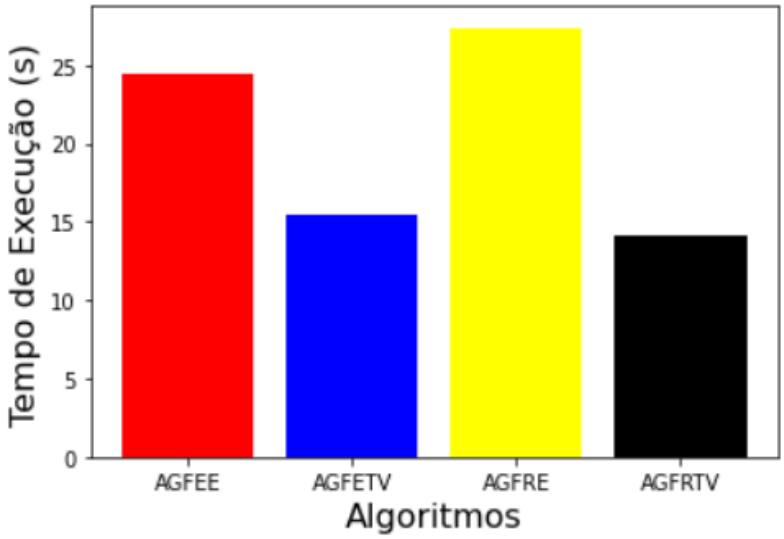


Fonte: Autor

4.5.2.2 - Resultados Obtidos no AG Proposto

Os testes realizados no AG simples foram os mesmos realizados no AG Proposto.

Gráfico 4.14 - Tempo de Execução AG Proposto (Combinatorial ‘01’)



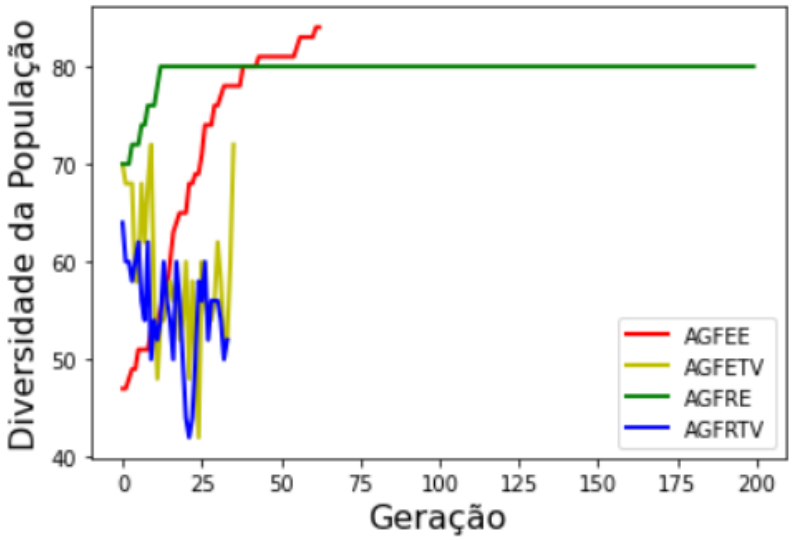
Fonte: Autor

Na Figura 4.14, o algoritmo AGFRE obteve o pior desempenho, com aproximadamente 27.4s, quando comparado ao desempenho dos outros algoritmos genéticos combinado com a Lógica Fuzzy que obtiveram AGFEE 24.4s, AGFETV 15.4s e o AGFRTV 14.1s.

Na seção 4.1.2, mostra que os algoritmos genéticos auto adaptáveis com método da substituição por Tempo de Vida têm um desempenho melhor, o mesmo acontece no problema combinatorial ‘01’.

A diversidade da população presente em uma execução, em que foram extraídas as 200 primeiras gerações de cada algoritmo como mostrado no Gráfico 4.15. Na qual, observou-se que os AG auto adaptáveis com Elitismo na substituição apresenta uma diversidade no início igual ao do AG auto adaptável com a substituição por Tempo de Vida, porém no decorrer das gerações essa diversidade muda, como explicado na seção 4.1.2.

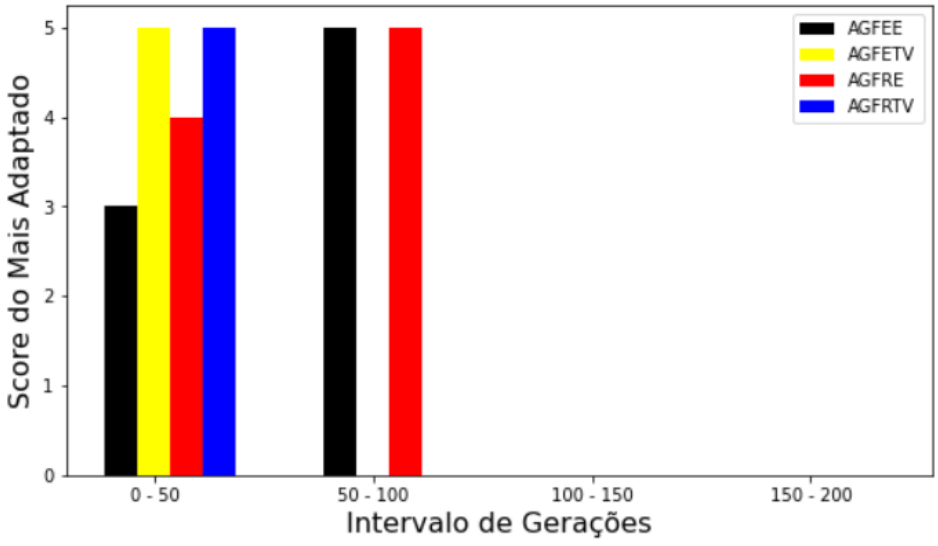
Gráfico 4.15 - Diversidade da População por Geração AG Proposto (Combinatorial ‘01’)



Fonte: Autor

O Gráfico 4.16 mostra o resultado dos cromossomos mais bem adaptados nas primeiras 200 gerações, sendo elas divididas em quatro intervalos que são [0,50[,[50,100[, [100,150[, [150,200] , em que cada intervalo é mostrado o cromossomo de maior *score* presente no determinado algoritmo. Sendo destacados os AG auto adaptáveis implementados com o Tempo de Vida, na qual se consegue encontrar o indivíduo ótimo no primeiro intervalo.

Gráfico 4.16 - Mais adaptado por intervalo de Geração AG Proposto (Combinatorial ‘01’)



Fonte: Autor

4.5.1.3 Comparação entre o AG Simples e o AG Proposto

A Quadro 4.3, está a comparação em porcentagem em relação aos AG executados no problema Combinatorial ‘01’, sendo o valor positivo indicando que foi *x%* maior que aquele AG avaliado e o valor negativo indica o quanto foi ganho, sendo realizado um truncamento para uma casa decimal.

Quadro 4.3 - Comparação do Tempo de Convergência AG Simples e AG Proposto (Combinatorial ‘01’)

	AGRE	AGRTV	AGEE	AGETV	AGFRE	AGFRTV	AGFEE	AGFETV
AGRE	X	60.6%	-6.0%	57.7%	33.8%	65.9%	41.0%	62.8%
AGRTV	-154.0%	X	-169.3%	-7.3%	-68.1%	13.5%	-49.5%	5.5%
AGEE	5.6%	62.9%	X	60.1%	37.5%	67.9%	44.4%	65.0%
AGETV	-136.6%	6.8%	-150.8%	X	-56.6%	19.4%	-39.4%	12.0%
AGFRE	-51.1%	40.5%	-60.2%	36.1%	X	48.5%	10.9%	43.7%
AGFRTV	-193.6%	-15.6%	-211.3%	-24.1%	-94.3	X	-73.0%	-9.2%
AGFEE	-69.7%	33.1%	-80.0%	28.2%	-12.2%	42.2%	X	36.9%
AGFETV	-168.8%	-5.8%	-185.1%	-13.6%	-78.0%	8.4%	-58.4%	X

Fonte: Autor

Como é mostrado no Quadro 4.3, quando é combinado o AG simples com a substituição por Tempo de Vida acontece uma redução no tempo de convergência, exemplo é o algoritmo AGEE comparado com o AGETV que obteve uma redução de 60.1%. Já nos algoritmos genéticos combinado com a Lógica Fuzzy, como o AGFEE comparado com o AGFETV, tem uma redução de 28.2%. Além disso, o AG combinado com a Lógica Fuzzy há uma redução no tempo de convergência comparado aos que não utilizam, exemplo o AGEE comparado com o AGFEE que tem uma redução no tempo de aproximadamente 44.4%, pois como dito na seção 4.1.2 há uma procura mais eficiente nas macrorregiões, o que torna mais rápido a pesquisa da solução do problema. Outro ponto importante, é a comparação com a seleção, sendo essa diferença não muito distante, tanto nos AG que aplicaram a Lógica Fuzzy ou nos AG Simples, como é observado a diferença no AGEE com o AGRE 5.8% e no AGFEE com o AGFRE -12.2%. Logo, os resultados obtidos no problema Combinatorial ‘01’ são similares ao do problema OneMax.

No próximo capítulo serão discutidas as conclusões obtidas nos experimentos anteriores.

CAPÍTULO 5

Conclusão

A partir das análises anteriores, observou-se que a o algoritmo genético com a taxa de mutação derivada das regras Fuzzy, de um modo geral, apresentou um desempenho no tempo de execução menor em comparação ao Algoritmo Genético Simples, na qual pode ser observado na seção 4.1.3 e 4.2.3, o que corrobora com o trabalho de BARCELLOS, 2000. Porém, a forma como a Lógica Fuzzy foi usada no algoritmo proposto é mais simples de implementar do que a apresentadas nesses trabalhos, o que confirma a adoção dessas modificações na sua implementação, como é apresentado nos trabalhos de (NUNES, 2018) , (CARVALHO, 2017) , (PINHO, 2007), mostrando assim a importância da modificação na taxa de mutação interativa sensível a população atual.

Além disso, foi mostrado também a importância de realizar teste com diferentes métodos de substituição e seleção, pois a redução do tempo de convergência é significativa, tanto no problema OneMax e no problema Combinatorial ‘01’, sendo essa diferença na maioria dos casos testados maior que 40%, isto é, os métodos que utilizaram a substituição por Tempo de Vida tiveram uma vantagem, porém sua diversidade foi comprometida. Já os que utilizaram o Elitismo tiveram uma diversidade maior e um tempo de convergência maior.

Logo, recomenda-se a aplicação do AG proposto a outras classes de problemas e analisar como este algoritmo se comporta. Desse modo, facilita ao desenvolvedor na tomada de decisão em quais problemas a adoção do algoritmo genético auto adaptável é recomendada.

5.1 Trabalhos Futuros

Como trabalhos futuros, destaca-se ampliar os problemas, exemplo: Caixeiro Viajante, Otimização de Funções, utilizando a metodologia adotada neste trabalho com foco na variação da taxa de mutação e variação do método da substituição e seleção para corroborar com os resultados obtidos neste trabalho. Atrelado a isso, analisar mais detalhadamente na Lógica Fuzzy as consequências da modificação da taxa de mutação em diferentes funções de representação de conhecimento como a Trapezoidal, Gaussiana.

Outra linha para trabalhos futuros, é verificar se a metodologia aplicada aqui pode ser estendida para todos os tipos de taxas como a de cruzamento e a variação do tamanho da população. Assim, poderá ser constatada a importância da modificação dos parâmetros durante a execução do algoritmo genético de forma iterativa.

Referências Bibliográficas

ARABAS, Jaroslaw; MICHALEWICZ, Zbigniew; MULAWKA, Jan. GAVaPS-a genetic algorithm with varying population size. In: **Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence**. IEEE, 1994. p. 73-78.

BARCELLOS, J. C. H. **Algoritmos genéticos adaptativos: um estudo comparativo**. Dissertação (Mestrado em Engenharia) — Escola Politécnica da Universidade de São Paulo, 2000.

BARG, EDUARDO KLAUS. **Protótipo de um Controlador de Temperatura Baseado em Lógica Fuzzy Utilizando um Microcontrolador**. 2002. Monografia (Bacharel) - Ciências da Computação, [S. l.], 2002.

BEDREGAL, Benjamín R. Callejas. **Introdução à Lógica Fuzzy**. [*S. l.*], 16 out. 2013. Disponível em: https://wp.ufpel.edu.br/weit/files/2013/03/Tutorial-WEIT2013.pdf. Acesso em: 18 maio 2022.

BRITO, Felipe Houat de et al. **GAIA-Uma proposta de agente inteligente acoplado a algoritmos evolucionários para ajuste dinâmico de parâmetros através de conceitos de design inteligente e lógica fuzzy**. 2011.

BURDELIS, Mauricio Alexandre Parente. **Ajuste de Taxas de Mutação e de Cruzamento de Algoritmos Genéticos Utilizando-se Inferências Nebulosas**. 2009. Dissertação (Mestre em Engenharia) - Departamento de Engenharia de Computação e Sistemas Digitais (PCS), [S. l.], 2009.

CARVALHO, W. L. O. **Estudo de parâmetros ótimos em algoritmos genéticos elitistas**. Dissertação (Mestrado em Matemática Aplicada e Estatística) — Universidade Federal do Rio Grande do Norte, Natal, 2017.

CAMARGO, G. d. M. **Controle da pressão seletiva em algoritmo genético aplicado a otimização de demanda em infra-estrutura aeronáutica**. Master's thesis, Escola Politécnica, Universidade de São Paulo. 2006.

CHEBBI, O.; CHAOUACHI, J. **Effective parameter tuning for genetic algorithm to solve a real world transportation problem**. International Conference on Methods and Models in Automation and Robotics, p. 370—-375, 2015.

COPPIN, B. **Inteligência Artificial**. Rio de Janeiro: LTC, 2010.

COX, Earl. **The fuzzy systems handbook**. New York: AP Professional, 1994.

EBERHART, R.; SIMPSON, P.; DOBBINS, R. **Computational intelligence PC tools: an indispensable resource for the latest in fuzzy logic, neural network and evolutionary computing**. American Press Inc., 1996.

DARWIN, Charles, **c** Disponível em https://www.literature.org/authors/darwin-charles/the-origin-of-species/ . Acesso em 06. jun. 2022

EIBEN, A. E. e Smith, J. E. **Introduction to Evolutionary Computing** . Springer Verlag. 2007.

EIBEN, A. E.; SMIT, S. K. **Parameter tuning for configuring and analyzing evolutionary algorithms**. Swarm and Evolutionary Computation, p. 19–31, 2011.

FERRARI, ALLAN C. KRAINSKI; LEANDRO, GIDEON VILLAR; OLIEVEIRA, GUSTAVO. **Evolução Diferencial com Parâmetros Ajustáveis por Lógica Fuzzy.** Anais do XX Congresso Brasileiro de Automática, [S. l.], p. 796-803, 24 set. 2014.

FREITAS, Leandro Martins. **EasyGA: algoritmos genéticos para iniciantes**. 2019. Monografia (Bacharel) - Instituto de Computação, Universidade Federal de Alagoas, [S. l.], 2019.

GIGUERE, Philippe; GOLDBERG, David E. **Population sizing for optimum sampling with genetic algorithms: A case study of the Onemax problem**. Genetic Programming, v. 98, p. 496-503, 1998.

JANG JS Roger. MATLAB: **Fuzzy logic toolbox user's guide: Version 1**. Math Works, 1997.

JAYNES, E.T. e G.L. Bretthorst: **Probability Theory: The Logic of Science**. Cambridge University Press, 2003, ISBN 9780521592710.

KATO, Paiva & Izidoro. **Algoritmos Genéticos**. In: BIOINFO - Revista Brasileira de Bioinformática e Biologia Computacional. 1ª ed.; vol. 1. Lagoa Santa: Editora Alfahelix, 2021. DOI: 10.51780/978-6-599-275326

KIM, K.-j.; HAN, I. Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. **Expert systems with Applications**, Elsevier, v. 19, n. 2, p. 125–132, 2000

LINDEN, R. **Algoritmos genéticos**. 3. ed. Rio de Janeiro: Editora Ciência Moderna, 2012.

LUCAS, Diogo C. **Algoritmos genéticos: uma introdução**. Apostila referente à disciplina de Inteligência Computacional, Universidade Federal do Rio Grande do Sul, Brasil, 2002.

MITCHELL, M. **An introduction to genetic algorithms**. MIT Press, 1999.

NUNES, Jancleiton Rodrigues de Oliveira. **Avaliação de Taxas de Cruzamento e Mutação em um Algoritmo Genético Baseado em Ordem Aplicado ao Problema**

do Caixeiro Viajante. 2018. Monografia (Bacharel) - Curso de Bacharelado em Sistemas de Informação, [S. l.], 2018.

PACHECO, Roberto C. S. **Tratamento de imprecisão em sistemas especialistas**. 1991. 85 f. Dissertação (Mestrado em Engenharia de Produção) – Engenharia de Produção e Sistemas, UFSC, Florianópolis.

PINHO, A. F.; MONTEVECHI, J. A. B.; MARINS, F. A. S. **Análise da aplicação de projeto de experimentos nos parâmetros dos algoritmos genéticos**. SISTEMAS GESTÃO, v. 2, n. 3, p. 319–331, 2007.

POZO, Aurora et al. **Computação evolutiva**. Universidade Federal do Paraná, 61p.(Grupo de Pesquisas em Computação Evolutiva, Departamento de Informática-Universidade Federal do Paraná), 2005.

ROSA, T. O.; LUZ, H. S. Conceitos básicos de algoritmos genéticos: teoria e prática. **Anais do XI encontro de estudantes de informática do Tocantins**, p. 27–37, 2009.

ROSS, Timothy J. **Fuzzy logic with engineering applications**. New York: McGraw-Hill, 1995.

SONI, N. e KUMAR, T. **Study of various mutation operators in genetic algorithms**. volume 5, pp. 4519–4521. 2014.

TARIG, Ali Abdurrahman E. S. **Controle de um braço robótico utilizando uma abordagem de agente inteligente**. 2001. 98 f. Dissertação (Mestrado em Informática) - Coordenação Pós-Graduação em Informática, Universidade Federal da Paraíba, João Pessoa.

TEIXEIRA, O. N. **Proposta de um novo algoritmo genético baseado na teoria dos jogos**. Programa de Pós-Graduação em Engenharia Elétrica, UFPA, 2005.

UMBARKAR, A. J. , P. D. S. **Crossover operators in genetic algorithms:a review**. ICTACT Journal on Soft Computing, 6(1):1083–1092 . 2015.

VILHENA VIEIRA LOPES, Roberta. **Um algoritmo genético baseado em tipos abstratos de dados e sua especificação em Z**. 2003. Tese (Doutorado em Informática) - Pós-Graduação em Ciência da Computação, Universidade Federal de Pernambuco, Recife.

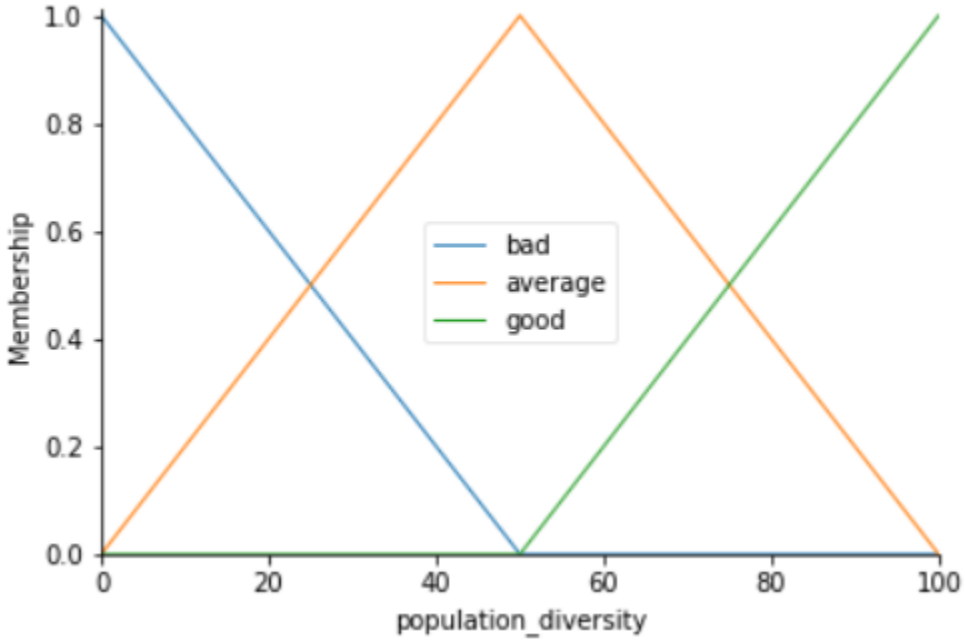
WANG, Wen-June. **New similarity measures on fuzzy sets and on elements**. Fuzzy sets and systems, v. 85, n. 3, p. 305-309, 1997.

Z. Michalewicz. **Genetic algorithms + data structures= evolution programs**. In: Computational Statistics. Springer-Verlag, 1996. p. 372-373.

ZINI, Érico & Neto, Alfredo & Garbelini, Enio. **ALGORITMO MULTIOBJETIVO PARA OTIMIZAÇÃO DE PROBLEMAS RESTRITOS APLICADOS A INDÚSTRIA**. 365-374. 10.5151/mathpro-cnmai-0064. 2015.

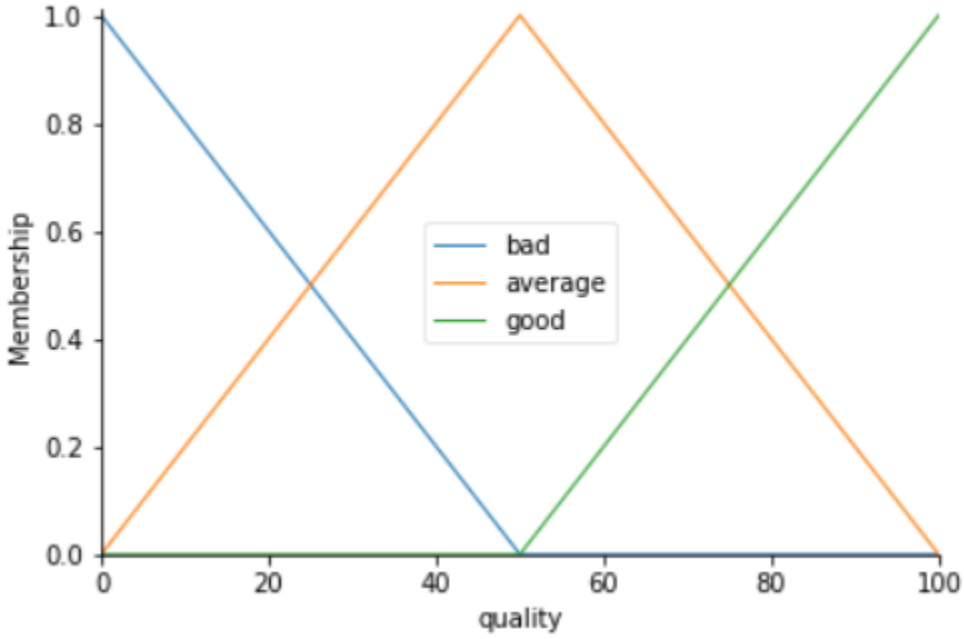
Apêndice

Apêndice A - Representação Gráfica Diversidade Populacional (Conjunto Fuzzy [“*bad*”, "*average*", “*good*”])



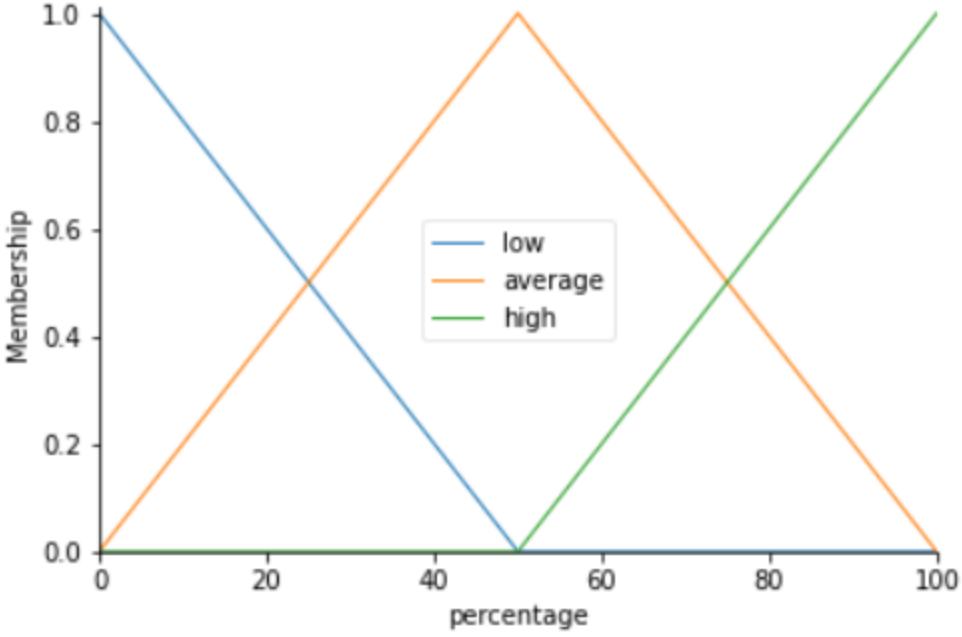
Fonte: Autor

Apêndice B - Representação Gráfica Qualidade (Conjunto Fuzzy [“*bad*”, "*average*", “*good*”])



Fonte: Autor

Apêndice C - Representação Gráfica Porcentagem de Mutação (Conjunto Fuzzy [“*low*”, "*average*", “*high*”])



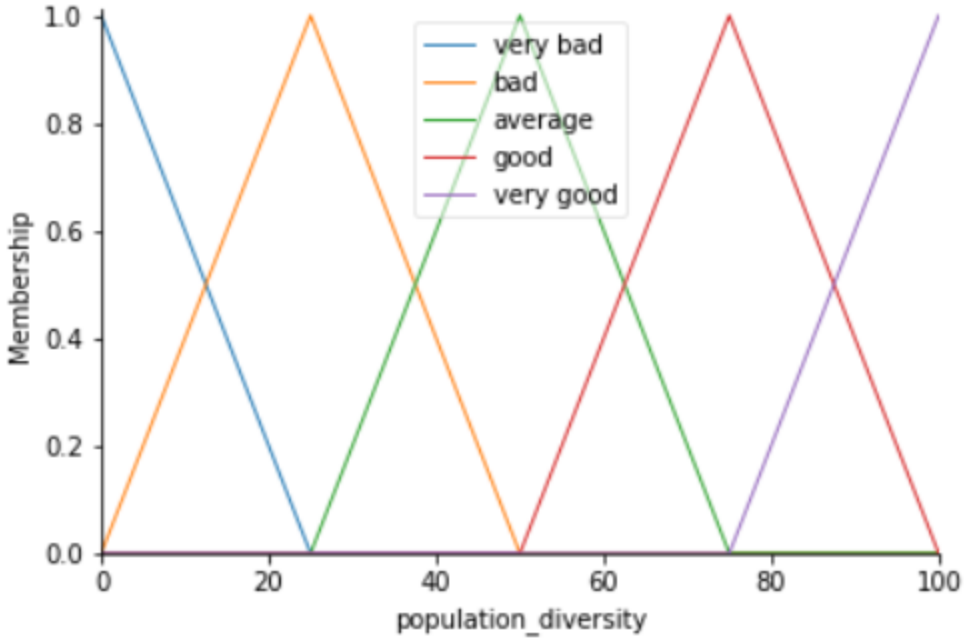
Fonte: Autor

Apêndice D - Regras Teste 1

Regra	Se	Qualidade	E	DP	Então	PM
1	Se	*bad*	E	*bad*	Então	*high*
2	Se	*bad*	E	*average*	Então	*high*
3	Se	*bad*	E	*good*	Então	*average*
4	Se	*average*	E	*bad*	Então	*high*
5	Se	*average*	E	*average*	Então	*average*
6	Se	*average*	E	*good*	Então	*average*
7	Se	*good*	E	*bad*	Então	*low*
8	Se	*good*	E	*average*	Então	*low*
9	Se	*good*	E	*good*	Então	*low*

Fonte: Autor

Apêndice E - Representação Gráfica Diversidade Populacional (Conjunto Fuzzy [“*very bad*”, “*bad*”, "*average*", “*good*”, “*very good*”])



Fonte: Autor

Apêndice F - Regras Teste 2

Regra	Se	Qualidade	E	DP	Então	PM
1	Se	*bad*	E	*very bad*	Então	*high*
2	Se	*bad*	E	*bad*	Então	*high*
3	Se	*bad*	E	*average*	Então	*average*
4	Se	*bad*	E	*good*	Então	*low*
5	Se	*bad*	E	*very good*	Então	*high*
6	Se	*average*	E	*very bad*	Então	*high*
7	Se	*average*	E	*bad*	Então	*average*
8	Se	*average*	E	*average*	Então	*average*
9	Se	*average*	E	*good*	Então	*low*
10	Se	*average*	E	*very good*	Então	*lolw*
11	Se	*good*	E	*very bad*	Então	*high*
12	Se	*good*	E	*bad*	Então	*low*
13	Se	*good*	E	*average*	Então	*low*
14	Se	*good*	E	*good*	Então	*low*
15	Se	*good*	E	*very good*	Então	*low*

Fonte: Autor

Apêndice G - Regras Teste 3

Regra	Se	Qualidade	E	DP	Então	PM
1	Se	*very bad*	E	*very bad*	Então	*high*
2	Se	*very bad*	E	*bad*	Então	*high*
3	Se	*very bad*	E	*average*	Então	*high*
4	Se	*very bad*	E	*good*	Então	*high*
5	Se	*very bad*	E	*very good*	Então	*high*
6	Se	*bad*	E	*very bad*	Então	*high*
7	Se	*bad*	E	*bad*	Então	*high*
8	Se	*bad*	E	*average*	Então	*average*

9	Se	*bad*	E	*good*	Então	*low*
10	Se	*bad*	E	*very good*	Então	*high*
11	Se	*average*	E	*very bad*	Então	*high*
12	Se	*average*	E	*bad*	Então	*average*
13	Se	*average*	E	*average*	Então	*average*
14	Se	*average*	E	*good*	Então	*low*
15	Se	*average*	E	*very good*	Então	*low*
16	Se	*good*	E	*very bad*	Então	*how*
17	Se	*good*	E	*bad*	Então	*low*
18	Se	*good*	E	*average*	Então	*low*
19	Se	*good*	E	*good*	Então	*low*
20	Se	*good*	E	*very good*	Então	*low*
21	Se	*very good*	E	*very bad*	Então	*low*
22	Se	*very good*	E	*bad*	Então	*low*
23	Se	*very good*	E	*average*	Então	*low*
24	Se	*very good*	E	*good*	Então	*low*
25	Se	*very good*	E	*very good*	Então	*low*

Fonte: Autor