MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21b/2016/02.02.12.42-TDI

OTIMIZAÇÃO DA ALOCAÇÃO DE EXPERIMENTOS NA CARGA ÚTIL DO FOGUETE DE SONDAGEM UTILIZANDO INTELIGÊNCIA COMPUTACIONAL HÍBRIDA

Elder Figueiredo

Dissertação de Mestrado do Curso de Pós-Graduação em Computação Aplicada, orientada pelo Dr. Lamartine Nogueira Frutuoso Guimarães, aprovada em 18 de fevereiro de 2016.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34P/3L4P8DB>

INPE
São José dos Campos
2016

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GB)
Serviço de Informação e Documentação (SID)
Caixa Postal 515 - CEP 12.245-970
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/6921
Fax: (012) 3208-6919
E-mail: pubtc@inpe.br

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21b/2016/02.02.12.42-TDI

OTIMIZAÇÃO DA ALOCAÇÃO DE EXPERIMENTOS NA CARGA ÚTIL DO FOGUETE DE SONDAGEM UTILIZANDO INTELIGÊNCIA COMPUTACIONAL HÍBRIDA

Elder Figueiredo

Dissertação de Mestrado do Curso de Pós-Graduação em Computação Aplicada, orientada pelo Dr. Lamartine Nogueira Frutuoso Guimarães, aprovada em 18 de fevereiro de 2016.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34P/3L4P8DB>

INPE
São José dos Campos
2016

Dados Internacionais de Catalogação na Publicação (CIP)

Figueiredo, Elder.

F469o Otimização da alocação de experimentos na carga útil do foguete de sondagem utilizando inteligência computacional híbrida / Elder Figueiredo. – São José dos Campos : INPE, 2016.
xxiv + 100 p. ; (sid.inpe.br/mtc-m21b/2016/02.02.12.42-TDI)

Dissertação (Mestrado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2016.
Orientador : Dr. Lamartine Nogueira Frutuoso Guimarães.

1. Foguete de sondagem. 2. Inteligência computacional. 3. Otimização. 4. Carga útil. I.Título.

CDU 629.765:004

Aluno (a): *Elder Figueiredo*

Título: " OTIMIZAÇÃO DA ALOCAÇÃO DE EXPERIMENTOS NA CARGA ÚTIL DO FOGUETE DE SONDAGEM UTILIZANDO INTELIGÊNCIA COMPUTACIONAL HÍBRIDA".

Aprovado (a) pela Banca Examinadora em cumprimento ao requisito exigido para obtenção do Título de *Mestre* em *Computação Aplicada*

Dr. Reinaldo Roberto Rosa

Presidente / INPE / SJCampos - SP

Dr. Lamartine Nogueira Frutuoso Guimarães

Orientador(a) / IEAv/CTA / SJCampos - SP

Dr. Solon Venâncio de Carvalho

Membro da Banca / INPE / SJCampos - SP

Dr. Stephan Stephany

Membro da Banca / INPE / SJCampos - SP

Dr. Valdir Gil Pillat

Convidado(a) / UNIVAP / São José dos Campos - SP

Este trabalho foi aprovado por:

( ) maioria simples

(X) unanimidade

São José dos Campos, 18 de Fevereiro de 2016

“Acredite que você pode, assim você já está no meio do caminho”.

Theodore Roosevelt

A meus pais, Claudinei (In Memoriam) e Tânia, que sempre me incentivaram e me ensinaram como tornar o estudo um prazer, e não uma obrigação.

AGRADECIMENTOS

Primeiramente ao meu orientador Lamartine Nogueira Frutuoso Guimarães, pela confiança depositada em mim, sua experiência, sua competência e disponibilidade em contribuir ativamente deste trabalho.

Aos professores do LAC-INPE, pela contribuição de conhecimento e incentivo, em especial aos professores Stephan Stephany e Reinaldo Roberto Rosa, responsáveis pelas orientações iniciais sobre a metodologia de trabalho da pós-graduação deste instituto.

Agradeço a contribuição dos servidores do Instituto de Aeronáutica e Espaço, localizado no Departamento de Ciência e Tecnologia Aeroespacial em São José dos Campos, por contribuir com meu conhecimento em foguetes para o desenvolvimento deste trabalho. Ao meu grupo de trabalho, pelo apoio e incentivo em aprimorar meus conhecimentos de forma a contribuir para as atividades do IAE. Em especial, ao Eng. Eduardo Dore Roda, que foi o primeiro contato e que disponibilizou sua equipe para atendimento às dúvidas relacionadas à área espacial, também ao TCel. Alexandre Nogueira Babosa, por sua presteza no atendimento e de sua equipe de dinâmica de voo, entre os quais: Alexandre Garcia, Guilherme da Silveira e Marcos Alécio dos Santos Romani, que contribuíram com seus conhecimentos e bibliografias importantes sobre o comportamento e análise de estabilidade de foguetes. Também, ao Claudinei José de Castro, da equipe de Integração e Ensaios, auxiliando no entendimento dos dados dos ensaios da carga útil.

A minha noiva, Rosy Gusmão, pelo incentivo e apoio em todos os momentos, principalmente nos mais críticos.

RESUMO

A alocação de experimentos na carga útil de um foguete de sondagem tem influência direta no seu comportamento durante o voo, podendo afetar sua performance e trajetória. Desta forma, o processo de distribuição de massas na carga útil deve ser planejado, com o objetivo de minimizar desbalanceamentos. Nos casos onde o desbalanceamento não pode ser otimizado, são adicionadas cargas para correção. Atualmente, esta distribuição é realizada por engenheiros, que com a base de conhecimento prévio, faz o planejamento da alocação em algumas semanas. A verificação do resultado final é primeiramente feita em um programa de desenho tridimensional. Após a fabricação, a carga útil montada é colocada em um equipamento que medirá o desbalanceamento residual. Caso necessário, são adicionados os lastros, para minimizar o desbalanceamento. Em cada missão de lançamento, diferentes experimentos são embarcados, sendo necessária uma nova análise e novos posicionamentos para atender aos requisitos. Este trabalho desenvolve uma ferramenta utilizando inteligência computacional para realizar este processo. Fornecendo os dados de entrada de cada experimento, o sistema realiza a busca por uma solução viável de alocação, otimizando os parâmetros envolvidos no balanceamento de cargas. Isto reduz o tempo gasto no projeto de leiaute dos experimentos e minimiza a necessidade de utilização de lastros.

EXPERIMENTS ALLOCATION OPTIMIZATION ON THE PAYLOAD OF SOUNDING ROCKETS USING HYBRID COMPUTATIONALINTELLIGENCE

ABSTRACT

The allocation of experiments in the payload of a sounding rocket has a direct influence on their behavior during the flight, which may affect its performance and trajectory. Thus, the mass distribution in the payload process must be planned in order to minimize unbalances. In cases where the unbalance cannot be optimized, charges for correction are added. Currently, this distribution is executed by engineers, who with the prior knowledge, do the planning the allocation in a few weeks. The final result is checked in a three-dimensional drawing program. After fabrication, the mounted payload is placed in a equipment that measures the residual unbalance. In this step, the ballasts are added, if necessary, to minimize the unbalance. In each release of mission, different experiments are embedded, requiring a new analysis and new positions to meet the requirements. This paper develops a tool using computational intelligence to realize this process. Providing the experiments data to the system, it performs a search for a viable allocation, optimizing the parameters involved in load balancing. This reduces the time spent on the layout design of the experiments and minimizes the need to use charges.

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

LISTA DE SIGLAS E ABREVIATURAS

AE	Algoritmos Evolutivos
AEP	Programa Allocator Experiments in the Payload
AG	Algoritmos Genéticos (*Genetic Algorithms*)
CAD	Computer Aided Design (programa de desenho auxiliado por computador)
CG	Centro de Gravidade
CP	Centro de Pressão
DCTA	Departamento de Ciência e Tecnologia Aeroespacial
DOF	"Degree of fulfillment" - Grau de pertencimento
DNA	Ácido Desoxirribonucleico
E	Margem Estática
IC	Inteligência Computacional
INPE	Instituto Nacional de Pesquisas Espaciais
MI	Momento de Inércia
MO	Multi-Objetivo
MOEA	Multi-Objective Evolutionary Algorithm
MOOP	Multi-Objective Optimization Problem
NGMOGA	*non-generational multi-objective genetic algorithm* (algoritmo genético multiobjetivo não-geracional)
PI	Produto de Inércia
VSB-30	Veículo de Sondagem Suborbital

LISTA DE SÍMBOLOS

$\preceq$ Dominância

α Alfa: parâmetro que define a extensão do ponto de cruzamento

β Beta: valor que define o grau de dispersão no ponto de cruzamento

θ Ângulo de inclinação do eixo de rolamento do foguete

λ Coordenada baricêntrica do triângulo

SUMÁRIO

1 INTRODUÇÃO

1.1. Considerações iniciais

Explorar economicamente o espaço, principalmente da órbita da Terra, tem como objetivo propiciar comunicação e monitoramento via satélite. Antes, porém, é necessário conhecer o ambiente, procurando desenvolver materiais e equipamentos que operem nas condições encontradas na órbita da Terra. Para permitir este fim utilizam-se os foguetes de sondagem. No caso do Brasil, um dos foguetes utilizados é o VSB-30. Este foguete é um veículo suborbital, composto por dois motores de propulsão sólida, e um compartimento, determinado de carga-útil, que leva os sensores e demais experimentos embarcados. Desta forma, torna-se importante a distribuição da carga no veículo, a fim de garantir o comportamento aerodinâmico, estabilidade, confiabilidade e segurança do lançamento e do voo.

1.2. Motivação

A alocação dos experimentos tem influência direta na estabilidade de um foguete. É necessário minimizar o desbalanceamento em cada plataforma que recebe os experimentos, visando garantir parâmetros de estabilidade adequados da carga útil, e propiciando que após sua integração com os motores responsáveis pela propulsão, este sistema atenda a determinados parâmetros especificados em projeto. No processo de alocação dos experimentos, os parâmetros de interesse são: centro de gravidade, produto de inércia e o ângulo de inclinação do eixo. Na solução deste problema, existem diversas distribuições possíveis dos experimentos, porém somente algumas atenderão os requisitos de qualidade, confiabilidade e segurança do voo. A elaboração de modelos matemáticos para determinar a melhor condição de distribuição é um processo que demanda muito tempo de modelamento e processamento. Cada voo necessita de um modelo diferente devido às mudanças dos experimentos embarcados. Atualmente uma equipe de engenheiros é responsável por planejar e avaliar possíveis configurações de

alocação. Utilizando um programa computacional de desenho tridimensional, calculam-se os parâmetros da carga útil. Caso a solução não atenda aos requisitos, uma nova configuração é elaborada, simulada e avaliada. Este passo se repete até que seja obtida uma configuração de alocação minimamente adequada dos experimentos.

1.3. Objetivo

O objetivo geral desta dissertação é utilizar paradigmas de inteligência computacional para obter, em tempo viável, parâmetros adequados e satisfatórios para alocação dos equipamentos. Esta solução também pode contribuir para evitar que uma má distribuição possa requerer alterações de características do veículo, como por exemplo, nas formas das empenas ou utilização de lastros para corrigira estabilidade.

O paradigma computacional selecionado para otimizar esta alocação é o algoritmo genético. Este paradigma se utiliza do processo evolutivo idealizado por Charles Darwin, passando-se pelas etapas de geração de uma população inicial, seleção e reprodução, as quais são repetidas até se chegar a uma geração mais apta a representar a melhor solução para o problema. A lógica fuzzy também é utilizada para atribuir o grau de aptidão de um indivíduo nesta otimização. A união destas tecnologias visa permitir ao algoritmo maior eficiência, proporcionando resultados mais adequados. A esta estratégia de combinar as técnicas de inteligência computacional denomina-se como hibridização.

O algoritmo genético desenvolvido neste trabalho busca as melhores configurações de alocação dos experimentos, que propicie um valor de balanceamento estático e dinâmico das plataformas da carga útil, atendendo os requisitos de um voo estável do veículo.

1.4. Organização do documento

A dissertação está organizada em 6 capítulos. No primeiro Capítulo foi apresentada a introdução deste trabalho. O Capítulo 2 realiza uma breve apresentação dos conceitos e tecnologias de foguetes. No capítulo 3 é apresentado os parâmetros de balanceamento de massa em foguetes de sondagem, que é o objetivo deste trabalho. O Capítulo 4 faz uma revisão bibliográfica sobre inteligência computacional, otimização multiobjetivo, fronteira de Pareto, algoritmo genético e lógica fuzzy. O Capítulo 5 traz o desenvolvimento do trabalho e os resultados obtidos. Finalmente, são apresentadas a conclusão e as referências bibliográficas, respectivamente.

2 FOGUETE DE SONDAGEM DE INTERESSE: VSB-30

2.1. Introdução

Os foguetes de sondagem têm como objetivo levar uma carga útil até a altitude requerida, ou prover certa permanência nesta altitude, com a precisão especificada a partir de um campo de lançamento (PALMÉRIO, 2008). Na última década, houve um aumento da demanda por foguetes de sondagem para a realização de experimentos em ambiente de microgravidade. Os testes de equipamentos para utilização no espaço são realizados através de voos suborbitais, reduzindo o custo dos testes e dando confiabilidade nos resultados.

O veículo suborbital, conhecido como VSB-30, utilizado como caso de estudo para este trabalho, possui dois propulsores sólidos, a carga útil e interfaces de montagem. Os motores são divididos em dois estágios. No momento do lançamento é dado ignição no primeiro estágio. Ao sair da rampa de lançamento, são ativados os propulsores de rolamento responsáveis por iniciar a rotação do foguete em torno de seu próprio eixo. No momento que encerra a queima do combustível do primeiro estágio, este se separa do restante do veículo, retornando para o solo em queda livre. Neste momento ocorre a ignição do motor do segundo estágio. No final da queima do combustível deste segundo estágio, um sistema mecânico retira a rotação do foguete. Logo em seguida ocorre a separação do segundo estágio, restando apenas a carga útil no ambiente de microgravidade. A carga útil fica neste ambiente por cerca de seis minutos, e devido à gravidade da Terra, retorna à atmosfera entrando em queda livre até o solo, onde será resgatada e realizada a análise de dados dos experimentos. O modelo de foguete abordado neste trabalho está apresentado na Figura 2.1.

Figura 2.1 - Estrutura do foguete de sondagem VSB-30.

Este foguete tem a capacidade de transportar cargas de até 400 kg, na faixa de 270 km de altitude (INSTITUTO DE AERONÁUTICA E ESPAÇO, 2013). Dos diversos parâmetros existentes, este trabalho trata aqueles relacionados à montagem da carga útil, e suas influências na dinâmica do voo do foguete. O VSB-30 foi escolhido como estudo de caso devido a algumas características que viabilizam este trabalho, entre elas: o sucesso obtido nos lançamentos já realizados, a existência de dados da carga útil para análise e comparação de resultados e por possuir futuras missões que propiciarão a aplicação do algoritmo deste trabalho.

A estabilidade de um foguete pode ser interpretada como sua capacidade de manter uma trajetória durante o voo. O valor do centro de pressão (CP) é definido como o centro de área da projeção do veículo, sendo um dado estabelecido no projeto de um foguete, e neste trabalho tem seu valor constante. O valor do centro de gravidade (CG) é obtido conhecendo-se a massa do foguete de sondagem completo, isto é, a massa dos motores e da carga útil acopladas. As massas dos motores são tratadas como constantes. A distribuição de experimentos da carga útil, que é discussão deste trabalho, é diferente em cada voo, e são responsáveis pela alteração do valor do centro de gravidade total do veículo. Para avaliar a estabilidade utiliza-se um parâmetro denominado margem estática. A margem estática é calculada levando-se em consideração os valores de centro de gravidade e o centro de pressão do foguete conforme equação:

$$E = \frac{(CP-CG)}{d_{max}}\,, \qquad (2.1)$$

onde,

E é a margem estática,

d_{max} é o maior diâmetro entre os motores existentes,

CP é o valor do centro de pressão e,

CG é o centro de gravidade final.

Para um voo seguro e com trajetória que atenda os requisitos da missão o comportamento desejável deve ser estável. Os comportamentos de voo do foguete em relação à sua estabilidade estão apresentados nas Figuras 2.2 e 2.3.

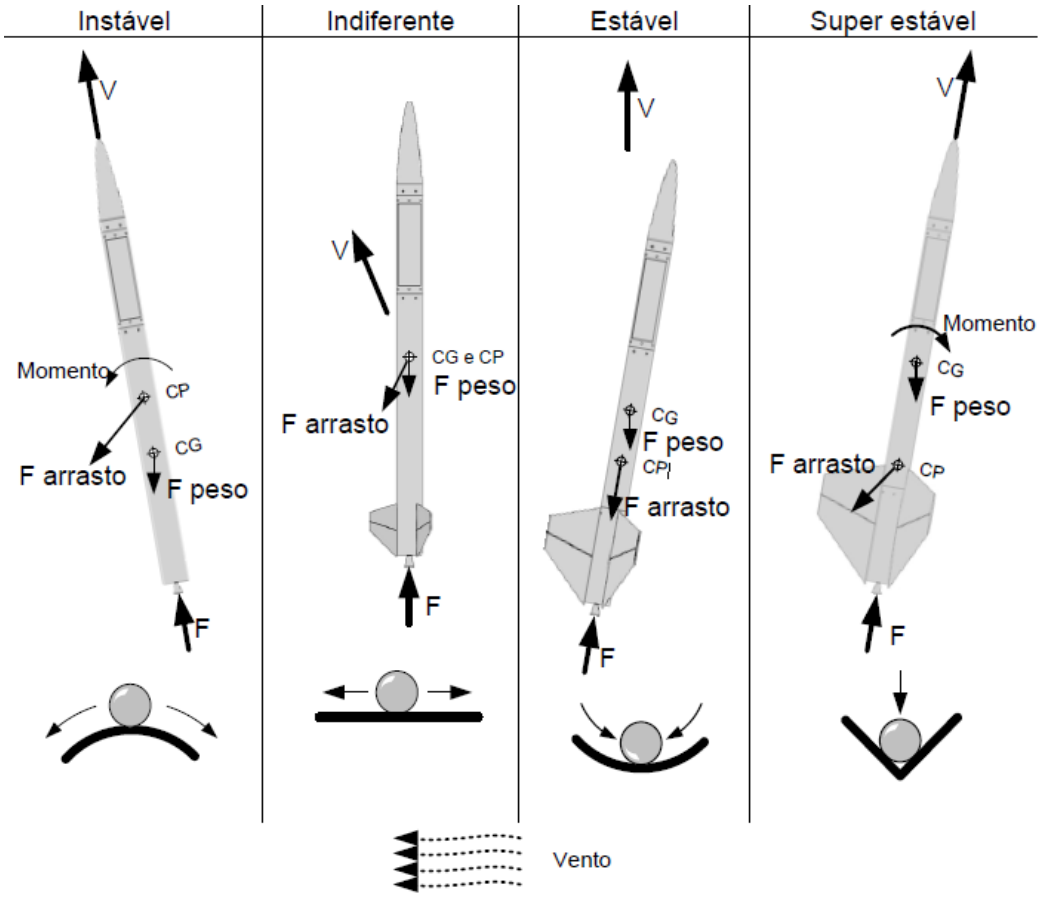


Figura 2.2 - Diagramas de corpo livre de foguetes com diferentes configurações aerodinâmicas.

Fonte: Porto (2007).

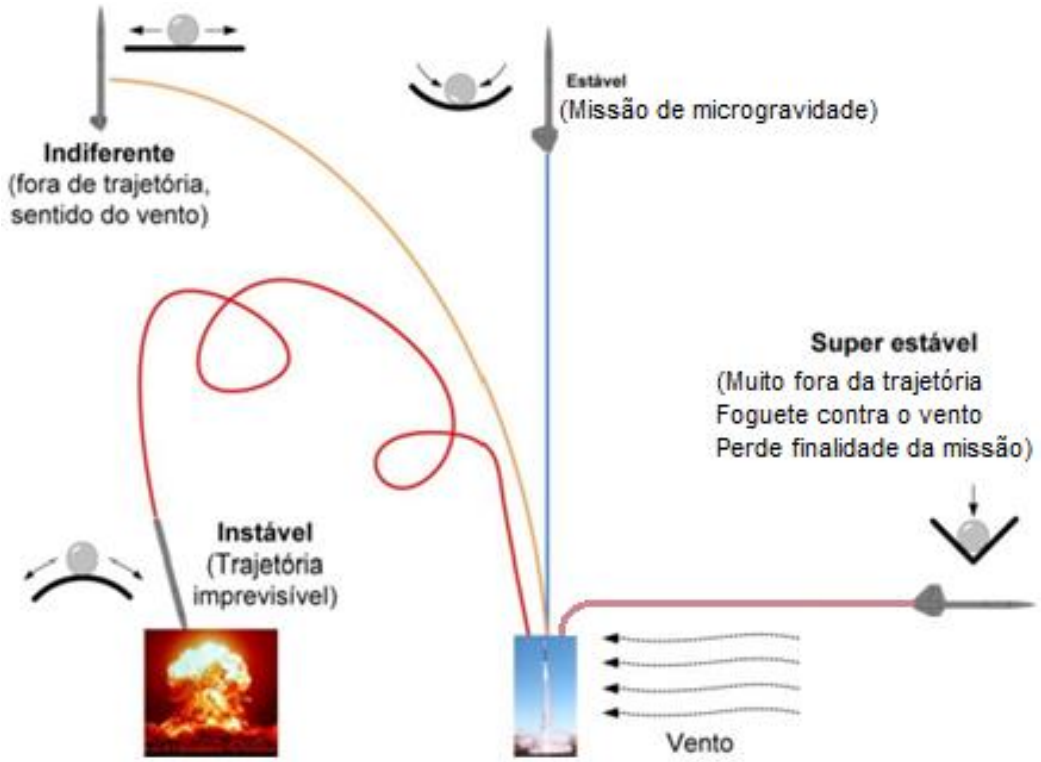


Figura 2.3 - Trajetórias das configurações aerodinâmicas.
Fonte: adaptado de Porto (2007).

Para que o foguete deste trabalho voe com a configuração estável a margem estática deve situar-se entre 1,3 e 1,5. Este valor é definido no momento de lançamento, pois sofrerá alterações do centro de gravidade durante o voo, isto devido à redução da massa dos motores. Contudo, a margem estática estará sempre em valores adequados para garantir a estabilidade do foguete.

2.2. Carga útil

A carga útil é o conjunto de módulos existentes acima do último motor do foguete, neste caso após o motor do segundo estágio. A Figura 2.4 apresenta a carga útil com seus módulos, interfaces e coifa. Dentro de cada módulo da carga útil existem as plataformas com os experimentos embarcados. A estrutura, a geometria e a distribuição de massas da carga útil são determinantes do comportamento do foguete, no que diz respeito à

aerodinâmica, à consequente estabilidade estática e dinâmica, ao comportamento estrutural e ao desempenho (PALMÉRIO, 2008).

Figura 2.4 - Modelo de carga útil.

Os módulos são compartimentos que guardam os experimentos. Cada módulo possui duas plataformas, classificadas como inferior e superior. Nas plataformas é que são fixados os experimentos. Os anéis de interface são responsáveis por interligar os módulos, isto é, permitir conexões de energia elétrica e de comunicação entre experimentos. A coifa tem a responsabilidade de promover o melhor compromisso aerodinâmico, propiciar o menor arrasto possível e evitar que fenômenos aerodinâmicos possam comprometer a sua estrutura.

2.3. Movimentos do foguete

Os movimentos de um veículo no espaço são definidos por três eixos cartesianos, ver Figura 2.5. Estes movimentos em torno de cada eixo recebem um nome, definido como:

- rolamento ("roll") a rotação em torno do eixo X
- arfagem ("pitch") a rotação em torno do eixo Y;
- guinada ("yaw") a rotação em torno do eixo Z.

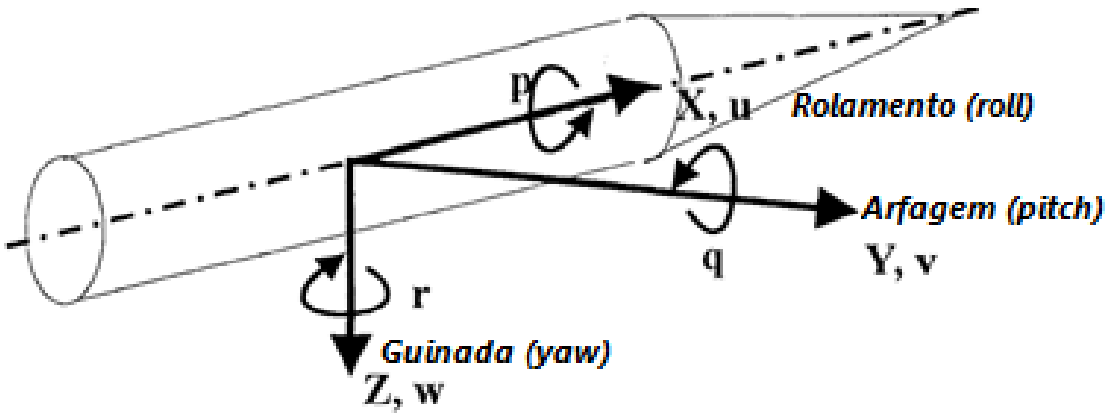


Figura 2.5 - Graus de liberdade do foguete no sistema cartesiano.
Fonte: Palmério (2008).

O único movimento permitido e necessário ao foguete é o rolamento. O movimento adequado de rolamento do foguete está diretamente condicionado à sua distribuição de massa. O balanceamento final da carga útil é formado por três parâmetros: centro de gravidade, momento de inércia e produto de inércia. Os valores resultantes do centro de gravidade e produto de inércia devem coincidir com o eixo de rotação do foguete, neste caso o eixo *x*. A existência de desbalanceamento residual acima do tolerado geram forças resultantes. Estas forças tendem a gerar um novo eixo de rotação desalinhado com o eixo do foguete, causando instabilidade de voo, podendo iniciar um movimento de precessão e comprometer o sucesso da missão. O momento de inércia deve ser conhecido e minimizado, pois é ele que determina a quantidade de força necessária para colocar e retirar o foguete do movimento de rolamento.

3 PARÂMETROS DE BALANCEAMENTO DA CARGA ÚTIL

Um sistema, no contexto deste trabalho, é composto por vários elementos. Analisar este sistema pode ser complicado e oneroso. Uma das grandes contribuições de Euler à mecânica foi descobrir que é possível analisar o movimento de um sistema, sem a necessidade de analisar o comportamento individual de cada elemento. A teoria do movimento de rotação de um sistema rígido é descrita pelas equações de Euler. Estas equações analisam os movimentos sempre em relação a um referencial inercial solidário ao sólido, esta técnica simplifica esta análise. O axioma, conforme Euler descreve em seus trabalhos e apresentado por Maia (1979):

> "Eis aqui, então, uma verdade muito importante: em cada corpo existem três eixos principais, que se cruzam ortogonalmente no centro de inércia do corpo. Esses eixos têm duas propriedades notáveis: uma delas é que o corpo pode girar livremente em torno de cada um deles; a outra é que o momento de inércia do corpo, relativo a um desses três eixos, é máximo, e o relativo a um dos outros dois eixos é mínimo, dentre os momentos de inércia do corpo, relativos a todos os eixos que passem pelo seu centro de inércia." (EULER, 1758).

Partindo deste conceito, seguem as fórmulas que permitem a análise estática e dinâmica da carga útil. A carga útil possui vários elementos, denominados experimentos. Desta forma pode ser determinado o balanceamento do sistema. Nas equações apresentadas nos próximos itens deste capítulo, por se tratar da contribuição de vários elementos para o sistema, será realizada a somatória das contribuições de cada massa.

3.1. Centro de gravidade

O Centro de Gravidade (CG), também conhecido como centro de massa de um objeto, como apresentado na Figura 3.1, pode ser calculado através das equações (GRAY; COSTANZO; PLHESHA, 2010):

$$x_{CG} = \frac{\sum_{i=1}^{n} m_i x_i}{\sum_{i=1}^{n} m_i}, \quad (3.1)$$

$$y_{CG} = \frac{\sum_{i=1}^{n} m_i y_i}{\sum_{i=1}^{n} m_i}, \quad (3.2)$$

$$z_{CG} = \frac{\sum_{i=1}^{n} m_i z_i}{\sum_{i=1}^{n} m_i}. \quad (3.3)$$

onde,

x_{CG},y_{CG} e z_{CG} são as distâncias das coordenadas do CG do sistema de elementos até o sistema de coordenadas de referência *xyz*;

x_i, y_i e z_i são as distâncias das coordenadas do centro geométrico de cada elemento até o sistema de coordenadas de referência *xyz*;

m_i é a massa decada um dos *n* elementos que compõem o sistema.

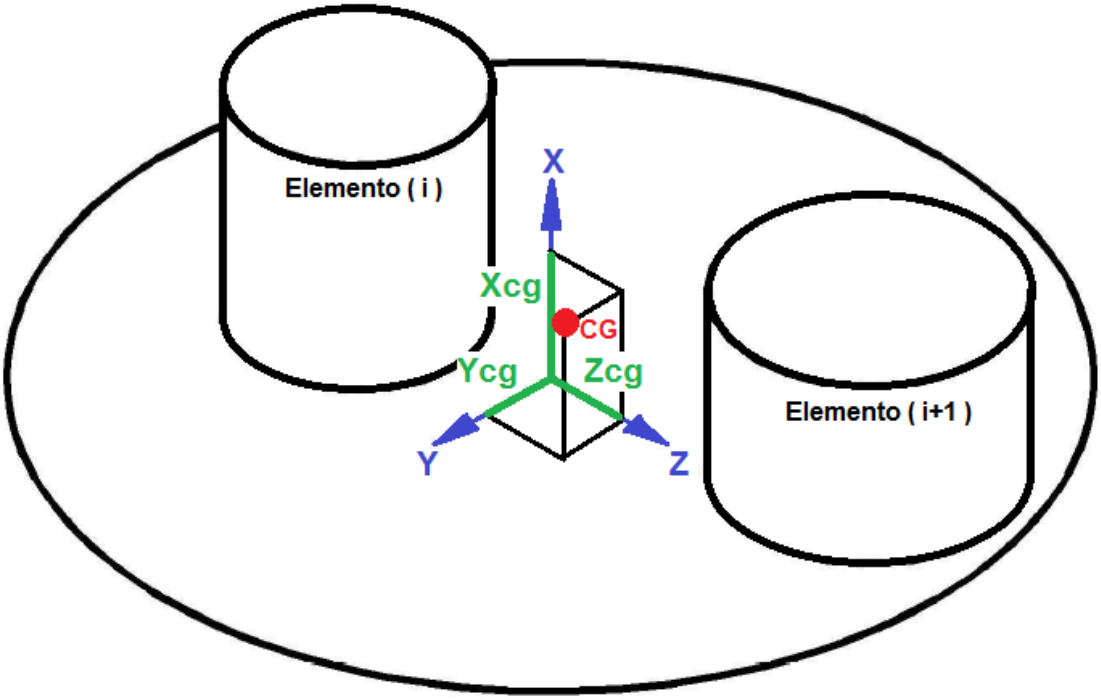


Figura 3.1 - Exemplo de um sistema para cálculo do CG.

A alocação de experimentos em cada plataforma da carga útil gera um vetor de CG resultante perpendicular ao plano *zy*. O primeiro objetivo desta otimização

é aproximar a posição deste vetor resultante à origem das coordenadas de referencia do foguete, isto é, coincidente com o eixo x do sistema cartesiano.

Um segundo cálculo necessário é determinar o centro de gravidade no eixo *x*. Ele será utilizado para calcular o valor da margem estática (*E*). No momento de lançamento, este valor deve situar-se no intervalo de 1,3 a 1,5. Isto proporciona uma situação estável de voo do foguete.

3.2. Momento de inércia

O momento de inércia é o produto da massa do elemento pela distância quadrática em relação ao eixo considerado. Este valor determina a quantidade de força necessária para colocar em movimento de rotação um determinado sistema, ou remover o rolamento. Quanto menor o valor do momento de inércia, menor será o valor da energia necessária para induzir o rolamento no foguete.

O primeiro passo é determinar o momento de inércia do elemento analisado, nesta otimização os elementos são objetos cilíndricos, como apresentado na Figura 3.2.

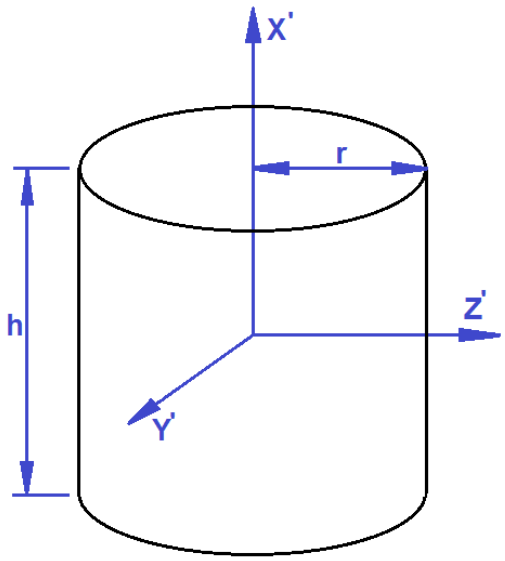


Figura 3.2 - Sólido cilíndrico e dimensões para cálculo do momento de inércia.

Os valores dos momentos de inércia $I_{x'}$,$I_{y'}$ e $I_{z'}$ do cilindro estão em relação ao sistema de coordenadas localizado no centro geométrico do sólido, identificados por x', y' e z'. Então, seus momentos de inércia em relação a este sistema de coordenadas são calculados pelas equações:

$$I_{y'} = I_{z'} = \frac{1}{12} m\,(3\,r^2 + h^2)\ , \tag{3.4}$$

$$I_{x'} = \frac{m\,r^2}{2}. \tag{3.5}$$

A próxima etapa é determinar a contribuição destes momentos em relação ao sistema de coordenadas principal da carga útil, identificado pelos eixos x, y e z, como representado na Figura 3.3. Para isto, utiliza-se o Teorema dos Eixos Paralelos, que permite calcular o momento de inércia de um elemento relativo a um eixo de rotação, quando já são conhecidos os momentos de inércia relativos a um eixo paralelo anterior e a distância entre os eixos avaliados.

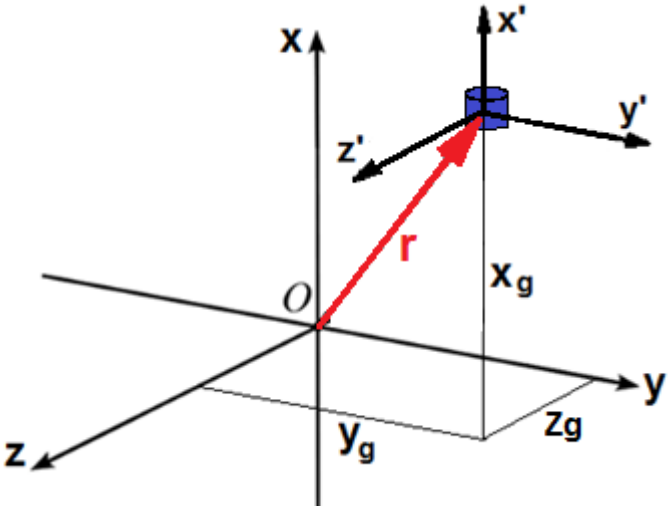


Figura 3.3 - Elemento com origem deslocada do sistema de coordenadas principal.

Desta forma, conhecendo os valores das distâncias x_g, y_g e z_g, apresentadas na Figura 3.3, os momentos de inércia do elemento em relação ao sistema de coordenadas da carga útil são calculados por:

$$I_x = I_{x'} + m\left({y_g}^2 + {z_g}^2\right)\ , \tag{3.6}$$

$$I_y = I_{y'} + m\left({x_g}^2 + {z_g}^2\right)\ , \tag{3.7}$$

$$I_z = I_{z\prime} + m\left(x_g{}^2 + y_g{}^2\right) \ , \tag{3.8}$$

onde,

$I_{x\prime}$, $I_{y\prime}$ e $I_{z\prime}$ são os momentos de inércia do elemento em relação a seu sistema de coordenadas,

I_x, I_y e I_z são os momentos de inércia do elemento em relação ao sistema de coordenadas da carga útil,

x_g, y_g e $z_g$são as distâncias do sistema de coordenadas de referência da massa em relação ao sistema de coordenadas principal x, y e z.

m é a massa do elemento avaliado.

Após calcular os valores de I_x, I_y e I_z para cada elemento do sistema, deve-se calcular o momento de inércia total em relação a cada eixo do sistema da carga útil. Este cálculo é realizado através da somatória dos momentos de cada objeto, conforme as equações:

$$I_{xx} = \sum_{i=1}^{n} (I_x)_i \ , \tag{3.9}$$

$$I_{yy} = \sum_{i=1}^{n} \left(I_y\right)_i \ , \tag{3.10}$$

$$I_{zz} = \sum_{i=1}^{n} (I_z)_i . \tag{3.11}$$

onde,

I_{xx}, I_{yy} e I_{zz} são os momentos de inércia do sistema de elementos em relação aos eixos x, y e z, respectivamente.

3.3. Produto de inércia

O produto de inércia é uma medida de balanceamento dinâmico, ou também descrito como a medida de assimetria da distribuição de massa de um corpo em relação aos planos xy, xz e yz (GRAY; COSTANZO; PLHESHA, 2010). É determinado através do cálculo do produto da massa de um objeto por suas coordenadas em relação aos dois eixos considerados (BOYTON; WIENER, 1998). É este parâmetro que pode influenciar o comportamento do foguete durante o voo na fase de rolamento. Um desbalanceamento maior que o

tolerado provoca um movimento de precessão do foguete, comprometendo sua trajetória e estrutura. Nesta análise deve-se definir um eixo de referência, conforme Boyton e Wiener (1998) é aconselhável selecionar o eixo de rotação do objeto, no caso do foguete o eixo x. É necessário garantir este balanceamento nos planos xy e xz.

Assim como apresentado no momento de inércia, o cálculo do produto de inércia se divide em três partes. Primeiramente são calculados os produtos de inércia em relação ao sistema de coordenadas posicionados no centro geométrico do elemento. Contudo, se ao menos dois eixos do elemento são de simetria, como apresentado anteriormente na Figura 3.2 do cilindro, os produtos de inércia serão nulos. Então temos:

$$I_{x'y'} = I_{y'z'} = I_{x'z'} = 0. \tag{3.12}$$

A próxima etapa é calcular os produtos de inércia de cada elemento em relação ao sistema de coordenadas da carga útil. Referenciando a Figura 3.3, temos as seguintes equações:

$$I_{xy} = I_{x'y'} + mx_g y_g, \tag{3.13}$$

$$I_{yz} = I_{y'z'} + my_g z_g, \tag{3.14}$$

$$I_{xz} = I_{x'z'} + mx_g z_g. \tag{3.15}$$

onde,

$I_{x'y'}$, $I_{y'z'}$ e $I_{x'z'}$ são os produtos de inércia do elemento em relação a seu sistema de coordenadas, e nulos neste caso em particular,

I_{xy}, I_{yz} e I_{xz} são os produtos de inércia do elemento em relação ao sistema de coordenadas da carga útil,

x_g, y_g e z_g são as distâncias do sistema de coordenadas de referência da massa em relação ao sistema de coordenadas principal x, y e z.

m é a massa do elemento avaliado.

Finalmente, os valores totais dos produtos de inércia em cada plano são realizados através das equações:

$$I_{xy} = \sum_{i=1}^{n} \left(I_{xy}\right)_i \,, \tag{3.16}$$

$$I_{xz} = \sum_{i=1}^{n} (I_{xz})_i \,, \tag{3.17}$$

$$I_{yz} = \sum_{i=1}^{n} \left(I_{yz}\right)_i \,. \tag{3.18}$$

onde,

I_{xy}, I_{xz} e I_{yz} são os produtos de inércia do sistema de elementos em relação aos planos xy, xz e yz, respectivamente, de acordo com o sistema de coordenadas da carga útil.

Os momentos relacionados podem ser agrupados matematicamente, sendo denominado como matriz de inércia ou como tensor de inércia. O tensor de inércia I é uma generalização, e apresenta a rotação em relação à eixos arbitrários, sendo representado pela equação:

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{bmatrix}. \tag{3.19}$$

Nos casos onde uma coordenada é perpendicular a um plano de simetria de um objeto, o produto de inércia deste plano será nulo. E se o sistema de coordenadas for definido de tal forma que I_{xy} = I_{xz} = I_{yz} = 0, os eixos deste sistema serão chamados de eixos principais de inércia (LAGES, 2006).

3.4. Ângulo de inclinação da carga útil

A resultante do produto de inércia de dois planos define um eixo de simetria da carga útil. Este eixo deve coincidir com o eixo referencial da rotação do foguete. Caso o ângulo de inclinação entre estes eixos seja significativo, o seu eixo de rotação sofrerá uma mudança de orientação, movimento denominado como precessão. A precessão é indesejada durante o voo, pois pode provocar alteração na trajetória, instabilidade ou esforços aerodinâmicos que comprometem a estrutura do foguete e o sucesso da missão. A Figura 3.4 apresenta um objeto desbalanceado. O eixo ideal está identificado por x e o

ângulo de inclinação do eixo resultante por x', provocando o movimento de precessão.

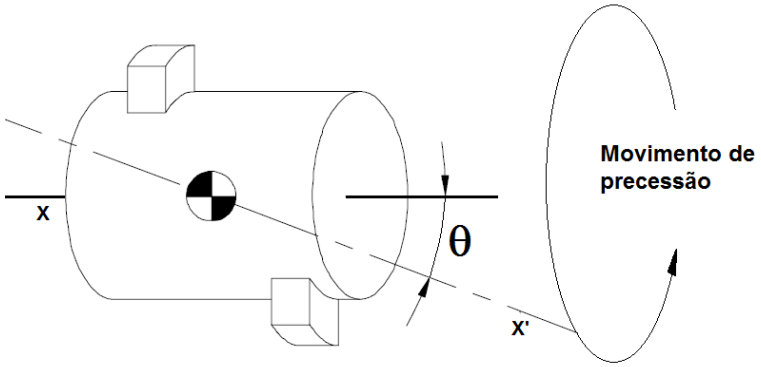


Figura 3.4 - Ângulo de inclinação do eixo do objeto.
Fonte: adaptado de Boyton; Wiener (1998).

O ângulo de inclinação pode ser calculado por:

$$\theta = 0{,}5 \arctan \frac{2\, I_{xr}}{(I_{xx} - I_o)}, \tag{3.20}$$

$$I_{xr} = \sqrt{I_{xy}^2 + I_{xz}^2}, \tag{3.21}$$

$$I_o = I_{yy} + I_{zz}. \tag{3.22}$$

onde,

I_{xr} é o produto de inércia resultante entre os planos xy e xz,

I_o é momento polar de inércia.

Desta forma, concluí-se que os parâmetros de balanceamento necessários são: o centro de gravidade, o produto de inércia e a inclinação do eixo da carga útil. O momento de inércia deve ser conhecido, pois é parte do cálculo do valor de θ. O próximo passo é definir a estratégia da otimização multiobjetivo. Para este problema, será utilizado um paradigma híbrido de inteligência computacional.

4 OTIMIZAÇÃO POR PARADIGMAS DE IC

4.1. Inteligência Computacional

Um problema é classificado intratável se o tempo necessário para resolvê-lo é considerado inaceitável para os requerimentos do usuário da solução (LINDEN, 2006). A inteligência computacional (IC) tem como objetivo permitir a resolução de problemas considerados intratáveis. Matematicamente classifica-se como problema tratável se o seu limite superior de complexidade é polinomial, e intratável se sua complexidade é não-polinomial (NP) (TOSCANI; VELOSO, 2001). Com isto, a inteligência computacional é uma técnica que busca a aproximação da solução perfeita em tempo viável, atendendo os requisitos do usuário.

A inteligência computacional é entendida como uma técnica que se utiliza de um conjunto de paradigmas, que baseadas em simulações de modelos comportamentais, e explorando as tolerâncias e incertezas do processo, buscam alcançar resultados de problemas com tratabilidade, robustez e baixo custo. Utiliza-se o termo heurística para descrever um método que baseado na experiência ou julgamento, parece conduzir a uma boa solução do problema, mas não garante produzir a solução ótima. Uma meta-heurística é uma estratégia de busca, não específica para um determinado problema, que tenta explorar eficientemente o espaço das soluções viáveis desse problema (BECCENERI; SILVA NETO, 2012).

4.2. Otimização Multiobjetivo

As soluções de problemas reais raramente envolvem apenas uma variável e uma meta, normalmente têm-se diversas variáveis e podem fazer com que existam diversas soluções. É importante mencionar que os objetivos podem ser conflitantes entre si (COELLO COELLO, 1999), e que o problema pode estar sujeito a restrições. Muitas vezes as metas são conflitantes, onde uma solução

ótima para determinado problema pode implicar em uma solução ruim para outra.

Os problemas com estas características são denominados de otimização multiobjetivo (MOOP - "multi-objective optimization problem"), por caracterizarem a necessidade de minimização simultânea de um conjunto de objetivos satisfazendo um conjunto de restrições. Neste panorama, o conceito de otimização pode ser baseado nas noções introduzidas por Francis Ysidro Edgeworth (1881) e depois generalizada por Vilfredo Pareto (1896), é chamado de Edgeworth-Pareto ótimo ou, simplesmente, Pareto-ótimo. O ótimo de Edgeworth-Pareto não nos fornece uma solução única, mas sim um conjunto de soluções não-dominadas. Este postulado tem sido base para o desenvolvimento de teoremas importantes na teoria de otimização multiobjetivos (LOBATO, 2008).

A solução de um problema multiobjetivo, seja minimizar ou maximizar um parâmetro, é formada por um conjunto de soluções que apresentam um compromisso entre os objetivos. Um conjunto de soluções é denominado conjunto Pareto-ótimo se, para cada solução do conjunto, não existe outra solução factível capaz de melhorar o valor de um dos critérios do problema, sem que simultaneamente cause um agravamento em pelo menos um dos demais critérios (AZUMA, 2011).

A formulação para definição dos requisitos e restrições de um problema de otimização multiobjetivo pode ser assim definido (DEB, 2001):

$$\begin{aligned} \text{Minimizar/Maximizar} \quad & f_m(x), && m = 1, 2, \dots, M; \\ \text{Restrita a} \quad & g_j(x) \geq 0, && j = 1, 2, \dots, J; \\ & h_k(x) = 0, && k = 1, 2, \dots, K; \\ & x_i^I \leq x_i \leq x_i^S && i = 1, 2, \dots, n\ . \end{aligned} \tag{4.1}$$

A solução x é um vetor transposto de n variáveis de decisões $x = (x_1, x_2, \dots, x_n)^T$. A última linha de restrições define os limites de variáveis,

restringindo cada variável de decisão x_i a assumir valores entre um intervalo inferior x_i^I e superior x_i^S. Este intervalo forma o espaço de variáveis de decisão, ou simplesmente espaço de decisão. Associados ao problema, a desigualdade J e a igualdade K representam o número de restrições. Os termos $g_j(x)$ e $h_k(x)$ são chamados de funções de restrição. A solução x que não satisfaz todas as restrições $(J + K)$ e, em todo o intervalo de n é denominada solução não factível. Considerando a formulação apresentada em (4.1), existem M funções objetivo que formam o vetor transposto $f(x) = (f_1(x), f_2(x), \dots, f_M(x))^T$. Cada função objetivo deve ser minimizada ou maximizada. Para cada solução x no espaço de decisão, existe um ponto no espaço dos objetivos, denominado pelo vetor$f(x) = z = (z_1, z_2, \dots, z_M)^T$.

O mapeamento entre o vetor de soluções de n dimensões e do vetor de objetivos de m dimensões é ilustrado na Figura 4.1, como proposto por Deb (2001) e Azuma (2011).

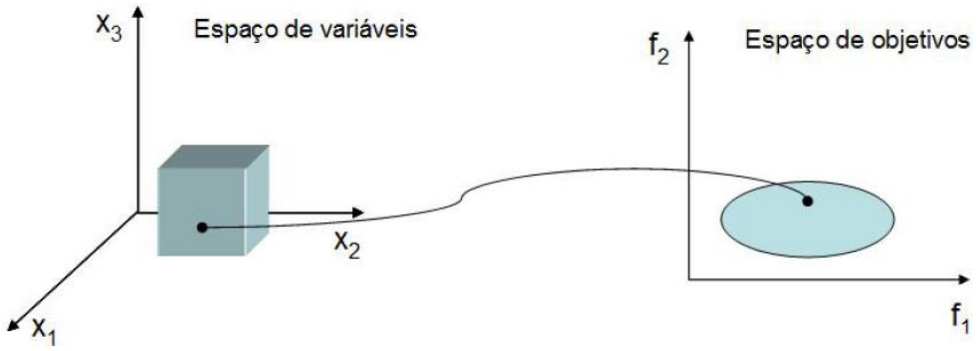


Figura 4.1 - Mapeamento de soluções no espaço de variáveis para o espaço de objetivos.

4.3. Pareto-ótimo

A representação de dominância de Pareto é utilizada para comparar duas soluções factíveis de um problema multiobjetivo. Considerando duas soluções *a* e *b*, afirma-se que *a* domina *b* (*a* $\preceq$ *b*) se as seguintes condições são satisfeitas:

a) A solução *a* é igual ou melhor que a solução *b* em todas as funções objetivo;

b) A solução *a* é superior a *b* em pelo menos uma função objetivo.

Quando não é possível definir as relações de dominância entre as soluções, estas são chamadas de conjunto de soluções não-dominadas, ou chamada de *Pareto-Ótimo*. A *fronteira de Pareto* é o conjunto das soluções que minimizam ou maximizam as funções-objetivo.

Como exemplo, analisaremos a Figura 4.2, em que está representado o espaço de objetivos com duas funções, a primeira, chamada de f_1, representada no eixo das abscissas, a segunda, chamada de f_2, no eixo das coordenadas, e cinco pontos possíveis de soluções.

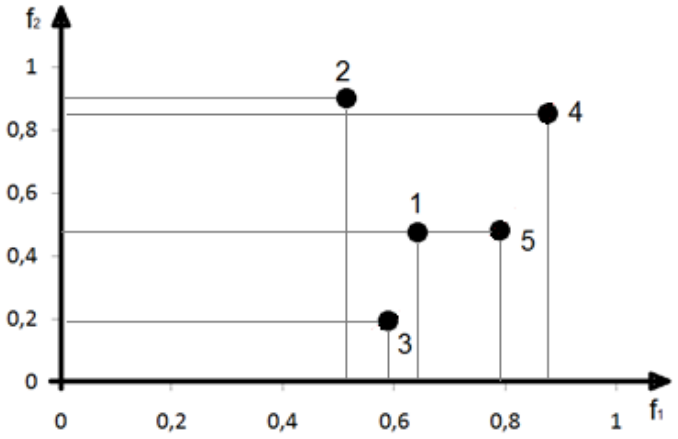


Figura 4.2 - Gráfico das funções objetivo e possíveis soluções.

Considere que a função 1 (f_1) necessita ser maximizada e a função 2 (f_2) minimizada. O próximo passo é comparar ponto a ponto, de acordo com os critérios de dominância apresentados anteriormente. Vejamos alguns exemplos:

a) Pontos 2 e 3: O ponto 2 é pior que 3 para f_1, e 2 é pior que 3 para f_2, portanto, 3 domina 2 ($3 \preceq 2$), então 3 representa uma possível solução que pertence à fronteira de Pareto, e 2 não é uma solução viável;

b) Pontos 3 e 5: O ponto 3 é pior que 5 para f_1, e 3 é melhor que 5 para f_2, portanto, não há como definir a dominância entre estes dois pontos, então ambos pertencem ao conjunto que forma a fronteira de Pareto;

Ao final da análise entre todos os pontos, concluí-se que os pontos 3, 4 e 5 dominam os pontos 1 e 2, descartando estes pontos como soluções viáveis. E como não é possível estabelecer a dominância entre os pontos 3, 4 e 5, estes pertencem ao conjunto da fronteira de Pareto, sendo representados por uma curva unindo estes pontos de soluções viáveis, como mostra a Figura 4.3.

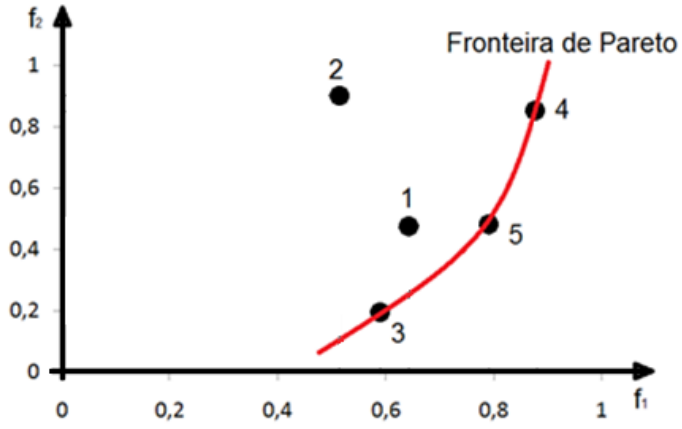


Figura 4.3 - Representação da possível fronteira de Pareto.

Caso existam mais pontos para análise, como os pontos 6, 7, 8, e 9, o comportamento desta curva pode alterar-se, como apresentado na Figura 4.4.

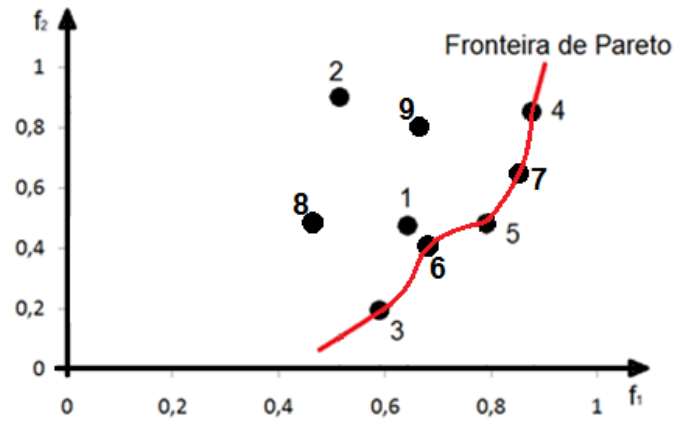


Figura 4.4 – Possível curva que representa a fronteira de Pareto.

Se informações adicionais sobre a importância relativa entre os objetivos são desconhecidas, todas as soluções Pareto-Ótimas são igualmente importantes. Assim, conforme Deb, Mohan e Mishira (2003), três importantes metas em otimização multiobjetivo se fazem necessárias:

1. Encontrar um conjunto de soluções que esteja o mais próximo possível da fronteira de Pareto;
2. Encontrar um conjunto de soluções com a maior diversidade possível;
3. Realizar as duas metas anteriores com a maior eficiência possível.

4.4. Algoritmo genético

Os Algoritmos Evolutivos (AEs) são técnicas de otimização que imitam os princípios da evolução natural para busca e otimização de problemas. Os AEs propiciam uma alternativa aos métodos matemáticos clássicos, como os métodos diretos ou método dos gradientes, e têm sido largamente explorados em problemas de otimização, buscando minimizar ou maximizar um vetor de funções-objetivo. Algumas dificuldades encontradas nos métodos clássicos são:

- a convergência para uma solução ótima depende da escolha da solução inicial;
- a maioria dos algoritmos tendem a ficar presos em soluções de ótimos locais;
- um determinado algoritmo pode ser eficiente para resolver um problema, mas ineficiente na solução de outros problemas;
- os algoritmos não são eficientes em resolver problemas que possuam um espaço de busca discreto.

Uma das características mais importantes dos AEs é que possibilitam encontrar soluções ótimas ou adequadas para um problema complexo sem usar informação adicional, como cálculo de derivadas de funções (GOLDBERG, 1989). Outro grande diferencial dos AEs tem sido na solução de problemas multiobjetivo. A utilização dos Algoritmos Evolutivos para solução de problemas

multiobjetivos é definido como MOEA (do inglês "MultiObjective Evolutionary Algorithm").

Com base na observação na evolução natural das espécies, os Algoritmos Genéticos (AG) têm sua filosofia embasada na teoria de Darwin. Portanto, é uma meta-heurística inspirada no comportamento da natureza. É um método de busca que a partir de uma população inicial sobrepondo todo o espaço de busca, passa pelos processos de combinação, provendo uma nova geração de indivíduos mais aptos a representarem a solução do problema. Este processo é repetido por um número de gerações determinadas, ou até que uma geração possa representar a solução com uma incerteza pré-definida.

De acordo com Reeves (2003), o Algoritmo Genético é uma meta-heurística. Os Algoritmos Genéticos pertencem a uma classe de algoritmos de busca probabilística inteligente (PARDALOS; RESENDE, 2003), que se baseiam no processo evolucionário de organismos biológicos na natureza, tratando de uma população de soluções e da combinação delas, de modo a gerar novas soluções.

O AG se inicia na concepção de uma população inicial, que são valores que podem representar uma solução para o problema. Estes indivíduos são avaliados através de uma função de adaptabilidade. Esta função permite verificar se a solução correspondente se aproxima da solução ideal. As funções de adaptabilidade geralmente são elaboradas a partir da definição matemática do problema em questão. Os indivíduos com maior adaptabilidade têm maiores probabilidades de serem selecionados para reprodução. Após a seleção, a próxima etapa é o cruzamento. Isto permite que as características de vários indivíduos possam ser mescladas gerando um novo indivíduo mais adaptado. Antes da finalização deste novo indivíduo, existe a mutação, que pode ocorrer com uma probabilidade baixa, mas é importante para evitar uma perda prematura de bom material genético. A mutação também permite que mesmo no processo de intensificação, seja possível realizar pequenas buscas em

outras regiões do espaço, impedindo que logo a população se concentre numa região de ótimo local.

A nova população gerada passa por uma reavaliação da sua adaptabilidade. As etapas anteriores podem se repetir diversas vezes dependendo do critério de parada definido. Pode-se utilizar como critério as seguintes premissas: um número fixo de gerações, ou sucessivas gerações até que os indivíduos representem um valor de adaptabilidade com uma incerteza baixa o suficiente para representar uma ótima solução do problema. A Figura 4.5 apresenta o sequenciamento dos eventos do algoritmo genético.

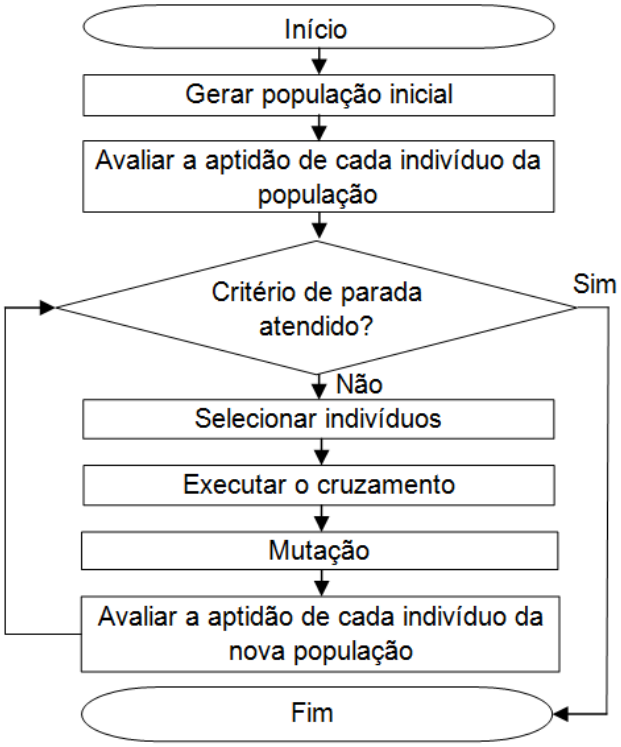


Figura 4.5 - Fluxograma dos eventos do algoritmo genético.

4.5. Lógica Fuzzy

Alguns conceitos no mundo não podem ser bem representados por limites bem definidos, ou valores exatos. Diante deste fato, Zadeh (1965) desenvolveu a teoria dos conjuntos nebulosos (*Fuzzy Sets*) para tratar informações vagas. Esta teoria trata a imprecisão ou incerteza que ocorre de maneira natural na

representação de fenômenos da natureza. Portanto, a lógica nebulosa aproxima a decisão computacional da decisão humana. Isto é realizado de forma que a decisão de uma máquina não dependa de informações binárias.

A lógica nebulosa consiste em aproximar a decisão computacional da decisão humana, tornando as máquinas mais capacitadas para o trabalho. Isso é realizado de forma que a decisão de uma máquina não dependa apenas de informações binárias do tipo "sim" ou "não", ou de valores "concretos", mas de maneira que também possa decidir com base em valores "abstratos" do tipo "um pouco mais", "talvez sim" e em outras tantas variáveis que representem os tratamentos de informações fornecidas pelo homem (PILLAT, 2012). A implementação de um sistema fuzzy é composto pelos seguintes blocos funcionais (SIMÕES; SHAW, 1999):

- interface de fuzzificação;
- base de regras;
- lógica de tomada de decisões;
- interface de defuzzificação.

A Figura 4.6, proposta por Pillat (2012), apresenta esta estrutura da lógica fuzzy.

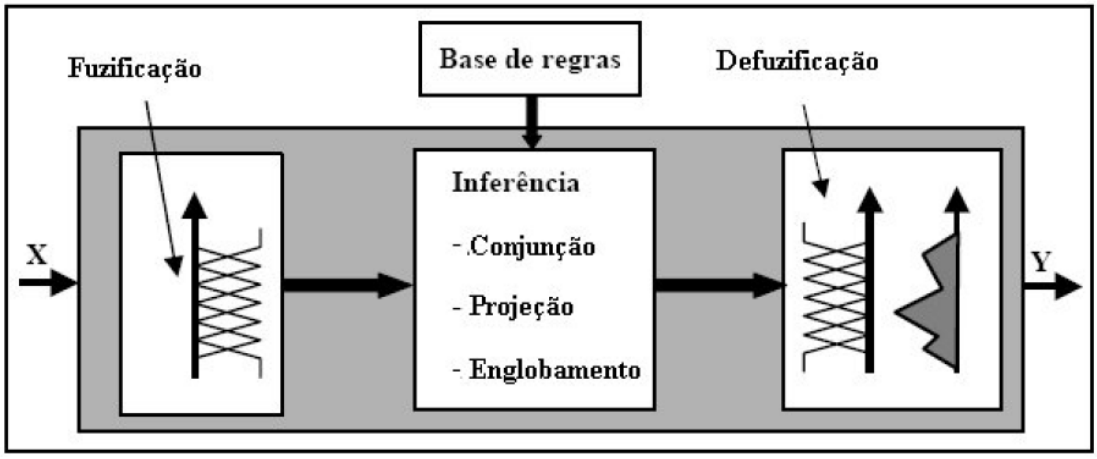


Figura 4.6 - Estrutura sistêmica de inferência nebulosa.

Fonte: Pillat (2012).

Esta estrutura é aplicada para inferir a aptidão dos indivíduos, analisando o seu grau de comprometimento com cada função objetivo que se deseja otimizar. Posteriormente, os indivíduos são colocados em ordem decrescente de aptidão, que é parte do processo do algoritmo genético.

A primeira etapa, denominada fuzzificação, consiste em receber os dados de entrada, ajustá-los para que sejam cobertos pelo universo de discurso previamente definido. Esta interface utiliza funções de pertinência contidas numa base de conhecimento, associando-as a rótulos linguísticos. As variáveis linguísticas são expressões que caracterizam qualitativamente as grandezas físicas. Estes valores numéricos são substituídos por termos como: pequeno, médio e grande; baixo, médio e alto; etc. Desta forma, o eixo das abscissas contém os valores da grandeza avaliada, o eixo das coordenadas o grau de pertinência desta grandeza ao rótulo linguístico. O grau de pertinência é um intervalo de 0 a 1, no qual se pode inferir que um valor pertence de 0% a 100% àquela variável linguística.

Como exemplo, será analisada a tomada de decisão da lógica fuzzy em classificar o grau de risco em aplicar na bolsa de valores. A variável de entrada é o índice da bolsa em pontos, a inferência fuzzy deverá qualificar o risco atribuindo as seguintes variáveis linguísticas: baixo, médio ou alto risco.

O primeiro passo é definir as funções de pertencimento da variável de entrada. Aqui foram definidas três funções triangulares, representando a quantidade de pontos da bolsa, como poucos pontos, médios pontos e altos pontos. A função de pertencimento pode ser determinada com base na estatística dos dados, ou através de redes neurais (TSOUKALAS; UHRIG, 1997).

Depois, definem-se as regras de inferência, que relacionam a entrada e saída de dados do problema. Estas regras representam a base de conhecimento da lógica fuzzy e fornecem as definições numéricas necessárias às funções de pertinência utilizadas no conjunto de regras. A lógica de tomada de decisões, incorporada na estrutura de inferência da base de regras, usa implicações

fuzzy para simular uma tomada de decisão humana. Estas regras são definidas por lógicas "se <condição> então <conclusão>". Por exemplo, no problema proposto anteriormente das alturas, podem-se escrever as seguintes regras:

Regra 1: se (Índice=Alto) então (Risco = Baixo);

Regra 2: se (Índice = Médio) então (Risco = Médio);

Regra 3: se (Índice = Pequeno) então (Risco = Alto).

Supondo o índice da bolsa em 22 pontos, deseja-se saber o grau de risco de aplicar na bolsa neste momento, como apresentado na Figura 4.7.

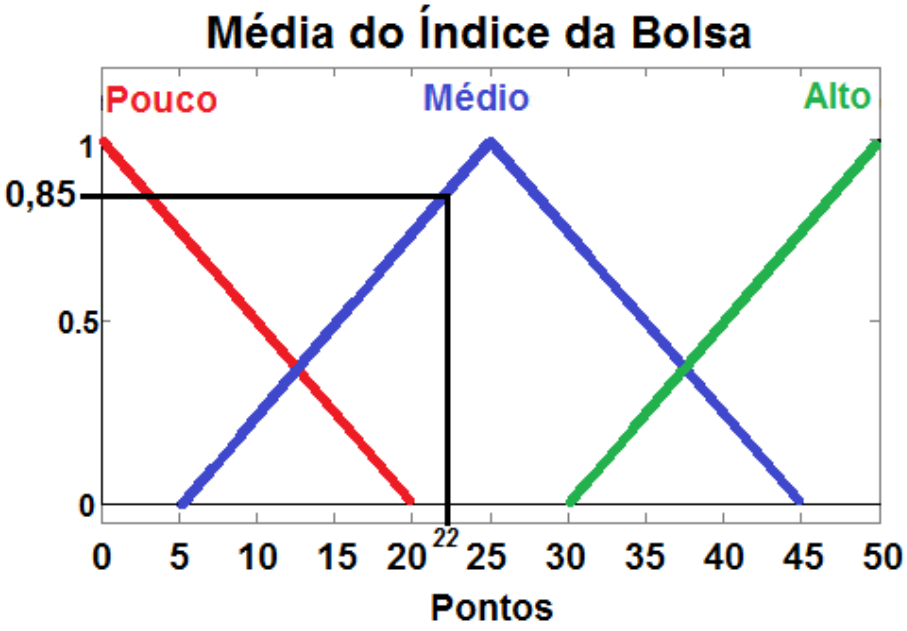


Figura 4.7 - Exemplo de fuzzificação.

Nesta situação, o valor de entrada é representado no eixo horizontal, uma reta vertical indica que foi ativada a regra 2 (índice = médio), em seguida calcula-se sua projeção no eixo das coordenadas. Este valor encontrado é denominado Grau de Pertencimento (DOF - "degree of fulfillment"), dado pela equação:

$$DOF_2 = \frac{22-5}{25-5} = 0{,}85. \qquad (4.2)$$

Após analisar o grau de pertencimento da regra ativada, um cálculo matemático define o grau de pertencimento da variável de saída para cada valor linguístico. O resultado da projeção na entrada ($DOF_2 = 0{,}85$) é transmitido para o gráfico de saída, dando início a etapa da defuzzificação. O

objetivo é obter um único valor numérico discreto, que possa representar os valores fuzzy inferidos da variável linguística de saída (SIMÕES; SHAW, 1999). As variáveis de saída estão apresentadas na Figura 4.8.

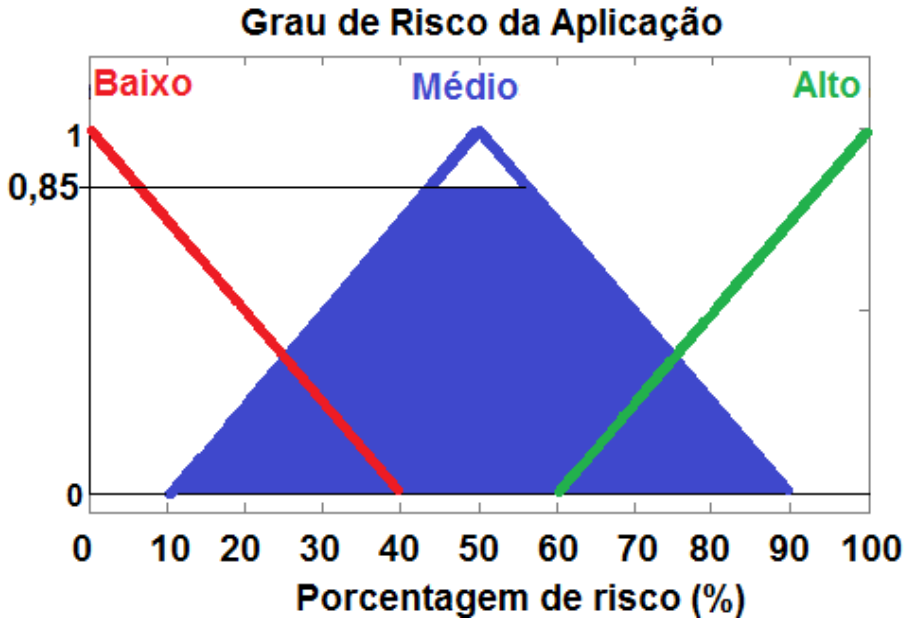


Figura 4.8 - Variável de saída da defuzzificação.

Os principais métodos de defuzzificação são os cálculos através do: centro da área, centro da soma e média do máximo. O método do centro da área consiste no cálculo do centro de gravidade da área que representa a variável linguística de saída. O método do centro da soma, o valor de saída é determinado através do cálculo pela média ponderada dos máximos valores de saída, sendo que os pesos são os resultados das inferências determinadas nas variáveis de entrada. Finalmente, o método de média dos máximos é calculado a média da contribuição do valor máximo de cada pertencimento da saída.

Para o exemplo tratado, através da regra 2, conhecemos que o risco é Médio, porém deve-se determinar o pertencimento à variável linguística "Médio Risco". Foi escolhido o método do Centro da Soma, e calculado pela equação:

$$u_{COS} = \frac{(DOF_{entrada} \,.\, CentroDOF_{Saída})}{(DOF_{entrada})} = \frac{(0{,}85 \,.\, 50)}{(0{,}85)} = 50\ \% \qquad (4.3)$$

Desta forma, conclui-se que no caso do índice da bolsa igual a 22 pontos, o grau de risco de investir na bolsa é de 50%.

Uma segunda situação, o índice é de 40 pontos, como apresentado na Figura 4.9.

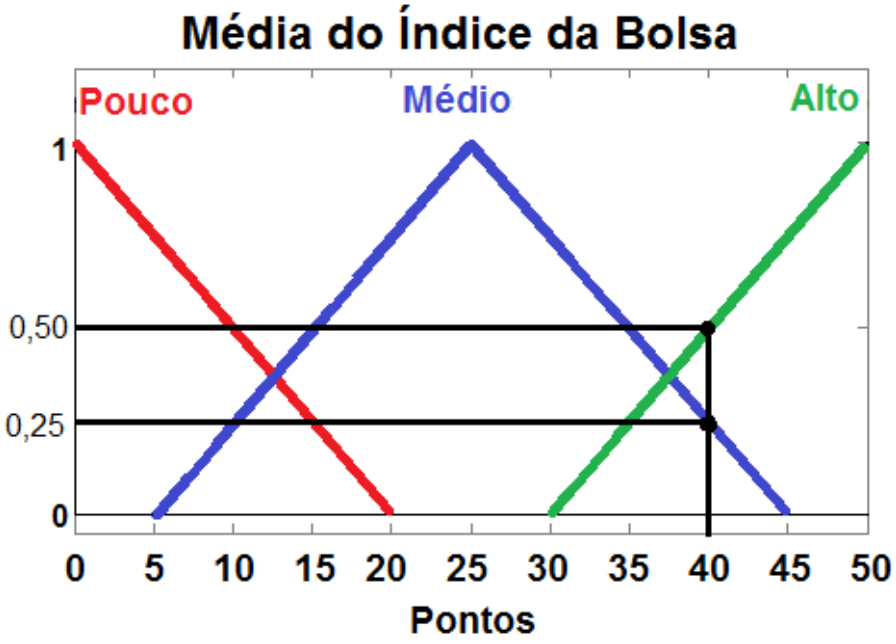


Figura 4.9 - Entrada da fuzzificação e funções de pertencimento ativadas.

Agora são ativadas as regras 2 (Pontos Médios) e 1 (Pontos Altos), então, calcula-se o grau de pertencimento para as duas regras da seguinte forma:

$$DOF_1 = \frac{40-30}{50-30} = 0{,}50\ , \tag{4.4}$$

$$DOF_2 = \frac{45-40}{45-25} = 0{,}25. \tag{4.5}$$

A Figura 4.10 mostra a implicação das duas regras ativadas na variável de saída da lógica fuzzy, isto é, risco baixo e médio.

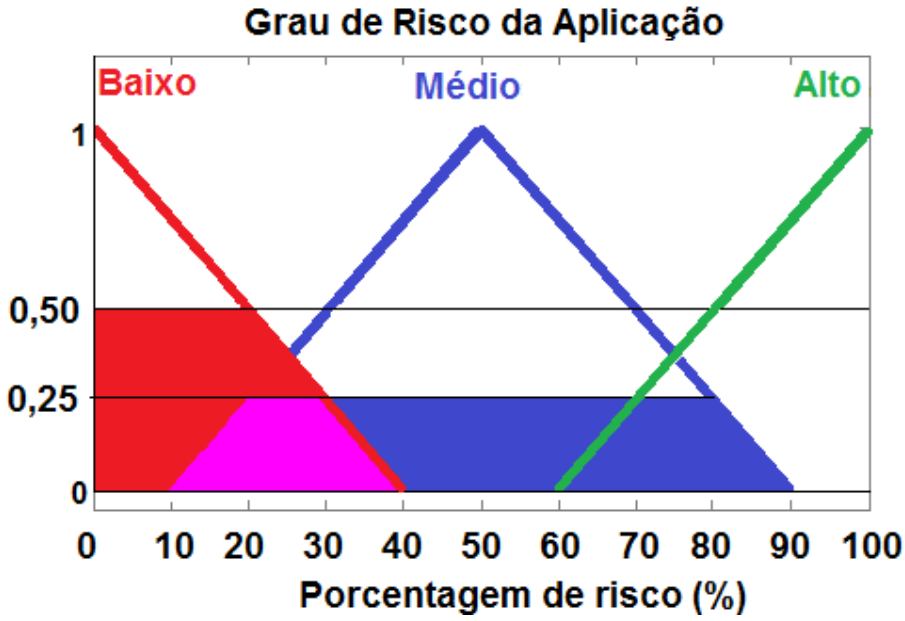


Figura 4.10 - Entrada da fuzzificação e funções de pertencimento ativadas.

Agora, calcula-se o grau de risco, com a contribuição das duas regras, temos:

$$u_{cos} = \frac{(0{,}25\,.\,50) + (0{,}50\,.\,0)}{(0{,}25 + 0{,}50)} = 16{,}7\ \% \tag{4.6}$$

Portanto, para o índice de 40 pontos, o risco é Baixo com grau de 16,7%.

A pesquisa bibliográfica deste Capítulo teve como propósito dar o embasamento necessário para o desenvolvimento de um programa. O objetivo deste programa é alocar os experimentos na carga útil do foguete otimizando os parâmetros de balanceamento estático e dinâmico, utilizando os conceitos de inteligência computacional abordados.

5 DESENVOLVIMENTO E APLICAÇÃO

5.1. Linguagem de programação

O código do programa desenvolvido foi todo elaborado em linguagem C. Esta linguagem proporciona um processamento do código mais rápido. Outra motivação de sua escolha é que não existe a necessidade de uma interface com usuário de alto nível. Para o programa os dados de entrada dos experimentos podem ser atribuídos através de um arquivo, ou direto dentro do código. Após o processamento, é fornecido um arquivo com os resultados das coordenadas cartesianas de cada objeto, que pode ser aberto em um programa de planilha eletrônica, editor de texto ou ferramenta CAD, como exemplos: AutoCAD®, SolidWorks®, CREO®, etc. Neste programa, foi implementado a saída no padrão do AutoCAD®, por esta ser a ferramenta mais utilizada em avaliações e geração de desenhos para a fabricação das plataformas.

5.2. Inteligência computacional híbrida

Na otimização da alocação dos experimentos na carga útil, foram combinados dois paradigmas de inteligência computacional. O Algoritmo Genético realiza o processo de busca por soluções ótimas e a lógica fuzzy determina a aptidão de cada indivíduo da população. Esta agregação de funcionalidades busca elevar o potencial de otimização quando comparado a utilização de somente um paradigma. A Figura 5.1 apresenta o fluxograma do algoritmo desenvolvido, com as responsabilidades atribuídas ao algoritmo genético e à lógica fuzzy.

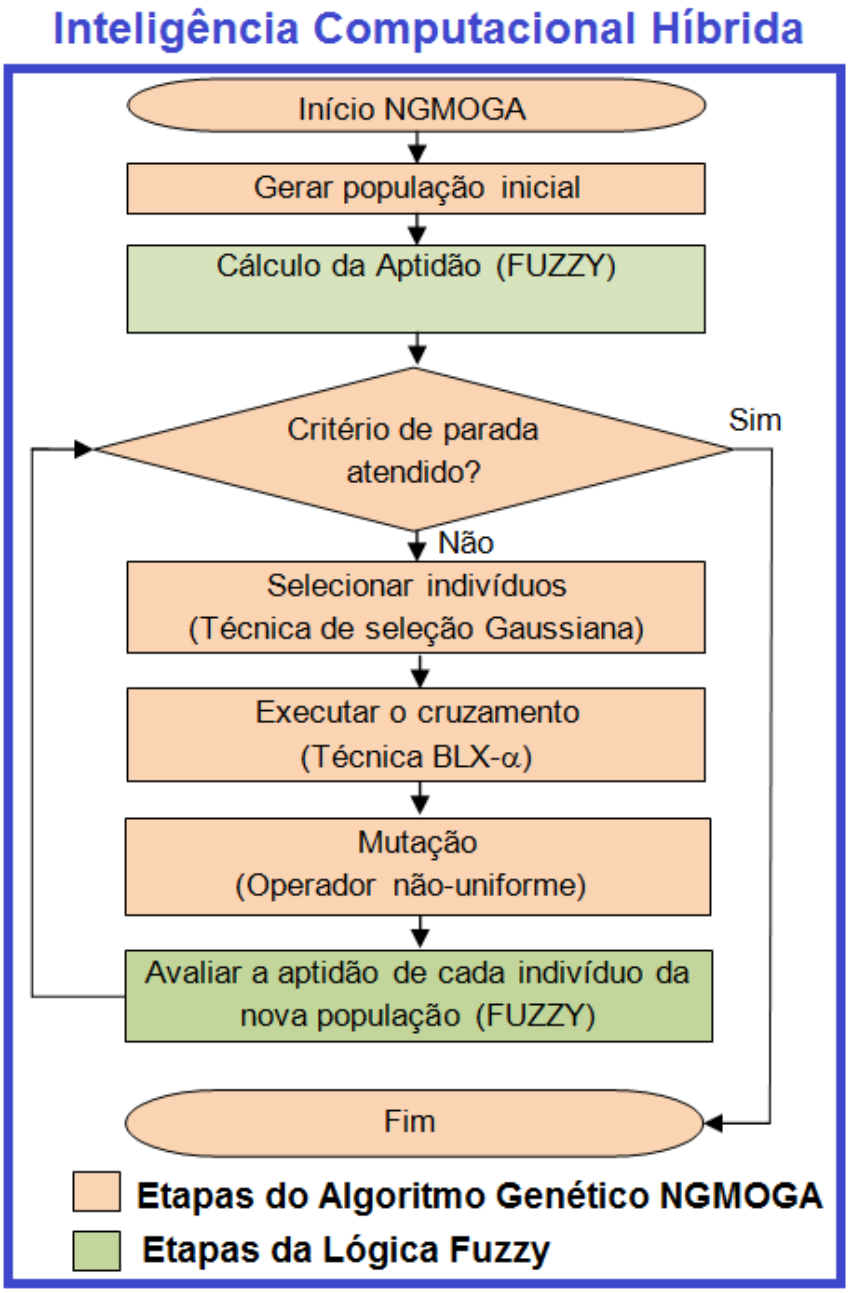


Figura 5.1 - Fluxograma da Inteligência Computacional Híbrida desenvolvida.

A seguir será apresentado, com detalhes, cada etapa da aplicação desenvolvida utilizando o conceito de hibridização de paradigmas de inteligência computacional.

5.3. Algoritmo genético

O Algoritmo Genético (AG) é uma meta-heurística que apresenta bom desempenho na solução de problemas com complexidade não-polinomial, de engenharia e multiobjetivo. O AG pode pesquisar diferentes regiões do espaço de busca, encontrando soluções para problemas difíceis, não-convexo, descontínuo e espaço de solução multimodal (KONAK ET AL, 2006). Outra vantagem, o AG não exige do usuário priorizar escala ou pesar objetivos, este é capaz de otimizar um problema multiobjetivo, sendo capaz de aproximar-se da verdadeira frente de Pareto (JONES; MIRRAZAVI; TAMIZ, 2001). Em Gen e Cheng (2000) são apresentadas as diversas áreas e trabalhos em que o AG apresentou sucesso na otimização, entre estas áreas cita-se: biologia, otimização combinatória, ciência da computação, engenharia do controle, engenharia de desenvolvimento, otimização. Também apresenta diversos trabalhos na hibridização dos AGs com outras meta-heurísticas como: AG com fuzzy e AG com redes neurais, para aplicações de processamento de imagens e reconhecimento de padrões.

Existem diversos trabalhos de pesquisa publicados que apresentam bons resultados de otimização multiobjetivo utilizando AG. Ao longo deste desenvolvimento e melhoramento do AG, surgiram diversas abordagens quanto sua atribuição de aptidão, elitismo e diversificação. A Tabela 5.1 apresenta um resumo das abordagens de AG e suas principais características.

Tabela 5.1 - Principais abordagens do AG na literatura.

Algoritmo	Vantagens	Desvantagens
VEGA	Primeira implementação direta e simples de MOGA	Tende a convergir para o extremo de cada objetivo
MOGA	Extensão do AG mono-objetivo para multiobjetivo.	Convergência lenta. Problemas associados aos

	Utiliza ordenação por Pareto.	tamanhos de parâmetros.
NSGA-II	Parâmetro único, bem testado e eficiente.	Distancia de aglomeração funciona somente no espaço de objetivos.
SPEA-2	Preserva os pontos extremos.	Cálculo da aptidão e da densidade são computacionalmente custosos.

Fonte: adaptada de Jones; Mirrazavi; Tamiz (2001).

Nesta dissertação foi implementada uma versão modificada do MOGA, conhecida como "Non-Generational Multi-Objective Genetic Algorithm" (NGMOGA). Este modelo apresenta vantagem em relação ao algoritmo genético tradicional, pois preserva as soluções boas que estão próximas à fronteira de Pareto. Esta preservação explica o termo não-geracional, já que a nova população é formada por indivíduos da nova e da antiga geração (BARBOSA, 2013). Embora outras técnicas tenham obtido sucesso em otimização, a técnica NGMOGA atende os requisitos de aproximar as soluções da fronteira de Pareto com baixo custo computacional e simplicidade de modelamento do algoritmo, conforme mostra Fonseca e Fleming (1993) no trabalho sobre MOGA.

5.3.1. Entrada de dados do algoritmo

A primeira etapa do desenvolvimento consiste em definir quais os dados de interesse de entrada para o programa. O programa de alocação de experimentos na carga útil será denominado como AEP (do inglês "Allocator Experiments in the Payload"). O desenvolvimento dividiu-se em duas etapas. A primeira foi elaborar uma versão inicial do programa, com o propósito de validar a lógica implementada com um número reduzido de parâmetros. A versão final utiliza todos os parâmetros necessários para otimizar o problema de alocação

de experimentos na carga útil. A Tabela 5.2 apresenta as diferenças entre a versão inicial e final.

Tabela 5.2 - Diferenças entre as versões do AEP.

Parâmetro	AEP inicial	AEP final
Objetos	06	20
Parâmetros dos experimentos	Raio, altura e massa	Raio, altura e massa
Quantidade de plataformas	01 (inferior)	10 (05 inferiores e 05 superiores)
Restrições de objetivos	Raio máximo plataforma Sobreposição de objetos	Raio máximo plataforma Sobreposição de objetos
Objetivos	Minimizar CG, PI e θ de um módulo	Minimizar CG, PI e θ da carga útil

5.3.2. Representação dos cromossomos

A modelagem dos dados segue o conceito da genética populacional, onde cada indivíduo possui características que os diferem dos outros. Estas características estão definidas em seu cromossomo, que consiste numa cadeia de DNA (Ácido Desoxirribonucleico) que contém suas informações genéticas. Um indivíduo possui um número de cromossomos, e cada cromossomo possui uma sequencia de genes, como apresentado na Figura 5.2.

Indivíduo							
Cromossomo A				Cromossomo B			
Gene 1	Gene 2	Gene 3	Gene 4	Gene 1	Gene 2	Gene 3	Gene 4

Figura 5.2 - Modelo genético generalizado de um indivíduo.

Utilizando o mesmo conceito apresentado, o espaço de soluções possíveis para otimização forma a população de indivíduos. Cada possível solução do problema é um indivíduo desta população. Cada indivíduo possui uma cadeia de DNA, na qual DNA possui vários cromossomos e que representam os experimentos. O cromossomo possui genes, que representam parâmetros para alocação do experimento. A Figura 5.3 apresenta esta hierarquia.

Indivíduo 1								
Experimento 1			Experimento 2			Experimento 3		
X	Y	Z	X	Y	Z	X	Y	Z

Figura 5.3 - Modelo genético de um indivíduo da população.

Onde x, y e z são os valores das coordenadas cartesianas de posicionamento dos experimentos na carga útil. A forma de representação utilizada é a baseada em números Reais. Estes são os dados manipulados pelo algoritmo genético, isto é, variam com as iterações do AG.

As informações de raio, altura, e massa formam os três atributos de cada objeto. Estes dados são fixos ao longo do algoritmo, pois são as informações dos experimentos a serem alocados na carga útil. Para armazenar estes dados, utiliza-se um vetor, de tamanho igual à quantidade de experimentos pela quantidade de características, representado pela Figura 5.4.

Experimento 1			Experimento 2			Experimento 3		
Raio	Altura	Massa	Raio	Altura	Massa	Raio	Altura	Massa

Figura 5.4 - Modelo de dados das características dos experimentos.

5.3.3. Cálculo da aptidão dos indivíduos com Fuzzy

O objetivo é garantir o balanceamento estático e dinâmico da carga útil, para isto devem-se minimizar cinco parâmetros: os centros de gravidade em y e z, os produtos de inércia nos planos xy e xz, e o ângulo θ do eixo de inclinação do foguete. Contudo podemos reduzir estes cinco objetivos, para três, agrupando os parâmetros de centro de gravidade y e z em um único, denominado centro de gravidade resultante. No produto de inércia, agrupa-se

os resultados dos planos xy e xz em um único parâmetro denominado produto de inércia resultante.

O cálculo da aptidão dos indivíduos, ou também chamada de função de avaliação, é realizada através da aplicação das equações de balanceamento citadas na formulação proposta por Deb (2001). Aplicando a redução de objetivos proposta anteriormente, tem-se a otimização descrita pelas equações:

$$\text{Minimizar} \quad f_{CGr}(y,z) = \sqrt{\left(\frac{\sum_{i=1}^{n} m_i y_i}{\sum_{i=1}^{n} m_i}\right)^2 + \left(\frac{\sum_{i=1}^{n} m_i z_i}{\sum_{i=1}^{n} m_i}\right)^2}\ , \qquad m = 1, 2, \dots, n; \quad (5.1)$$

$$f_{Ixr}(x) = \sqrt{{I_{xy}}^2 + {I_{xz}}^2}\ , \quad (5.2)$$

$$\theta = 0{,}5 \arctan\frac{2\, I_{xr}}{(I_{xx} - I_o)}\ , \quad (5.3)$$

$$\text{Restrito a} \quad r_{imáx} = r_i + \sqrt[2]{y_i^2 + z_i^2} \leq R_p\ , \qquad i = 1, 2, \dots, n; \quad (5.4)$$

$$r_i + r_j < \sqrt{(y_i - y_j)^2 + (z_i - z_j)^2}\ . \qquad \begin{matrix} j = 1, 2, \dots, n; \\ \text{para } i \neq j \\ \text{e } x_i - x_j > 0 \end{matrix} \quad (5.5)$$

onde,

$f_{CGr}(y, z)$ função objetivo do centro de gravidade resultante,

m_i massa do experimento,

f_{Ixr} função objetivo do produto de inércia resultante,

θ função objetivo da inclinação do eixo do foguete,

$r_{imáx}$ é o raio máximo de alocação do experimento,

r_i e r_j raios de posicionamento dos experimentos avaliados,

R_p raio da plataforma,

y_i, z_i, y_j e z_j são as distâncias das coordenadas do centro do experimento,

x_i *e* x_j são as alturas dos objetos nos módulos,

Os valores das coordenadas estão em relação ao plano definido pelo eixo central *x* (eixo de rolamento) do veículo, ver Figura 2.5 apresentada anteriormente.

Na literatura, em otimização multiobjetivo, a aptidão é calculada como a média aritmética, ou a média ponderada de cada objetivo. Contudo, selecionar os pesos adequados costuma ser uma tarefa complicada, devido à dificuldade em estabelecer o grau de importância de cada requisito. A lógica fuzzy tem como principal objetivo tratar estas decisões em que não há um valor ideal para um parâmetro. Sua lógica permite a simulação do entendimento e avaliação de resultados feita por um especialista, e tomar a decisão mais adequada à situação, neste caso, atribuir uma aptidão ao indivíduo.

A primeira etapa consiste em normalizar os resultados das funções objetivo, atribuindo valores entre 0 e 1. Isto é feito para cada função objetivo, calculando a divisão entre o valor da função objetivo do indivíduo pelo valor máximo deste objetivo existente na população, conforme a equação:

$$u_{fi} = \frac{f_{obj}(i)}{MAX\left(f_{obj}(i)\right)}, \tag{5.6}$$

onde,

u_{fi} é o valor normalizado da função do indivíduo,

$f_{obj}(i)$ é o valor do parâmetro do indivíduo,

$MAX\left(f_{obj}(i)\right)$ é o maior valor do parâmetro analisado em toda população.

Depois, deve-se elaborar o universo de discurso, que nesta otimização será dividido em três entradas. Em cada entrada definem-se as funções de pertencimento, nomeando-as convenientemente conforme a análise do valor da variável. A Figura 5.5 mostra o universo de discurso do valor do centro de gravidade resultante, suas três funções de pertinência com seus rótulos linguísticos, como ótimo, regular e péssimo.

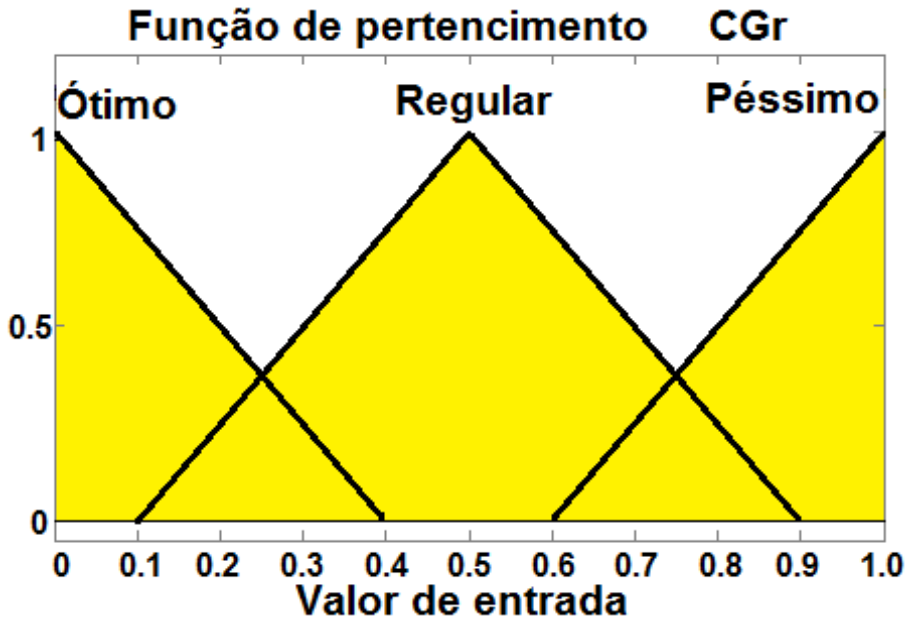


Figura 5.5 - Universo de discurso do centro de gravidade.

Esta normalização também é feita para as funções objetivo do produto de inércia, ângulo de inclinação do foguete e na saída para o valor da aptidão. Todas estas variáveis normalizadas entram nas regras de inferência fuzzy, que toma a decisão e atribui um valor de aptidão ao indivíduo, como apresentado no diagrama da Figura 5.6.

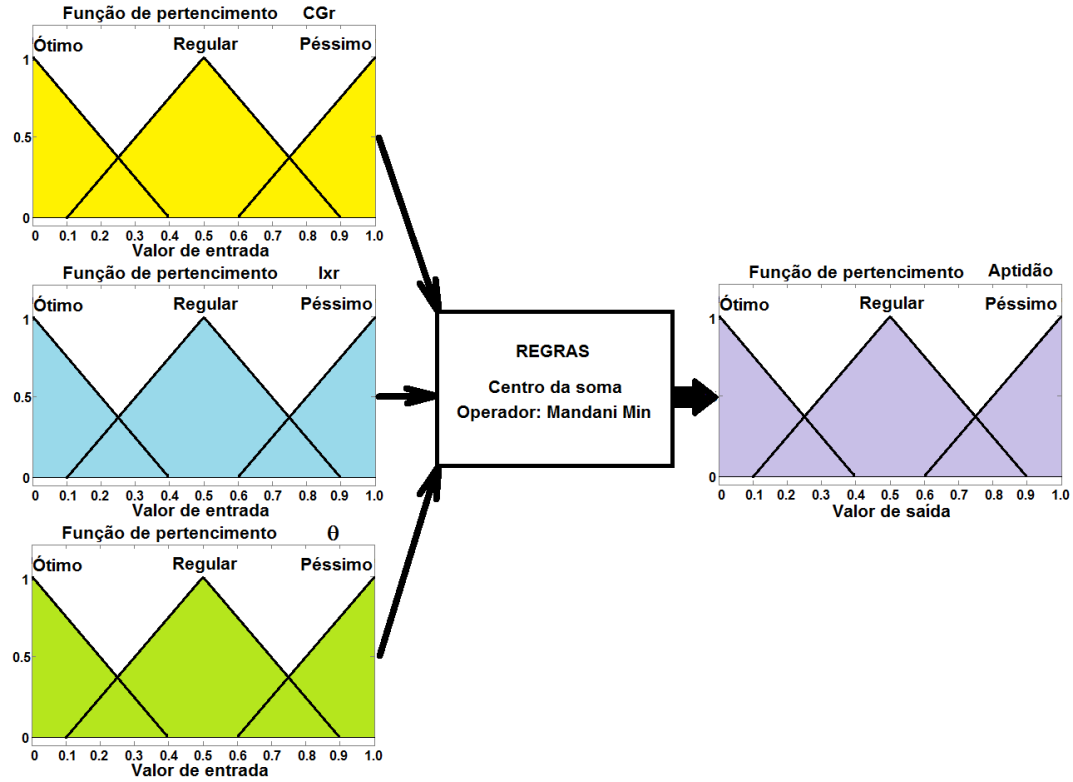


Figura 5.6 - Diagrama completo da lógica fuzzy.

A próxima etapa consiste em definir as regras para interpretação das variáveis de entrada, realizando a defuzzificação e retornando o valor da aptidão do indivíduo. As regras de inferência fuzzy consistem em definir uma sequencia de implicações se <condição> então <conclusão>, como as aplicadas em lógica de programação. No caso desta otimização multiobjetivo, o lado esquerdo da operação contém três condições, totalizando vinte e sete regras. O operador utilizado para implicação é o de intersecção de conjuntos, também conhecido como operador Mandani-Min (TSOUKALAS; UHRIG, 1997). A Tabela 5.3 apresentam as regras de inferência definidas para esta otimização. Todos os indivíduos da população do algoritmo genético serão avaliados e receberão um valor de aptidão a partir destas regras.

Tabela 5.3 - Tabela de inferência fuzzy.

	CGr	=	*O*		*Ixr*	=	*O*		θ	=	*O*		*A*	=	*O*
	CGr	=	*O*		*Ixr*	=	*O*		θ	=	*R*		*A*	=	*R*
	CGr	=	*O*		*Ixr*	=	*O*		θ	=	*P*		*A*	=	*P*
	CGr	=	*O*		*Ixr*	=	*R*		θ	=	*O*		*A*	=	*O*
	CGr	=	*O*		*Ixr*	=	*R*		θ	=	*R*		*A*	=	*R*
	CGr	=	*O*		*Ixr*	=	*R*		θ	=	*P*		*A*	=	*P*
	CGr	=	*O*		*Ixr*	=	*P*		θ	=	*O*		*A*	=	*P*
	CGr	=	*O*		*Ixr*	=	*P*		θ	=	*R*		*A*	=	*P*
	CGr	=	*O*		*Ixr*	=	*P*		θ	=	*P*		*A*	=	*P*
	CGr	=	*R*		*Ixr*	=	*O*		θ	=	*O*		*A*	=	*O*
	CGr	=	*R*		*Ixr*	=	*O*		θ	=	*R*		*A*	=	*R*
	CGr	=	*R*		*Ixr*	=	*O*		θ	=	*P*		*A*	=	*P*
	CGr	=	*R*		*Ixr*	=	*R*		θ	=	*O*		*A*	=	*O*
se	*CGr*	=	*R*	e	*Ixr*	=	*R*	e	θ	=	*R*	então	*A*	=	*R*
	CGr	=	*R*		*Ixr*	=	*R*		θ	=	*P*		*A*	=	*P*
	CGr	=	*R*		*Ixr*	=	*P*		θ	=	*O*		*A*	=	*P*
	CGr	=	*R*		*Ixr*	=	*P*		θ	=	*R*		*A*	=	*P*
	CGr	=	*R*		*Ixr*	=	*P*		θ	=	*P*		*A*	=	*P*
	CGr	=	*P*		*Ixr*	=	*O*		θ	=	*O*		*A*	=	*R*
	CGr	=	*P*		*Ixr*	=	*O*		θ	=	*R*		*A*	=	*P*
	CGr	=	*P*		*Ixr*	=	*O*		θ	=	*P*		*A*	=	*P*
	CGr	=	*P*		*Ixr*	=	*R*		θ	=	*O*		*A*	=	*R*
	CGr	=	*P*		*Ixr*	=	*R*		θ	=	*R*		*A*	=	*P*
	CGr	=	*P*		*Ixr*	=	*R*		θ	=	*P*		*A*	=	*P*
	CGr	=	*P*		*Ixr*	=	*P*		θ	=	*O*		*A*	=	*P*
	CGr	=	*P*		*Ixr*	=	*P*		θ	=	*R*		*A*	=	*P*
	CGr	=	*P*		*Ixr*	=	*P*		θ	=	*P*		*A*	=	*P*

onde,

O: é a função de pertinência ótima;

R: é a função de pertinência regular;

P: é a função de pertinência ruim ou péssima;

CGr: é a variável de entrada de centro de gravidade resultante;

Ixr: é a variável de entrada do produto de inércia resultante;

θ: é a variável de entrada do ângulo de inclinação do eixo do foguete;

A: é a variável de saída de aptidão;

A defuzzificação, isto é, a conversão do valor linguístico na saída atribuindo um valor numérico de aptidão, é realizada a partir do cálculo conhecido como centro das somas. Este método leva em conta a sobreposição de várias regras, levando em consideração no valor de saída a contribuição de cada regra, conforme a equação:

$$u = \frac{\sum_{i=1}^{N} u_i \cdot \sum_{k=1}^{n} \mu_{Ok}(u_i)}{\sum_{i=1}^{N} \sum_{k=1}^{n} \mu_{Ok}(u_i)} \tag{5.7}$$

onde,

μ_{Ok} é ponto de máximo da altura das funções de pertinência,

μ_i é a função de pertinência considerada,

u é o valor de pertinência na saída, ou valor da aptidão.

Em seguida, os indivíduos são colocados em ordem decrescente de aptidão. O ordenamento é necessário antes do processo de seleção de indivíduos a realizarem o cruzamento, propiciando que a técnica de pressão de seleção gaussiana funcione corretamente.

5.3.4. Seleção de indivíduos para cruzamento

Sendo o AG uma meta-heurística bioinspirada, a estratégia de seleção deve priorizar que os indivíduos mais aptos tenham maior probabilidade de transferir sua carga genética, como sugerida pela teoria da evolução de Darwin. Geralmente, são utilizadas as técnicas da roleta e do torneio. Estas técnicas visam aumentar a pressão de seleção, isto é, aumentar a probabilidade dos melhores indivíduos contribuírem com sua carga genética para a próxima geração. Caso contrário, realizando uma seleção aleatória uniforme, provoca demora de convergência da população para a otimização, e perda dos indivíduos com boa aptidão. A técnica do sorteio utiliza a probabilidade de seleção proporcional à aptidão dos indivíduos; a técnica do torneio consiste em selecionar dois indivíduos, e o que possui melhor aptidão é selecionado, passando para o processo de cruzamento (LACERDA; CAVALHO, 1999). Estes processos apresentam algumas desvantagens, por exemplo, no método

da roleta, se os indivíduos da população possuírem uma diferença absoluta de aptidão muito pequena entre eles, o que normalmente ocorre depois de algumas gerações, todos os indivíduos terão a mesma probabilidade de serem sorteados para cruzamento, como o exemplo mostrado na Figura 5.7.

Indivíduos	Aptidão	Porcentagem
A	19,8	20,3%
B	19,6	20,1%
C	19,5	20,0%
D	19,3	19,8%
E	19,1	19,6%

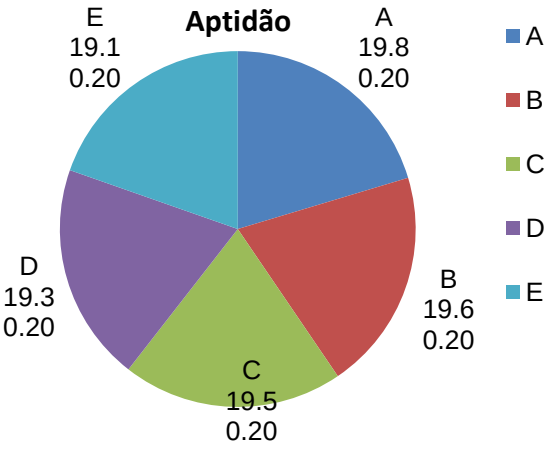


Figura 5.7 - Técnica da roleta com pequena variação de probabilidades.
Fonte: adaptado de Linden (2006).

A técnica implementada foi baseada em um novo conceito de pressão de seleção. Esta nova técnica também consiste em um sorteio, contudo o número aleatório gerado tem um comportamento Gaussiano, isto é, tem uma distribuição normal. O programa, inicialmente, realiza o sorteio de dois números com distribuição uniforme, depois estes são transformados em uma distribuição Gaussiana, como proposto por Box e Muller (1958), através da equação:

$$X_1 = (-2 \log_e U_1)^{\frac{1}{2}} \cos(2\pi U_2). \qquad (5.8)$$

onde,

X_1 é o número gerado aleatoriamente com distribuição normal,

U_1 e U_2 são números aleatórios gerados com distribuição uniforme.

As curvas apresentadas na Figura 5.8.a mostram as curvas teóricas do comportamento das distribuições, sendo a uniforme na cor azul e a normal em verde. Na Figura 5.8.b, estão as curvas dos resultados práticos, obtidos através do algoritmo implementado para sortear números aleatórios.

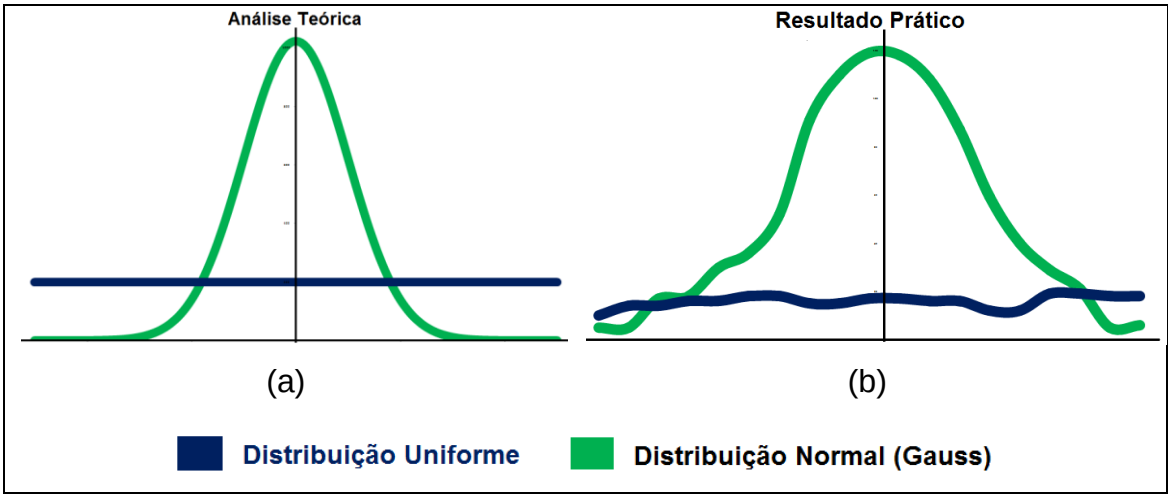


Figura 5.8 - Curvas teóricas de distribuição (a), curvas de resultados práticos (b).

Esta pressão de seleção favorece que os indivíduos mais aptos transfiram sua carga genética, contudo, permite que os menos aptos também o façam com uma menor frequência. Na otimização desejada, os indivíduos com melhor aptidão estão no centro da distribuição, obtendo uma maior chance de serem sorteados.

Esta técnica permite que um indivíduo menos apto, continue com a possibilidade de contribuir com sua característica após o cruzamento, explorando outros espaços de busca, evitando a convergência prematura da população para uma região de ótimo local.

Para validar esta técnica, foram utilizadas funções teste, apresentadas na Tabela 5.4. Esta estratégia é sempre abordada para validação de novas metodologias, como realizadas em: Andre et al (2000), Deb et al (2002) e Galski (2006). Funções matemáticas, que têm várias regiões de ótimos locais, mas somente um valor de ótimo global, são utilizadas em diversos trabalhos para validação de meta-heurística, ou para apresentação de novas abordagens das técnicas de busca.

Tabela 5.4 - Funções teste para a pressão de seleção.

Função	Domínio	Valor do ótimo global
Eason $f(x) = -\cos(x_1)\cos(x_2)\, e^{(-(x_1-\pi)^2-(x_2-\pi)^2)}$	$100 \leq x_i \leq 100$ 2 dimensões	$x = (\pi, \pi)$ $f(x) = -1$
Griewank $f(x) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right)$	$-600 \leq x_i \leq 600$ $n = 1,2,4,6,8,10$	$x = (0, \ldots, 0)$ $f(x) = 0$
Rosembrock $f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$	$-2.048 \leq x_i \leq 2.048$ 2 dimensões	$x = (1,1)$ $f(x) = 0$

Para todas as funções, o algoritmo apresenta eficiência na otimização, encontrando a região do ótimo global. A Figura 5.9 apresenta uma análise da convergência para a função de Griewank. Esta função é considerada crítica, pois, utilizando dez dimensões no espaço de busca, tem as regiões de ótimos locais com aptidão bem próxima da aptidão do ótimo global, tornando difícil a convergência correta desta otimização.

O gráfico mostra que para dimensões mais baixas, o algoritmo consegue realizar a otimização em poucas gerações. Conforme a dimensão do problema é elevada, a convergência é mais lenta, comportamento normal para qualquer implementação de paradigmas de inteligência computacional. Em todas as situações pode-se perceber a convergência das soluções para a minimização da função, validando a funcionalidade da técnica de pressão de seleção gaussiana.

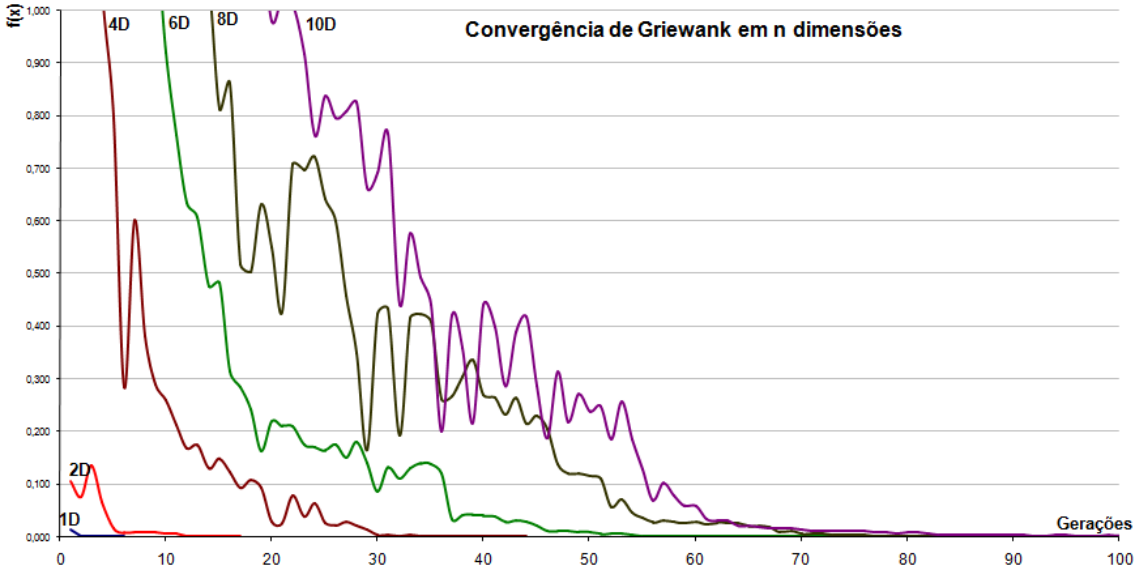


Figura 5.9 - Análise de convergência do AG usando a função teste de Griewank.

5.3.5. Parâmetros de cruzamento e mutação

O operador de cruzamento permite a exploração de regiões desconhecidas no espaço de busca. As técnicas mais aplicadas na representação binária são a utilização de cruzamento de um ponto e a de dois pontos. Existe a possibilidade de utilizar valores maiores que dois pontos, porém, não são frequentemente utilizados em trabalhos de AG. Na técnica de um ponto, é sorteado um ponto de corte do cromossomo, isto é, neste ponto será realizada a troca de informações de cada indivíduo, conforme a Figura 5.10. Os filhos são a mistura de uma parte dos cromossomos dos pais.

Pai 1	**0**	**1**	**1**	**0**	**1**	**0**	**1**	**1**
Pai 2	*1*	*0*	*1*	*1*	*0*	*1*	*1*	*0*
Cruzamento								
Filho 1	**0**	**1**	**1**	*1*	*0*	*1*	*1*	*0*
Filho 2	*1*	*0*	*1*	**0**	**1**	**0**	**1**	**1**

Figura 5.10 - Exemplo de cruzamento de um ponto para representação binária.

Os operadores convencionais não são eficientes na representação real. Nesta técnica, os cromossomos estão representados por:

$pai_1 = (p_{11},\ p_{12}, \ldots, p_{1n}) pai_2 = (p_{21},\ p_{22}, \ldots, p_{2n})$

$filho_1 = (f_{11},\ f_{12}, \ldots, f_{1n}) filho_2 = (f_{21},\ f_{22}, \ldots, f_{2n})$

onde,

$p_{11},\ p_{12},\ p_{21}, p_{22}, f_{11}, f_{12}, f_{21}$ *e* f_{22} são números Reais que representam as grandezas numéricas dos indivíduos.

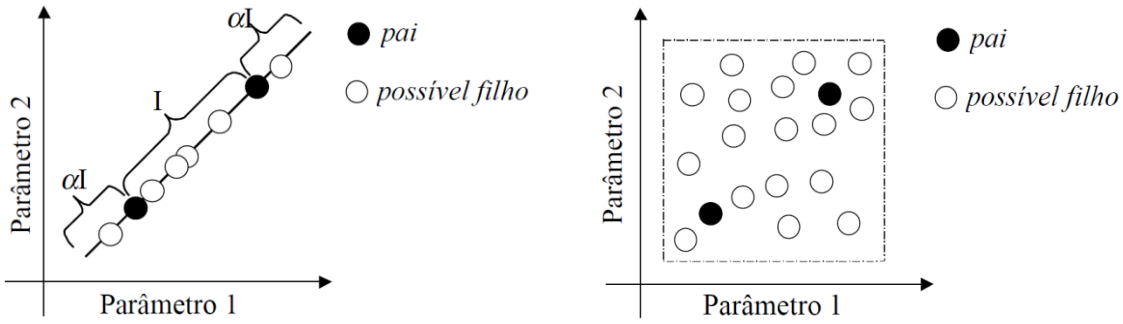
Desta forma, outros métodos de cruzamento foram desenvolvidos. A Tabela 5.5 apresenta diversas implementações de operadores de cruzamento.

Tabela 5.5 - Lista de operadores de cruzamento.

- o cruzamento da média (DAVIS, 1991)	$f = \frac{(p_1 + p_2)}{2}$
- média geométrica	$f = \sqrt{p_1 p_2}$
- linear	$f_1 = 0{,}5p_1 + 0{,}5p_2$ $f_2 = 1{,}5p_1 - 0{,}5p_2$
- aritmético (MICHALEWICZ, 1994)	$f_1 = \beta p_1 + (1 - \beta) p_2$ $f_2 = (1 - \beta) p_1 + \beta p_2$
- BLX-α - *blend crossover* (ESHELMAN e SCHAFFER, 1993)	$f_1 = p_1 + \beta (p_2 - p_1)$ onde, $\beta \in U(-\alpha,\ 1 + \alpha)$, e $\alpha = 0{,}5$

Neste algoritmo foi utilizado o cruzamento BLX-α, ou conhecido como cruzamento de mistura. A vantagem deste método é a busca em várias direções do espaço de busca, pois quando α=0, os filhos podem situar-se sobre um intervalo I entre os pontos dos pais. Aumentado o valor de α, pode-se estender este intervalo de forma linear, como mostra a Figura 5.11a. Quanto ao valor de β, se for igual para todos os pares de genes, mantém o comportamento da Figura 5.11a, porém, se utilizar um valor de β para cada gene, o possível filho pode situar-se em algum ponto de uma região limitada

por um retângulo entre os pontos dos pais, ver Figura 5.11b (LACERDA; CARVALHO, 1999).



(a) (b)

Figura 5.11 - Valor de $\alpha \neq 0$ e β igual (a). Valor de $\alpha \neq 0$ e β diferente (b).
Fonte: Gray; Costanzo; Plesha (2010).

O operador de mutação também possui diversas implementações, dentre as quais as apresentadas na Tabela 5.6.

Tabela 5.6 - Funções Lista de operadores de mutação.

- Uniforme: trocar um gene por um número aleatório;	$c = \begin{cases} U(lim_{inf}, lim_{sup}) \\ p_i \end{cases}$
- Gaussiana: substitui um gene por um número aleatório de uma distribuição normal;	$c = \begin{cases} N(p_i, \sigma) \\ p_i \end{cases}$
- Limite (MICHALEWICZ, 1994): substitui o gene por um dos valores limites do intervalo permitido;	$c = \begin{cases} lim_{inf} \\ lim_{sup} \\ p_i \end{cases}$

Foi utilizado o operador de mutação não-uniforme, proposto por Michalewicz (1994), por apresentar na literatura bons resultados na busca por soluções em outras regiões do espaço, promovendo a diversificação necessária para o AG. O operador é implementado de acordo com as funções:

$$c = \begin{cases} p_i + \left(lim_{inf} - p_i\right) f(G), \ se\ r_1 < 0{,}5 \\ p_i - \left(p_i - lim_{sup}\right) f(G), \ se\ r_1 > 0{,}5, \\ p_i\ , \ caso\ contrário \end{cases} \quad (5.9)$$

onde,

$$f(G) = \left(r_2\left(1 - \frac{G}{G_{max}}\right)\right)^b, \quad (5.10)$$

$r_1\ e\ r_2\ \in U(0{,}1)$ gerado aleatoriamente,

G é a geração corrente,

G_{max} é o número máximo de gerações,

b é um parâmetro do sistema de determina a forma da função.

O cruzamento entre os indivíduos são monitorados e armazenados em cada geração. O objetivo é evitar que nas gerações seguintes exista uma grande quantidade de indivíduos com características semelhantes, isto é, minimizar a perda de diversidade da população. Para isto, uma rotina de verificação permite que um determinado par de indivíduos realize apenas um único cruzamento, contudo um mesmo indivíduo pode participar de outros cruzamentos.

A mutação tem o papel de permitir ao AG a busca em outras regiões do espaço, evitando a convergência prematura da população. Isto é definido por uma taxa de mutação é um valor que define se este processo ocorre ou não após o cruzamento. Na literatura sempre é associado a uma probabilidade baixa de ocorrência, utilizando-se valores entre 0,5% e 1,0%.

No entanto, após algumas gerações, existe a tendência natural de a população adquirir as características do melhor indivíduo da população, criando um subconjunto, conhecido como super indivíduos. Porém, se a taxa de mutação for alta, existe o risco de perda prematura de bons indivíduos.

O processo de mutação foi melhorado com a finalidade de: evitar a convergência prematura dos indivíduos, e permitir que o algoritmo realize uma busca em outras regiões do espaço. Para isto, foi incorporado ao operador de

mutação, a mesma equação $f(G)$, de forma a promover um crescimento da taxa de mutação a cada geração, elevando-se de 0,5% no início até 5% no final das iterações, como apresentado na Figura 5.12, a curva representa o comportamento da taxa de mutação ao longo das gerações.

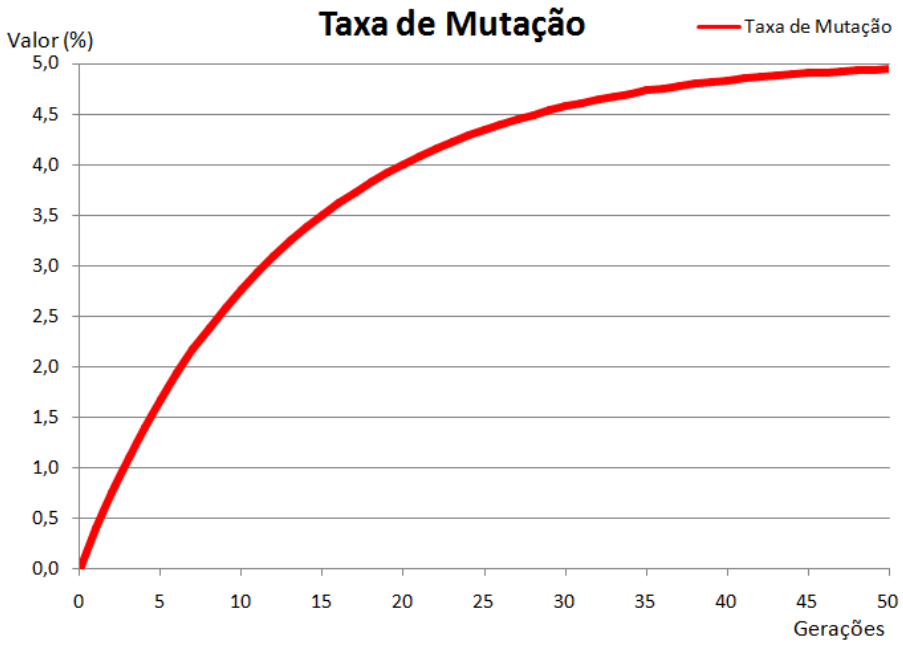


Figura 5.12 - Curva de crescimento da taxa de mutação a cada geração.

5.3.6. Elitismo

Elitismo é uma pequena modificação no algoritmo, não alterando significativamente o tempo de execução, mas garante que o desempenho do AG seja sempre crescente. Consiste em manter uma quantidade de indivíduos da geração anterior que possuem boa aptidão, isto é, eles não morrem, eles avançam à geração seguinte, continuando a contribuir com suas características. A taxa de elitismo é de até 10%, podendo ser menor que este, caso os valores de aptidão os indivíduos da geração atual forem melhores. Obrigatoriamente, os indivíduos desta elite substituem os piores da geração atual. O resultado, apresentado na Figura 5.13, mostra a curva da convergência da população mais acentuada utilizando a técnica de elitismo,

acelerando a otimização, e uma menor oscilação comparada ao AG tradicional, promovendo pouca perda de bons indivíduos durante as gerações.

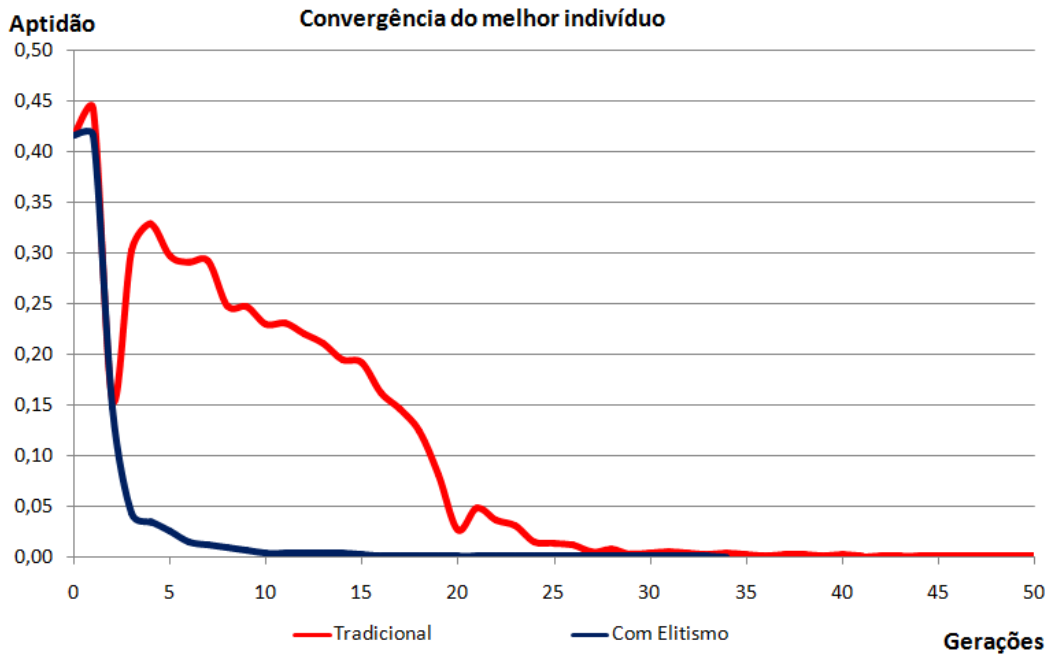


Figura 5.13 - Convergência da aptidão do melhor indivíduo.

5.4. Resultados da alocação na plataforma

A validação do programa foi realizada em duas etapas. A primeira etapa exigia a alocação de experimentos em uma plataforma. Nesta primeira situação foi possível minimizar os valores do centro de gravidade e produto de inércia. A ordem de grandeza dos valores obtidos são consideradas ótimos resultados para esta otimização. O objetivo deste teste não é comparar os resultados com outros, e sim apresentar a capacidade de minimização de parâmetros de balanceamento do algoritmo. O gráfico da Figura 5.14 mostra a minimização dos parâmetros, avaliados através de sua aptidão a cada geração do algoritmo.

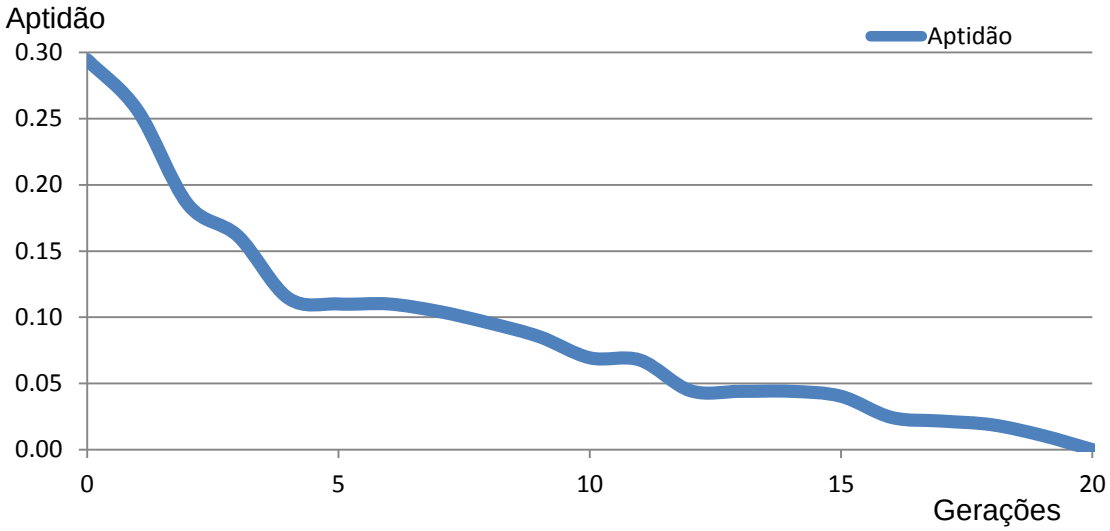


Figura 5.14 - Minimização de parâmetros através da otimização multiobjetivo.

A tabela 5.7 apresenta os parâmetros de inicialização do AG e os resultados de balanceamento dos testes.

Tabela 5.7 - Resultados do teste do AG em uma plataforma.

	Teste 1	**Teste 2**	**Teste 3**
População	1000	1000	1000
Gerações	20	20	20
Objetos	10	10	10
Centro de gravidade (x,y)	$2{,}0 \cdot 10^{-7}$m	$4{,}5 \cdot 10^{-7}$m	$9{,}1 \cdot 10^{-8}$m
Produto de Inércia (Ixz)	$-1{,}8 \cdot 10^{-3}$ kg.m^2	$-8{,}0 \cdot 10^{-3}$ kg.m^2	$1{,}7 \cdot 10^{-3}$ kg.m^2
Produto de Inércia (Iyz)	$6{,}5 \cdot 10^{-3}$ kg.m^2	$-4{,}8 \cdot 10^{-3}$ kg.m^2	$9{,}6 \cdot 10^{-4}$ kg.m^2

A Figura 5.15 foi elaborada em um programa CAD, e mostra a disposição final dos objetos nos três casos de testes apresentados. Os objetos cilíndricos estão apoiados sobre a plataforma, que é circular e plana.

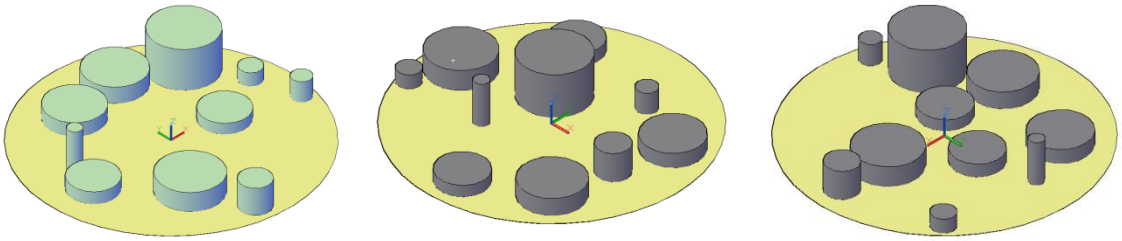


Figura 5.15 - Resultados em CAD dos testes 1, 2 e 3 da esquerda para direita.

5.5. Resultados da alocação na carga útil

A segunda etapa de validação consistiu em alocar experimentos para a carga útil completa, é necessário alocar um total de vinte experimentos, distribuídos em cinco módulos e dez plataformas. Os experimentos estão previamente definidos em qual plataforma devem ser colocados. Esta situação ocorre quando existem muitas restrições entre si, e precisam de uma definição do módulo que deve ser alocado. Tratar estas restrições dentro do algoritmo é inviável, pois os espaços de soluções factíveis são esparsos e descontínuos. Com isto, o algoritmo genético tem a função de otimizar os parâmetros de balanceamento estático e dinâmico. A otimização da margem estática fica parcialmente comprometida, pois a única alteração permitida nesta busca é o posicionamento dos módulos no foguete. Esta abordagem é apresentada nos resultados deste trabalho, por ser esta a característica do foguete utilizado como referência para teste do algoritmo.

Outra possibilidade é tratar sem restrições a alocação dos experimentos, isto é, podem ser alocados em qualquer plataforma e módulo. Nesta abordagem a margem estática tem um campo maior de oscilação, contudo não foi implementado devido à primeira situação ser a representação mais adequada do problema.

Os parâmetros de configuração do AG estão na Tabela 5.8. Os resultados da otimização estão apresentados na Tabela 5.9.

Tabela 5.8 - Parâmetros de configuração do AG.

Indivíduos	Objetos	Gerações
400	20	20

Tabela 5.9 - Resultados da alocação de 20 experimentos em 5 módulos.

Teste	CGr (kg.m)	Ixr (kg.m²)	Io (kg.m²)	θ (°)	tempo (s)
1	0,000	$7{,}41 \cdot 10^{-3}$	1,11	0,007	893
2	0,000	$1{,}76 \cdot 10^{-2}$	1,32	0,017	840
3	0,000	$2{,}22 \cdot 10^{-2}$	1,06	0,021	824
4	0,000	$1{,}26 \cdot 10^{-2}$	1,22	0,012	697
5	$1{,}00 \cdot 10^{-6}$	$1{,}59 \cdot 10^{-2}$	1,08	0,015	1581
6	0,000	$1{,}98 \cdot 10^{-2}$	1,06	0,019	749
7	0,000	$1{,}49 \cdot 10^{-2}$	1,11	0,014	973
8	$1{,}00 \cdot 10^{-6}$	$1{,}40 \cdot 10^{-2}$	1,07	0,015	925
9	0,000	$1{,}19 \cdot 10^{-2}$	1,13	0,011	580
10	$3{,}00 \cdot 10^{-6}$	$1{,}08 \cdot 10^{-2}$	1,13	0,010	1002
Média	$\mathbf{5{,}00 \cdot 10^{-7}}$	$\mathbf{1{,}47 \cdot 10^{-2}}$	**1,13**	**0,014**	**906,5**

Para confiabilidade do voo do foguete, é desejável que os valores do desbalanceamento dinâmico seja menor que 0,10 kg.m², e o ângulo de inclinação do seu eixo menor que 0,25°. Os resultados do algoritmo desenvolvido neste trabalho serão comparados com os valores obtidos na missão do VSB-30 microG1.

A Figura 5.16 mostra a convergência da otimização do centro de gravidade. A linha em vermelho representa o valor do centro de gravidade da carga útil sem os lastros, igual a 0,44 kg.m. A linha verde mostra o resultado após a colocação de lastro para minimizar o centro de gravidade, obtendo o valor de 0,02 kg.m. O algoritmo obteve o resultado médio de 5.10^{-7} kg.m.

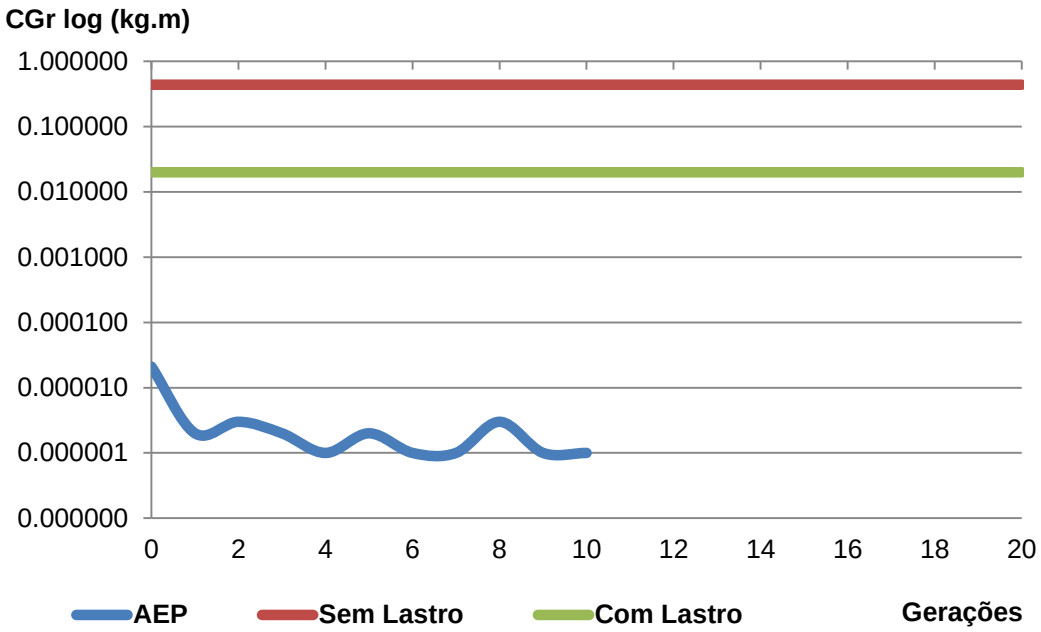


Figura 5.16 - Convergência do balanceamento estático.

O produto de inércia final, sem os lastros foi de 1,11 kg.m², representado pela linha vermelha no gráfico da Figura 5.17. Após a colocação dos lastros chegou-se ao valor de 0,08 kg.m², representado pela linha verde. O algoritmo obteve como melhor resultado $1{,}08.10^{-2}$ kg.m² ao final das iterações. Contudo, a partir da quarta geração, o algoritmo já encontra um balanceamento dinâmico melhor que a do voo analisado, como mostra o comportamento da curva em azul.

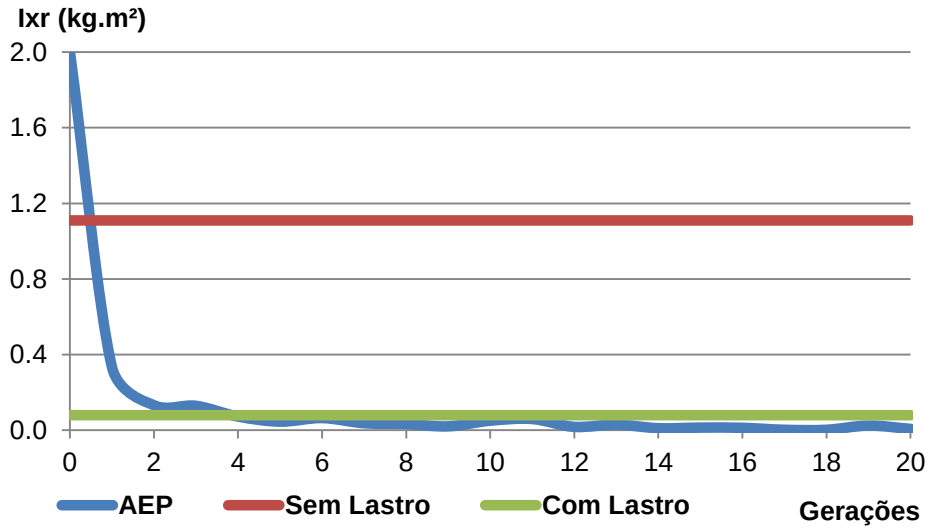


Figura 5.17 - Convergência do balanceamento dinâmico.

Finalmente, analisando o comportamento da inclinação do eixo da carga útil. Na inexistência dos valores do ângulo residual no voo do VSB-30, os resultados do algoritmo foram comparados somente à meta de 0,25°, representado na linha verde no gráfico da Figura 5.18, sendo este o valor máximo permitido da inclinação sem afetar o comportamento de voo. A partir da segunda geração já foi obtido um resultado adequado de inclinação, mas no final das iterações, chegou-se ao valor de 0,014°.

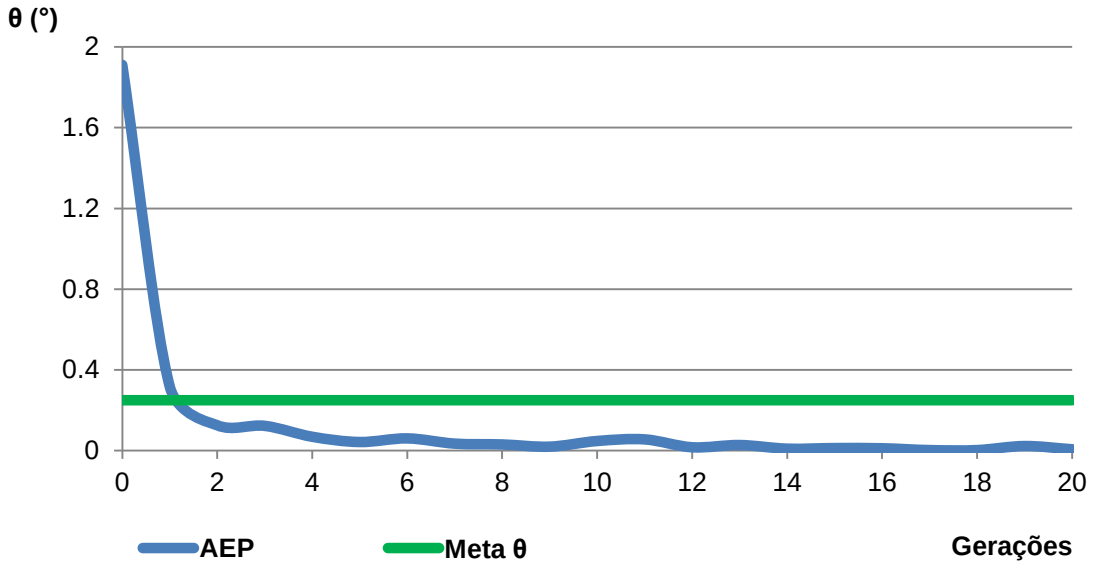


Figura 5.18 - Convergência do ângulo de inclinação do eixo do foguete.

Os dados gerais da carga útil do voo do VSB-30 estão apresentados na Tabela 5.10. Nesta análise verifica-se que aproximadamente 5% (3,71 kg) da capacidade total de experimentos foram ocupados por massas que poderiam ser evitadas, permitindo que houvesse mais espaço físico e margem de carga para alocar novos experimentos. Outro problema da utilização de lastro é seu impacto na trajetória, apogeu e tempo de microgravidade da carga útil.

Tabela 5.10 - Dados gerais da carga útil.

	Massa (kg)
Capacidade máxima da carga útil	400
Estrutura da carga útil	309
Capacidade de Experimentos	**91**
Experimentos	77 (≈ 95%)
Lastro	3,71 (≈ 5%)
Total	**389,7**

Uma análise na otimização multiobjetivo é verificar o comportamento do algoritmo quanto à capacidade de determinar a fronteira de Pareto. Neste caso estudado, apresenta três objetivos conflitantes, pois melhorar a estabilidade estática não está diretamente relacionado à melhoria da estabilidade dinâmica. Ainda soma-se que não existe relação direta da minimização do ângulo de inclinação do eixo da carga útil com os objetivos anteriores. Diante disso, a fronteira de Pareto será analisada em três dimensões, gerando o que denominaremos de Superfície de Pareto.

Por se tratar de grandezas com ordens numéricas muito diferentes, os parâmetros de balanceamento da carga útil foram normalizados para valores no intervalo entre zero e um [0,1]. Este processo permite a visualização gráfica da superfície de Pareto apresentada na Figura 5.19. A superfície se apresenta irregular no início do algoritmo, devido à existência de soluções dominadas e não-dominadas na população.

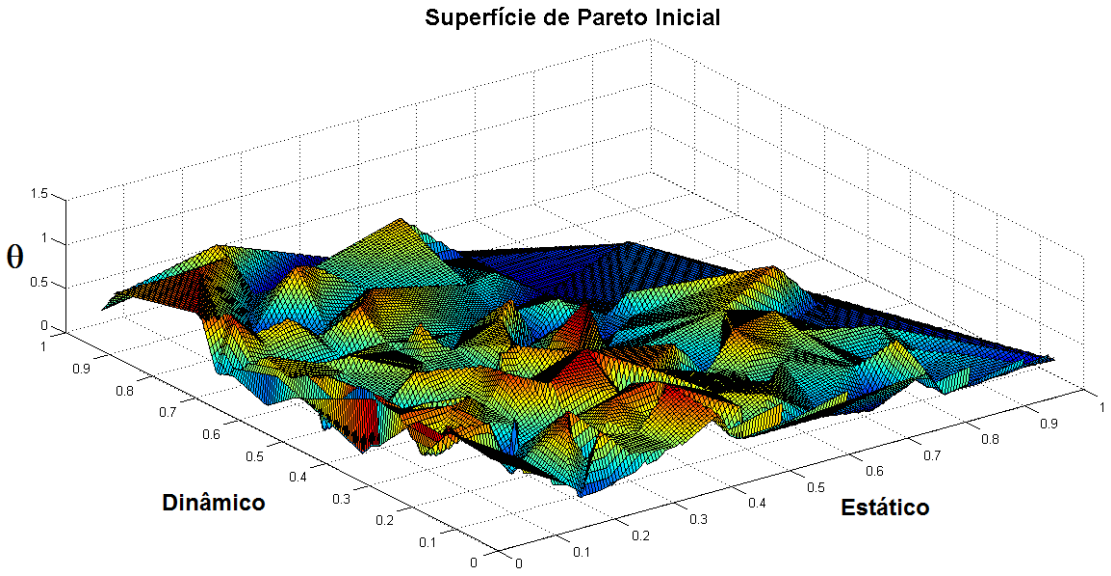


Figura 5.19 - Superfície de Pareto irregular no início da otimização.

Com as iterações do algoritmo genético, a implementação do NGMOGA, elimina gradualmente os indivíduos dominados, pois seus valores de aptidão são piores em relação aos não-dominados. Na última geração a superfície de Pareto apresenta menos oscilações em toda sua extensão, e aparece uma região próxima à origem dos eixos, onde se encontram os indivíduos mais aptos a representar uma solução para a otimização. Na Figura 5.20 temos a representação gráfica da superfície de Pareto resultante da otimização.

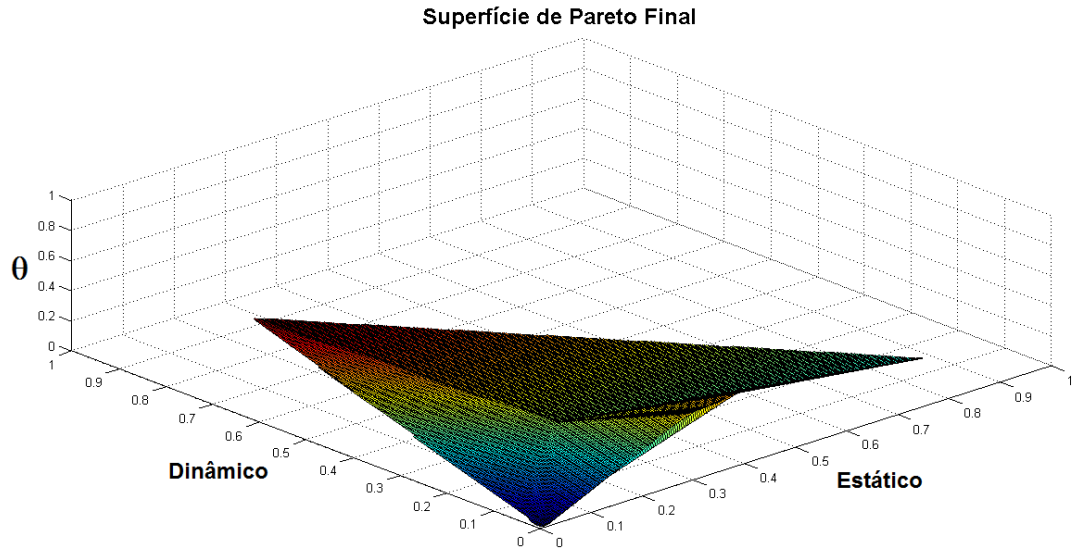


Figura 5.20 - Superfície de Pareto resultante no final da otimização.

A Tabela 5.11 apresenta os dados dos experimentos alocados pelo programa AEP na carga útil do foguete. Os experimentos foram tratados como objetos cilíndricos para todos os casos, porém é apresentada no Anexo B, a possibilidade de solução para experimentos retangulares.

Tabela 5.11 - Experimentos embarcados na carga útil.

Módulo	Prato	Experimento	Raio (mm)	Altura (mm)	Massa (kg)
1	Superior	GPS	20	69	0,32
		ARD	17	13,2	0,05
		MCLI	16	10,2	0,17
		GPS (UFRN/IAE)	25	63	0,44
	Inferior	CADEN	63	272	8,76
		DMLM	77	248	11,10
2	Superior	TCM-A (UFSC)	51	210	6,02
		TCM-B (UFSC)	47	252	7,08
		ECEM-A (UFSC)	55	230	2,24
	Inferior	TCM-C (UFSC)	40	161	2,40
		TCM-D (UFSC)	20	231	5,76
		ECEM-B (UFSC)	24	230	4,74
3	Superior	FORNO (INPE)	45	340	9,60
		UCE (INPE)	40	180	1,60
	Inferior	RDA (DLR)	40	25	0,22
		CRE (IAE)	44	25	0,32
4	Superior	Cartões (Escolas)	95	3	0,36
	Inferior	EEM (Escolas)	120	181	3,84
5	Superior	Fonte Câmera	30	28	0,06
	Inferior	CAIXA (UFRN)	85	250	11,90
Massa Total de Experimentos					**76,98**

O resultado final da alocação está apresentado na Figura 5.21, com a visualização em duas posições da carga útil, com as plataformas, experimentos e coifa.

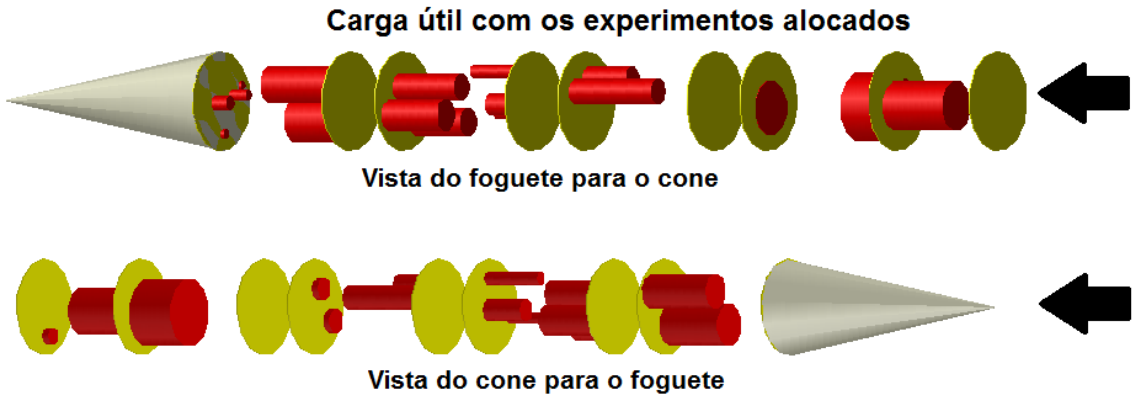


Figura 5.21 - Desenho em CAD da localização final dos experimentos na carga útil.

O desenho da Figura 5.21 foi elaborado em uma ferramenta CAD. O programa AEP fornece na saída os dados de alocação e dos objetos no formato de texto interpretável por esta ferramenta.

A Tabela 5.12 faz um resumo comparativo dos processos do algoritmo AEP com os dados do voo do VSB-30. O programa AEP consegue minimizar os valores de balanceamento estático, dinâmico e do eixo de inclinação atingindo resultados adequados para o voo, tanto quanto os realizados pelo processo tradicional.

Tabela 5.12 - Comparação de resultados do processo.

	CG (kg.m)	Ir (kg.m²)	θ (°)	Tempo (s)
VSB-30 sem lastro	$4{,}4 \cdot 10^{-1}$	$1{,}10 \cdot 10^{0}$	0,037	288.000
VSB-30 com lastro	$2{,}0 \cdot 10^{-2}$	$8{,}00 \cdot 10^{-2}$	0,003	302.000
AEP médio	$3{,}0 \cdot 10^{-7}$	$1{,}53 \cdot 10^{-2}$	0,015	976
AEP ótimo	0,000	$7{,}40 \cdot 10^{-3}$	0,007	893

A grande destaque é a performance do tempo, o AEP é aproximadamente trezentas vezes mais rápido que o processo tradicional. Os resultados do algoritmo foram satisfatórios, atendendo os requisitos especificados em projeto para garantir a qualidade, segurança e sucesso da missão de lançamento.

6 CONCLUSÃO

Alocar os experimentos na carga útil de um foguete de sondagem é importante para garantir seu comportamento e trajetória durante o voo no foguete de sondagem. Para isto, devem-se balancear as cargas de forma a promover a estabilidade estática, dinâmica e do ângulo do eixo de inclinação da carga útil. Este é um problema de otimização multiobjetivo, no qual, algoritmos de otimização evolutivos são comprovadamente eficientes. O algoritmo genético realiza esta otimização obtendo resultados de balanceamento final adequados para o voo do foguete, sem a necessidade de adição de cargas para correção, e em tempo viável, isto é, o programa realiza a atividade aproximadamente de trezentas vezes mais rápido que o método tradicional.

No desenvolvimento do algoritmo genético foi aplicada a técnica de elitismo, visando minimizar a perda de indivíduos com boa aptidão, mantendo-o na geração seguinte, continuando a contribuir com suas características na busca por uma boa solução. No processo de seleção dos indivíduos para cruzamento, foi desenvolvida uma nova técnica de pressão de seleção. Baseado no conceito da teoria de Darwin, no qual os indivíduos mais aptos tem maior probabilidade de sobrevivência, ou de contribuir com características para a próxima geração. Esta nova técnica utiliza o ordenamento dos indivíduos de acordo com sua aptidão, sendo selecionados através de um sorteio com distribuição normal. Desta forma, os melhores indivíduos têm maior probabilidade de serem sorteados, mas os demais indivíduos continuam contribuindo, com menor probabilidade, na otimização dos objetivos em outros espaços de busca.

Outra contribuição no paradigma do algoritmo genético é a variação da taxa de mutação ao longo das iterações. Este procedimento visa permitir que o algoritmo, nas gerações mais avançadas, possa percorrer novos espaços de busca e reduzir o problema de perda prematura de diversidade dos indivíduos.

A característica de inteligência computacional híbrida foi devido à incorporação da lógica Fuzzy no algoritmo genético. A lógica Fuzzy atribuiu valores de aptidão normalizados entre 0 e 1 aos indivíduos, baseados nos resultados das três funções objetivo desta otimização. Através de seu processo de inferência atribuiu variáveis linguísticas avaliando qualitativamente a aptidão de cada indivíduo. Sua implementação visa substituir os métodos tradicionais dos algoritmos genéticos, onde normalmente é utilizada a média aritmética ou a média ponderada de objetivos, onde o processo de atribuir pesos aos objetivos não são triviais.

Finalmente, as análises do algoritmo genético implementado comprovam sua eficiência, ratificando as técnicas implementadas para aplicar em problemas de otimização. Neste trabalho, a otimização consistia em alocar adequadamente os experimentos visando garantir a estabilidade estática e dinâmica da carga útil. Os resultados do programa desenvolvido, denominado como AEP ("Allocator Experiments in the Payload"), foram satisfatórios, atingindo uma configuração de alocação que garante os parâmetros de estabilidade sem a necessidade dos lastros para correção.

Futuramente, o resultado deste trabalho será aplicado nas próximas missões dos foguetes VSB-30, conforme os quinze lançamentos previstos e listados no Apêndice A. Esta metodologia pode ser modificada e ampliada visando atender outros projetos de foguetes, ou ainda, ser aplicado em outras classes de aplicações, como a aeronáutica, automobilística, etc.

REFERÊNCIAS BIBLIOGRÁFICAS

ANDRE, J ET AL. An improvement of the standard genetic algorithm fighting premature convergence in continuous optimization. **Advances in Engineering Software**, v.32, n.1, p.49-60, 2000.

AZUMA, R. M. **Otimização multiobjetivo em problema de estoque eroteamento gerenciados pelo fornecedor**. Dissertação (Mestrado em Engenharia Elétrica) - Unicamp, Campinas-SP, 2011.

BARBOSA, A. N. **Desenvolvimento e avaliação de abordagens de otimização multidisciplinar e sua aplicação em projeto de foguetes de sondagem.** 2013. 256 p. (sid.inpe.br/mtc-m19/2013/03.11.19.44-TDI). Tese (Doutorado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2013. Disponível em: <http://urlib.net/8JMKD3MGP7W/3DMQBH5>. Acesso em: 23 fev. 2015.

BECCENERI, J. C.; SILVA NETO, J. A. **Técnicas de inteligência computacional inspiradas na natureza:** aplicação em problemas inversos em transferência radiativa.São Carlos, SP: SBMAC, 2009. 122 p.

BOX, G. E. P.; MULLER, M. E. A note on the generation of random normal deviates. In: **The annals of mathematical statistics,** 1958. Vol. 29, n°2, p. 610-611.Disponível em<http://projecteuclid.org/euclid.aoms/1177706645>. Acesso em: 15 Mai. 2015.

BOYTON, R.; WIENER, K. **How to calculate mass properties.** Berlin, 1998. (Space Eletronics Inc.CT 06037).

COELLO COELLO, C. A. A comprehensive survey of evolutionary-based multiobjective optimization techniques. **Knowledge and information systems - An International Journal**, p. 269-308, 1999.

DAVIS, L. **Handbook of genetic algorithms**. New York: Van Nostrand Reinhold, 1991.

DEB, K. **Multi-objective optimization using evolutionary algorithms.** New York: John Wiley & Sons, 2001.

DEB, K. ET AL. A fast and elitist multiobjective genetic algorithm: NSGA-II. **IEEE Transactions on Evolutionary Computation**, v.6, n. 2, p.182-197, Abr. 2002.

DEB, K.; MOHAN, M.; MISHIRA, S. **A fast multi-objective evolutionary algorithm for finding well-spread pareto-optimal solutions**. Kanpur: Indian Institute of Technology, 2003.

EDGEWORTH, F. Y. **Mathematical physics.** London: C Kegan Paul & Co,1881.p.79.

ESHELMAN, L. J.; SCHAFFER, D. J. **Real-coded genetic algorithms and interval-schemata.** San Mateo: [s.n.], 1993.

EULER, L. **Récherches sur la conoissance mécanique des corps.** Berlin: Mémoires de l'Académie des Sciences de Berlin, p. 131-153, 1758.

FONSECA, C. M.; FLEMING, P. J. Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. In: INT. CONF. ON GENETIC ALGORITHMS,5., 1993. **Proceedings...**San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.1993.

GALSKI, R. L. **Desenvolvimento de versões aprimoradas híbridas, paralela e multiobjetivo do método da otimização extrema generalizada e sua aplicação no projeto de sistemas espaciais**. 2006. 279 p. (INPE-14795-TDI/1238). Tese (Doutorado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2006. Disponível em: <http://urlib.net/sid.inpe.br/mtc-m17@80/2006/11.30.19.04>. Acesso em: 03 mai. 2015.

GEN, M.; CHENG, R. **Genetic algorithms and engineering optimization.** Ashikaga-Japan: John Wiley & Sons inc., 2000.

GOLDBERG, D. E. **Genetic algorithms in search, optimization, and machine learning.** Boston: Addison-Wesley Longman Publishing Co. Inc, 1989.

GRAY, C. M.; COSTANZO, F.; PLESHA, M. E. **Engineering mechanics.** NewYork: McGraw Hill Global Education Holdings, LLC, 2010.ISBN9780077275549.

HIBBELER, R. C. **Mecânica dinâmica.** 8. ed. Rio de Janeiro: LTC, 1998. v. tradução.

INSTITUTO DE AERONÁUTICA E ESPAÇO. **O foguete de sondagem VSB-30**. [s.l.], [s.n.], 2013. Disponível em: <http://www.iae.cta.br/site/page/view/pt.vsb30.html>. Acesso em: 20 Set. 2015.

JONES, D. F.; MIRRAZAVI, S. K.; TAMIZ, M. Multiobjective meta-heutistics: an overview of the current state-of-the-art. **European Journal of Operational Research,** v. 137, n. 1,p. 1-9, 2001.

KONAK, A. ET AL. Multi-objective optimization using algorithms: a tutorial. **Reliability Engineering & System Safety**,v. 91, n. 5, p. 992-1007, Set. 2006.

LACERDA, E. G. M.; CARVALHO, A. C. P. L. F. **Introdução aos algoritmos genéticos.** Porto Alegre: UFRGS, 1999.

LAGES, W. F. **Tensor de inércia.** Porto Alegre: Universidade Federal do Rio Grande do Sul, 2006. (ENG04479).

LINDEN, R. **Algoritmos genéticos.** Rio de Janeiro: Brasport, 2006. p. 348, ISBN 854-7452-265-1, 2006.

LOBATO, F. S. **Otimização multi-objetivo para o projeto de sistemas de engenharia.**Tese de Doutorado - Universidade Federal de Uberlândia-MG, Uberlândia, 2008. p. 249.

MAIA, L. P. M. **Dinâmica do sistema.** Rio de Janeiro: Ioneli, 1979.

MICHALEWICZ, Z. **Genetic algorithms + data structures = evolution programs.**3. ed. Charlote: Springer-Verlag, 1994.

PALMÉRIO, A. F. **Introdução à tecnologia de foguetes.**7ª. ed. [S.l.]: [s.n.], 2008.ISBN 85-905989-1-8.

PARDALOS, P. M.; RESENDE, M. G. C. **Handbook of applied optimization.** Oxford: University Press, 2003. ISBN 989-0-19-512594-8.

PARETO, V. **Cours d'economie politique.**1. ed. Lausanne: F. Rouge, 1896.

PILLAT, V. G. **Modelagem inteligente da análise de traços de espectros eletromagnéticos reflexivos da ionosfera, com a aplicação do paradigma de lógica nebulosa**. 2012. 117 p. (sid.inpe.br/mtc-m19/2012/11.19.12.07-TDI). Tese (Doutorado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2012.

PORTO, B. F. **Teoria, metodologia de projeto e desenvolvimento de motores foguete a propelente sólido e foguetes experimentais.** Trabalho de conclusão de curso - Pontifícia Universidade Católica do Paraná, Curitiba, 2007.

REEVES, C. Genetic algorithms. In: GLOVER, F; KOCHENBERGER, G. A. (eds.). **Handbook of metaheuristics.**Boston: Kluwer Academic Publishers group, 2003. cap. 3, p. 55-82.

SIMÕES, M. G.; SHAW, I. S. **Controle e modelagem fuzzy.** São Paulo: Edgard Blücher, 1999. ISBN 85-212-0248-2.

TOSCANI, L. V.; VELOSO, P. A. S. **Complexidade de algoritmos.** Porto Alegre: Sagra-Luzzato, 2001.

TSOUKALAS, L. H.; UHRIG, R. E. **Fuzzy and neural approaches engineering.** 1. ed. New York: John Wiley & Sons, 1997.

ZADEH, L. A. Fuzzy sets**. Information Control,** v. 8, p. 338–353,1965.

APÊNDICE A - MISSÕES FUTURAS DO VSB-30

A lista a seguir apresenta as próximas missões previstas de lançamento do VSB-30.

Lançamento	Missão	Local
Abril /2016	MAIUS 1	ESRANGE - Suécia
Maio / 2016	MAPHEUS 6	ESRANGE - Suécia
Julho / 2016	HIFIRE 4	Austrália
Outubro / 2016	Micro G2	CLA - Brasil
Novembro / 2016	INPE 15 (VS-30)	CLA - Brasil
Março / 2017	TEXUS 54	ESRANGE - Suécia
Março / 2017	TEXUS 55	ESRANGE - Suécia
Abril / 2017	MAIUS 2	ESRANGE - Suécia
Maio / 2017	MAPHEUS 7	ESRANGE - Suécia
Novembro / 2017	HIFIRE 5	Austrália
Março / 2018	MAIUS 3	ESRANGE - Suécia
Abril / 2018	MASER 14	ESRANGE - Suécia
Maio / 2018	ATEK	ESRANGE - Suécia
Novembro / 2018	TEXUS 56	ESRANGE - Suécia
Novembro / 2018	TEXUS 57	ESRANGE - Suécia

APÊNDICE B - RESUMO DO ARTIGO SUBMETIDO À JATM

TITLE: MULTI-OBJECTIVE OPTIMIZATION WITH GENETIC ALGORITHM TO BALANCE ROCKET'S PAYLOAD

Abstract: The allocation of experiments in the payload of a rocket influence their behavior during flight. The main objective of this contribution is to ensure load balancing in the distribution of experiments on the platforms. This balance is composed of three independent parameters, characteristics of a multi-objective problem. It should then be optimize, trying to find the appropriate values for: center of gravity, product of inertia and moment of inertia. The final position of the center of gravity and product of inertia should coincide with the main axis of rocket rotation. If there is a significant residual unbalance, during the flight phase in which the rocket starts rolling around its axis, there are forces that tend to generate a misaligned axis with the respect to the shaft rocket reference, causing instability in flight, risking the success of the mission. Determine the best layout configurations of the experiments in the payload may require a relatively long time if executed manually or by computer modeling based on brute force, also known as trial and error. Using meta-heuristics is a viable option, because they have satisfactory results and were consolidated in several previous works. The genetic algorithm is a meta-heuristic inspired by the natural evolution of species, as shown in Darwin's theory. This tool adjusted to the specific requirements of this problem allows to obtain a set of feasible solutions to the experiments allocation, ensuring the rocket flight safety.

APÊNDICE C - RESTRIÇÃO DE EXPERIMENTOS RETÂNGULARES

Não é permitida a sobreposição de experimentos, dois objetos distintos o_i e o_j ($i \neq j$), numa mesma superfície do prato, porém, alguns pratos receberão experimentos em sua superfície superior e outros na inferior.

Para análise da sobreposição será utilizado o método de localização de pontos em um polígono. Cada objeto que será alocado possui quatro pontos que definem os vértices do quadrilátero. O algoritmo deve verificar se cada um destes está localizado na região interna de outro objeto, caracterizando assim a sobreposição.

O polígono mais básico é o triângulo, então este será utilizado no modelo. Para isto o formato quadrilátero do objeto será dividido em sua diagonal, formando dois triângulos, conforme Figura A.1, em seguida será verificado se os pontos dos vértices de outros objetos estão na região interna ou externa destas figuras.

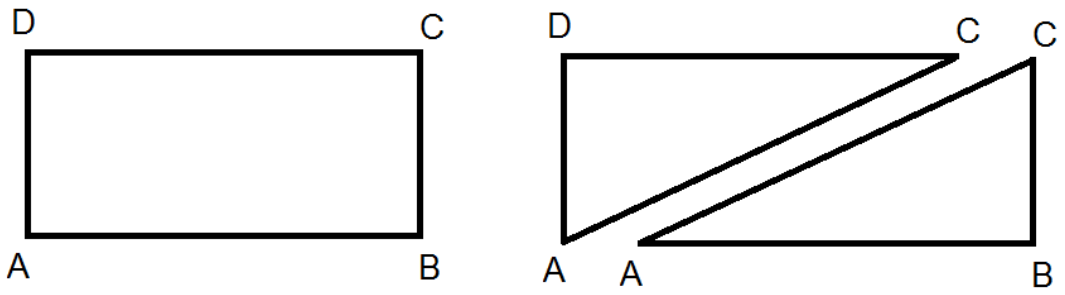


Figura A.1 - Divisão do objeto em figuras triangulares.

O teste de sobreposição utiliza um método baseado em coordenadas baricêntricas do triângulo. Conforme o método, um ponto p qualquer em um plano R^2 pode ser escrito na forma:

$$p = \lambda_1.p_1 + \lambda_2.p_2 + \lambda_3.p_3 \qquad \text{(A.1)}$$

onde os coeficientes λ_1, λ_2 e λ_3 são números reais, satisfazendo:

$$\lambda_1 + \lambda_2 + \lambda_3 = 1 \qquad \text{(A.2)}$$

Estes coeficientes λ representam as coordenadas baricêntricas de p em relação aos pontos p_1, p_2 e p_3, apresentado na Figura A.2.

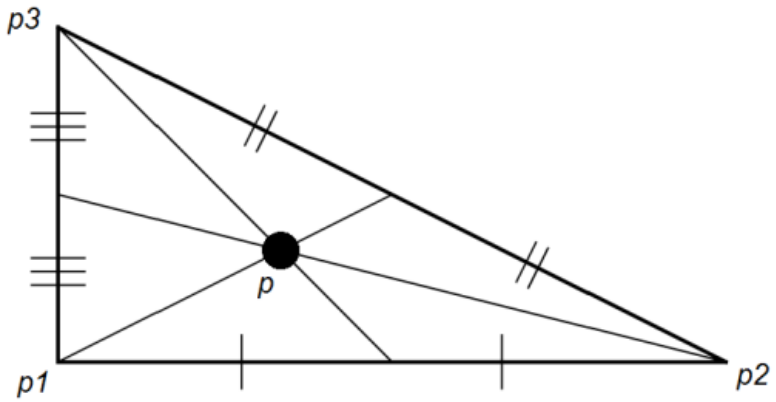


Figura A.2 - Pontos dos vértices e o baricentro de um triângulo.

Os valores dos coeficientes λ_1, λ_2 e λ_3 podem ser determinados através da solução de um sistema de três equações:

$$\begin{aligned} \lambda_1.x_1 + \lambda_2.x_2 + \lambda_3.x_3 &= x_p \\ \lambda_1.y_1 + \lambda_2.y_2 + \lambda_3.y_3 &= y_p \\ \lambda_1 + \lambda_2 + \lambda_3 &= 1 \end{aligned} \qquad \text{(A.3)}$$

onde, $p = (x_p, y_p)$ é um ponto qualquer em R^2, e $p_i = (x_i, y_i)$ para $i = 1, 2\ e\ 3$ são os pontos que definem os vértices do triângulo. Aplicando a regra de Cramer, temos:

$$\lambda_1 = \frac{\begin{vmatrix} x_p & x_2 & x_3 \\ y_p & y_2 & y_3 \\ 1 & 1 & 1 \end{vmatrix}}{\begin{vmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{vmatrix}}, \quad \lambda_2 = \frac{\begin{vmatrix} x_1 & x_p & x_3 \\ y_1 & y_p & y_3 \\ 1 & 1 & 1 \end{vmatrix}}{\begin{vmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{vmatrix}} \quad \text{e} \quad \lambda_3 = \frac{\begin{vmatrix} x_1 & x_2 & x_p \\ y_1 & y_2 & y_p \\ 1 & 1 & 1 \end{vmatrix}}{\begin{vmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{vmatrix}} \qquad \text{(A.4)}$$

A análise do sinal das coordenadas baricêntricas indica a região do plano em que se encontra o ponto $p = (x_p, y_p)$. Caso os três valores sejam maiores que zero, $\lambda_1 > 0, \lambda_2 > 0\ e\ \lambda_3 > 0$, o ponto p está dentro do triângulo, invalidando esta solução de alocação, como mostra a Figura A.3.

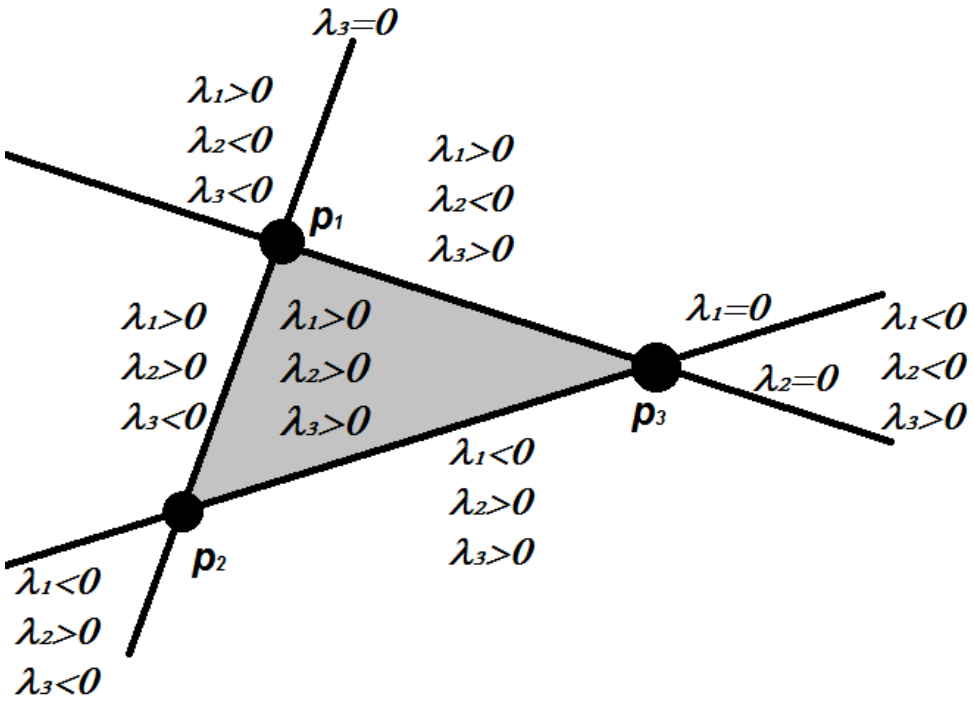


Figura A.3 - Análise dos sinais das coordenadas baricêntricas.

APÊNDICE D - CÓDIGO FONTE DO PROGRAMA

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>
#include "math.h"

#define INDIVIDUOS 400        //Linhas da Matriz populacao
#define OBJETO 10
#define CARACTERISTICAS 2   // y, z,
#define PARAMETROS 13
//0CGy 1CGz 2CGx 3Iyy 4Izz 5Ixx 6Ixy 7Ixz 8Iyz 9CGr 10Ixr 11Io 12teta
#define GERACOES 20
#define PI 3.14159265358979323846
#define aptidao_meta  0.000001 //Criterio de parada
#define qtde_regras 27        //Regras Fuzzy

/*************************************************************/
//VARIAVEIS GLOBAIS
double limite_inf=-150.0, limite_sup=150.0, Rmax=170;
double **pop_dim, **new_pop_dim, **elite;  //Matriz de população
double **pop_parametros;      //Matriz de parametros da populacao
int qtde_elite = INDIVIDUOS/10;     //Define a quantidade da elite
double *aptidao,*new_aptidao, *elite_aptidao;
double populacao[1], valor, maximos[PARAMETROS];
double v1,v2,s,r;       //Utilizados no gerador numero aleatório
int *cruzamentos; //Vetor analisa cruzamentos [pai1pai2] concatenado

double cg_max=0.0, mi_max=0.0, pi_max=0.0, Iresidual=0.0, CGf=0.0;
double CGy=0.0, CGz=0.0, CGx=0.0;
double Iyy=0.0, Izz=0.0, Ixx=0.0;
double Ixy=0.0, Ixz=0.0, Iyz=0.0;
double CGr=0.0, Ixr=0.0, Io=0.0, teta=0.0;
double h, Raio,y,z;
int DIM = CARACTERISTICAS*OBJETO;  //Colunas da Matriz populacao
int restricao = 0;            //Avalia a solucao 0=VALIDA 1=INVALIDA
int i,k,d;                    //Indices de laços for
int s1,s2,geracao,mutacoes=0,crossover=0;
int ind_massa=2;
int cont_rest = 0;
int cont_aptos = 0;

//Variaveis do Fuzzy
float cg = 0.20, in = 0.30, teta_f = 0.95, e=0.0;
float t_cg, t_in, saida=0.0;
int v=0;
int regras_ativas[qtde_regras];
int ativas_cgfuz[qtde_regras];
int ativas_ifuz[qtde_regras];
int ativas_tfuz[qtde_regras];
float u[qtde_regras];
float cg_o, cg_r, cg_p, i_o, i_r, i_p, t_o, t_r, t_p;
float DOF[qtde_regras];
```

```
//Matriz características dos OBJETOS[RAIO][ALTURA][MASSA][altX][PRATO]
//[PRATO] convertido altura do centro do objeto ate o referencial

//Dados VSB30 uG1
//Matriz caracteristicas dos
OBJETOS[RAIO][ALTURA][MASSA][altX][PRATO][MODULO]

float objetos[20][6]= {
{20,69,0.32,0,10,1},{17,13.2,0.05,0,10,1},{16,10.2,0.17,0,10,1},{25,63
,0.44,0,10,1},{63,272,8.76,0,9,1},{77,248,11.1,0,9,1},{51,210,6.02,0,8
,2},{47,252,7.08,0,8,2},{55,230,2.24,0,8,2},{40,161,2.4,0,7,2},{70,231
,5.76,0,7,2},{75,230,4.74,0,7,2},{90,340,9.6,0,6,3},{40,180,1.6,0,6,3}
,{40,25,0.22,0,5,3},{44,25,0.32,0,5,3},{95,3,0.36,0,4,4},{120,181,3.84
,0,3,4},{30,28,0.06,0,2,5}, {85,250,11.9,0,1,5} };

void ajusteXcgObjetos();
void encontraMaximos();
float fuzzy(float cg, float in, float teta_t);

/*************************************************************/
//FUNCAO OBJETIVO CG
double funcaoObjetivoCG (int a)
{
      valor = 0;
      double smy = 0.0, smz = 0.0, smx = 0.0, soma_massa=0.0;
      double parcial = 0.0;
      double y,z,x;
      int o=0;
      for (k=0; k<DIM; k=k+CARACTERISTICAS)
            {
                  y = pop_dim[a][k];
                  z = pop_dim[a][k+1];
                  x = objetos[o][3];
                  smy = smy + y*objetos[o][ind_massa];
                  smz = smz + z*objetos[o][ind_massa];
                  smx = smx + x*objetos[o][ind_massa];
                  soma_massa = soma_massa + objetos[o][ind_massa];
                  o++;
            }
      smy = smy/soma_massa;
      smz = smz/soma_massa;
      smx = smx/soma_massa;
      parcial = (smy*smy)+(smz*smz);
      valor = sqrt(parcial);
      CGy = smy; CGz = smz; CGx = smx;
      CGr = sqrt((CGy*CGy)+(CGz*CGz));
      return(valor);
}

/*************************************************************/
//FUNCAO OBJETIVO MI
double funcaoObjetivoMI (int a)
{
valor = 0;
double mi_objeto = 0.0, miy_temp = 0.0, miz_temp = 0.0, mix_temp=0.0;
double y,z,x;
```

```
int o=0;
for (k=0; k<DIM; k=k+CARACTERISTICAS)
      {
      y = pop_dim[a][k];
      z = pop_dim[a][k+1];
      x = objetos[o][3];
      mi_objeto = (objetos[o][2] * objetos[o][0] * objetos[o][0])/2;
      mix_temp = mix_temp + (mi_objeto + (x*x*objetos[o][2]));
      mi_objeto = (objetos[o][2]/12) *
      (3*(objetos[o][0]*objetos[o][0])+(objetos[o][1]*objetos[o][1]));
      miy_temp = miy_temp + (mi_objeto + (y*y*objetos[o][2]));
      miz_temp = miy_temp + (mi_objeto + (z*z*objetos[o][2]));
      o++;
      }
      Ixx = mix_temp;
      Iyy = miy_temp;
      Izz = miz_temp;
      valor = miy_temp + miz_temp;
      Io = Iyy + Izz;
      return(valor);
}

/*****************************************************************/
//FUNCAO OBJETIVO PI
double funcaoObjetivoPI (int a)
{
      valor = 0;
      double pi_xy=0.0, pi_xz=0.0, pi_yz=0.0;
      double y,z,x;
      int o=0;
      for (k=0; k<DIM; k=k+CARACTERISTICAS)
            {
                  y = pop_dim[a][k];
                  z = pop_dim[a][k+1];
                  x = objetos[o][3];
                  pi_xy = pi_xy + x*y*objetos[o][ind_massa];
                  pi_xz = pi_xz + x*z*objetos[o][ind_massa];
                  pi_yz = pi_yz + y*z*objetos[o][ind_massa];
                  o++;
            }
      Ixy = pi_xy;
      Ixz = pi_xz;
      Iyz = pi_yz;
      valor = Iyz;
      Ixr = sqrt( (Ixy*Ixy) + (Ixz*Ixz) );
      return(valor);
}

/*****************************************************************/
//FUNCAO RESTRICAO RAIO DA PLATAFORMA
double Restricao_Raio (int a)
{
      int o=0;
      for (k=0; k<DIM; k=k+CARACTERISTICAS)
            {
                  y = new_pop_dim[a][k];
```

```
                  z = new_pop_dim[a][k+1];
                  h = sqrt((y*y)+(z*z));
                  //printf("y=%.3lf\tz=%.3lf\th=%.3lf\n",y,z,h);
                  Raio = h + objetos[o][0];
                  if(Raio>Rmax)
                  {
                        restricao = 1; cont_rest++;
                  }
                  o++;
            }
      return(restricao);
}

/*************************************************************/
//FUNCAO RESTRICAO SOBREPOSICAO
//Matriz caract OBJETOS[RAIO][ALTURA][MASSA][altX][PRATO][MODULO]
double Restricao_Sobreposicao (int a)
{
      double h, H;            //h distancia entre centros dos objetos
      double yi,zi,yj,zj;     //H Soma dos Raios de 2 Objetos
      double r_A,r_B, h_obj1, h_obj2, verifica;
      int o=0,oo=1;
      restricao = 0;
      // ANALISE DE SOBREPOSIÇAO NO PLANO
      for (k=0; k<DIM;)
      {
            for(d=k+2;d<DIM;d=d+2)
            {
            if(objetos[o][4] == objetos[oo][4] || objetos[o][5] ==
objetos[oo][5])
                  {
                  //printf("Calcula raios! \n");
                  yi = new_pop_dim[a][k]; zi = new_pop_dim[a][k+1];
                  r_A = objetos[o][0];
                  yj = new_pop_dim[a][d]; zj = new_pop_dim[a][d+1];
                  r_B = objetos[oo][0];
                  h = ((yi-yj)*(yi-yj))+((zi-zj)*(zi-zj));
                  h = sqrt(h);      H = r_A + r_B;
                  //printf("Objetos [%i] x [%i]\n",o,oo);
                  if(h<H)
                  {
                        if(objetos[o][4] == objetos[oo][4])
                        {
                        //printf("Restricao prato\n");
                        restricao = 1;  cont_rest++;
                        k=DIM*DIM;  d=k;  //Força saída loop
                        }
                        //testar altura do mÃ³dulo
                        else
                        {
                              int pratoA = objetos[o][4];
                              int pratoB = objetos[oo][4];
                              if (pratoA % 2 == 0)
                              {     h_obj1 = objetos[o][3] -
objetos[o][1]/2;  }
                              else
```

```
                                    {     h_obj1 = objetos[o][3] +
                                    (objetos[o][1]/2);      }

                                    if (pratoB % 2 == 0)
                                    {
                                    h_obj2 = objetos[oo][3] -
                                    objetos[oo][1]/2; }
                                    else
                                    {
                                    h_obj2 = objetos[oo][3] +
                                    objetos[oo][1]/2; }

                                    if(pratoA < pratoB)
                                    {
                                    verifica = h_obj2 - h_obj1;   }
                                    else
                                    {
                                    verifica = h_obj1 - h_obj2;   }

                                    if(verifica <= 0.0)
                                    {
                                    restricao = 1;
                                    cont_rest++;
                                    k=DIM*DIM;  d=k;  //Força saída loop

                                    }
                              }
                              }
                        }
                        oo++;
                        }
                  o++;
                  oo=o+1;
                  k+=2;
      }
      return(restricao);
}

/*************************************************************/
void ajusteXcgObjetos()
{
for(int f=0; f<OBJETO; f++)
{
      int opcao = objetos[f][4];
      switch (opcao)
      {
      case 0:
            objetos[f][3]= (objetos[f][1]/2);
            break;
      case 1:
            objetos[f][3]= 606.0 + (objetos[f][1]/2) - CGx;
            break;
      case 2:
            objetos[f][3]= 1019.0 - (objetos[f][1]/2) - CGx;
            break;
      case 3:
```

```
            objetos[f][3]= 1019.0 + (objetos[f][1]/2) - CGx;
            break;
      case 4:
            objetos[f][3]= 1549.0 - (objetos[f][1]/2) - CGx;
            break;
      case 5:
            objetos[f][3]= 1772.0 + (objetos[f][1]/2) - CGx;
            break;
      case 6:
            objetos[f][3]= 2302.0 - (objetos[f][1]/2) - CGx;
            break;
      case 7:
            objetos[f][3]= 2525.0 + (objetos[f][1]/2) - CGx;
            break;
      case 8:
            objetos[f][3]= 3055.0 - (objetos[f][1]/2) - CGx;
            break;
      case 9:
            objetos[f][3]= 3278.0 + (objetos[f][1]/2) - CGx;
            break;
      case 10:
            objetos[f][3]= 3808.0 - (objetos[f][1]/2) - CGx;
            break;
      default:
            printf("ExceÃ§Ã£o\n");
      }
      }
}

/*****************************************************************/
void alturaPratoCAD()
{
for(int f=0; f<OBJETO; f++)
{
      int opcao = objetos[f][4];
      switch (opcao)
      {
      case 0:
            objetos[f][3]= 0.0;
            break;
      case 1:
            objetos[f][3]= 606.0;
            break;
      case 2:
            objetos[f][3]= 1019.0 - (objetos[f][1]);
            break;
      case 3:
            objetos[f][3]= 1019.0;
            break;
      case 4:
            objetos[f][3]= 1549.0 - (objetos[f][1]);
            break;
      case 5:
            objetos[f][3]= 1772.0;
            break;
      case 6:
```

```
            objetos[f][3]= 2302.0 - (objetos[f][1]);
            break;
      case 7:
            objetos[f][3]= 2525.0;
            break;
      case 8:
            objetos[f][3]= 3055.0 - (objetos[f][1]);
            break;
      case 9:
            objetos[f][3]= 3278.0;
            break;
      case 10:
            objetos[f][3]= 3808.0 - (objetos[f][1]);
            break;
      default:
            printf("ExceÃ§Ã£o\n");
      }
      }
}

/*****************************************************************/
//Gerador de numeros aleatorios com dispersao de Gauss
float aleatorio_gauss()
{
do {
      v1 = 2.0 * ((float) rand()/RAND_MAX) - 1;
      v2 = 2.0 * ((float) rand()/RAND_MAX) - 1;
      s = v1*v1 + v2*v2;
      } while ( s >= 1.0 );
if (s == 0.0)
{     r = 0.0;}
else
{     r = (v1*sqrt(-2.0 * log(s) / s))/4;}
if(r<0)
      r = r * (-1);
if(r>1)
      r = 1.0;
return(r);
}

/*****************************************************************/
void calculaInercia()
{
for (i=0; i<INDIVIDUOS; i++)
      {
      //Calculo CG
      CGx=0.0;
      ajusteXcgObjetos();
      funcaoObjetivoCG(i);
      pop_parametros[i][0] = CGy;
      pop_parametros[i][1] = CGz;
      pop_parametros[i][2] = CGx;
      ajusteXcgObjetos();
      //Calculo Momento Inercia
      funcaoObjetivoMI(i);
      pop_parametros[i][3] = Iyy;
```

```
      pop_parametros[i][4] = Izz;
      pop_parametros[i][5] = Ixx;
      //Calculo Produto de Inercia  CGr Ixr Io teta
      funcaoObjetivoPI(i);
      pop_parametros[i][6] = Ixy;
      pop_parametros[i][7] = Ixz;
      pop_parametros[i][8] = Iyz;
      //Calculo Desbalanceamentos Residuais
      pop_parametros[i][9] = CGr;
      pop_parametros[i][10] = Ixr;
      pop_parametros[i][11] = Io;

      teta = (2*Ixr)/(Ixx-Io);
      teta = 0.5 *( atan( teta ) ) * 180 / PI;
      pop_parametros[i][12] = teta;
      }
}

/*************************************************************/
double calculaAptidao()
{
//0CGy 1CGz 2CGx 3Iyy 4Izz 5Ixx 6Ixy 7Ixz 8Iyz 9CGr 10Ixr 11Io 12teta
for(int i=0; i<INDIVIDUOS; i++)
      {
      CGr = pop_parametros[i][9];
      Ixr = pop_parametros[i][10];
      Io = pop_parametros[i][11];
      teta = pop_parametros[i][12];
      if(teta < 0.0)
            {     teta = teta * (-1.0);   }
      CGr = CGr/maximos[9];
      Ixr = Ixr/maximos[10];
      Io = Io/maximos[11];
      teta = teta/maximos[12];
      aptidao[i] = fuzzy(CGr,Ixr,teta);
      printf("%.3lf;%.3lf;%.3lf;%.3lf\n",CGtotal,Ixy,Ixz,aptidao[i]);

      }
return (aptidao[INDIVIDUOS]);
}

/*************************************************************/
//BUBBLESORT - ordenaçao do vetor Aptidao Populacao
void bubbleSort(double x[], double **p)
{
double apt_temp, pop_temp;
int i,j,troca;
troca = 1; // troca = 1 houve troca - continua no loop
for(j=INDIVIDUOS-1; (j>=1) && (troca ==1); j--)
      {
      troca = 0; // Nao houve troca, encerra loop
      for(i=0; i<j; i++)
      {
            if(x[i]>x[i+1])
            {
                  //Troca vetor aptidao
```

```
                  apt_temp = x[i];
                  x[i] = x[i+1];
                  x[i+1] = apt_temp;
                  //troca vetor populacao
                  for(d=0; d<DIM; d++)
                  {
                        pop_temp = p[i][d];
                        p[i][d] = p[i+1][d];
                        p[i+1][d] = pop_temp;
                  }
                  troca = 1;
            }
      }
      }
      //return(populacao);
}

/*****************************************************************/
void encontraMaximos()
{
      double conversao=0.0;
      for(int dd=0; dd<PARAMETROS ; dd++)
      {     maximos[dd] = 0.0;      }

      for(i=0; i<INDIVIDUOS; i++)
      {
            for(int dd=0; dd<PARAMETROS ; dd++)
            {
                  conversao = pop_parametros[i][dd];
                  if(pop_parametros[i][dd]< 0)
                  {     conversao = conversao * (-1); }

                  if(conversao > maximos[dd])
                  {     maximos[dd] = conversao;      }
            }
      }
}

/*****************************************************************/
void main()
{
int semente = 1454062488;  //Fixo para analisar confiabilidade
srand(semente);

//Alocaçao Dinaminca da Matriz Populaçao na memoria
pop_dim = (double **)calloc(INDIVIDUOS,sizeof(double *));
for(i= 0; i < INDIVIDUOS; i++)
{     pop_dim[i] = (double *)calloc(DIM,sizeof(double));    }

new_pop_dim = (double **)calloc(INDIVIDUOS,sizeof(double *));
for(i= 0; i < INDIVIDUOS; i++)
{     new_pop_dim[i] = (double *)calloc(DIM,sizeof(double));      }

pop_parametros = (double **)calloc(INDIVIDUOS,sizeof(double *));
for(i= 0; i < INDIVIDUOS; i++)
```

```
{     pop_parametros[i] = (double*)calloc(PARAMETROS,sizeof(double));}

elite = (double **)calloc(qtde_elite,sizeof(double *));
for(i= 0; i < qtde_elite; i++)
{     elite[i] = (double *)calloc(DIM,sizeof(double));     }

if(!new_pop_dim)
{
      printf("** Erro: Memoria Insuficiente **");
      exit(0);
}

//Alocaçao do Vetor Aptidao
aptidao = (double *)calloc(INDIVIDUOS,sizeof(double));
new_aptidao = (double *)calloc(INDIVIDUOS,sizeof(double));
elite_aptidao = (double *)calloc(qtde_elite,sizeof(double));
cruzamentos = (int *)calloc(INDIVIDUOS/2,sizeof(int));

double start_t, end_t, tempo;
double numero_rand;
start_t=clock();  //Inicio contagem de tempo
//Zera Matriz Cruzamentos
int initial_cross=0;
initial_cross*=pow(10,ceil(log10(INDIVIDUOS+1)));
initial_cross+=INDIVIDUOS;
initial_cross*=pow(10,ceil(log10(INDIVIDUOS+1)));
initial_cross+=INDIVIDUOS;

geracao = 0;

//GERANDO POPULACAO INICIAL - Verificado e Validado
printf("Programa iniciado...\n");
int restricao_raio = 0;
int obj = 0;
for(i=0; i<INDIVIDUOS; i++)
      {
      do{
            obj = 0;
            restricao = 0;
            for(d=0; d<DIM; d+=2)
                  {
                  do{
                        restricao_raio = 0;
                        //Sorteio do valor de Y
                        numero_rand = rand()%RAND_MAX;
                        valor = (((limite_sup-
limite_inf)*numero_rand)/RAND_MAX)+limite_inf;
                        new_pop_dim[i][d] = valor;
                        //Sorteio do valor de Z
                        numero_rand = rand()%RAND_MAX;
                        valor = (((limite_sup-
limite_inf)*numero_rand)/RAND_MAX)+limite_inf;
                        new_pop_dim[i][d+1] = valor;
                        y = new_pop_dim[i][d];
                        z = new_pop_dim[i][d+1];
                        h = sqrt((y*y)+(z*z));
```

```
                        Raio = h + objetos[obj][0];
                        if(Raio>Rmax)
                              {
                                    restricao_raio = 1; cont_rest++;
                              }
                  }while(restricao_raio == 1);
                  restricao_raio = 0;
                  obj++;
            }
            CGx = 0.0;
            ajusteXcgObjetos();
            Restricao_Sobreposicao(i);
            } while(restricao == 1);
      }
      for(i=0; i<INDIVIDUOS; i++)
            for(d=0; d<DIM; d++)
                        pop_dim[i][d] = new_pop_dim[i][d];

      calculaInercia();
      encontraMaximos();
      calculaAptidao();
      bubbleSort(aptidao,pop_dim);

      for(i=0; i<INDIVIDUOS; i++)
      {
            if( (pop_parametros[i][10]/1000000 < 0.08 ) &&
(pop_parametros[i][12]/1000000 < 0.25) )
            cont_aptos ++;
      }
      //FINAL DA GERACAO INICIAL
      printf("Geracao;CGr;Ixr;Io;teta;aptidao;tempo;aptos\n");
      printf("%i;",geracao);
      printf("%.6lf;%.6lf;%.6lf;%.3lf;%.6lf;%i\n",pop_parametros[0][9]
/1000000,pop_parametros[0][10]/1000000,pop_parametros[0][11]/1000000,p
op_parametros[0][12],aptidao[0], cont_aptos);

/*********************************************************/
      //Inicia as iteracoes (GERACOES) do AG

      double expoente,delta;
      int taxa_mutacao;
      //printf("Geracao = \n");
      do
      {
            crossover=0; mutacoes=0; cont_rest=0;     cont_aptos= 0;
            //Armazenar os melhores da geraçao atual no grupo de elite
            for(i=0; i<qtde_elite; i++)
            {
                  for(d=0; d<DIM; d++)
                        {
                              elite[i][d] = pop_dim[i][d];
                        }
                  elite_aptidao[i]=aptidao[i];
            }
            //Zera vetor cruzamentos para esta geraçao
            for(i=0; i<INDIVIDUOS/2; i++)
```

```
      cruzamentos[i] = initial_cross+1;

expoente = pow((1- (double)geracao/GERACOES),1.5);
delta = pow(2,expoente);
delta = 10+(10*(1 - delta));

taxa_mutacao = 100 - ((int)delta);
geracao++;
printf("%i;",geracao);
double pai1, pai2, filho1, filho2, beta1, beta2;
double rd, rd_mutacao;

//Realiza CROSSOVER para toda populacao
for(int q=0; q<INDIVIDUOS; q=q+2)
{
      crossover++;
      //Seleciona Individuos sem repetir pais
      int flag_cruz=0, res=0;
      do{   //Do While para nao repetir os pais
            flag_cruz = 0;
            //Seleciona Primeiro individuo
            s1 = (int) (aleatorio_gauss()*(INDIVIDUOS-1));
            if(s1 <0)
                  s1 = s1 * (-1);
            //Seleciona Segundo individuo
            s2 = (int) (aleatorio_gauss()*(INDIVIDUOS-1));
            if(s2 <0)
                  s2 = s2 * (-1);
            if(s2 == s1)
                  s2++;
            if (s1>INDIVIDUOS-1)
                  s1 = INDIVIDUOS-2;
            if (s2>INDIVIDUOS-1)
                  s2 = INDIVIDUOS-3;
            /*Unir os inteiros dos filhos pais sorteados:
Obs: Soma-se 1 para nao ter o filho ZERO pois o sistema
            nao concatena filho 0, depois subtrai-se o 1
            para não gerar erro no apontamento da matriz
            */
            res=0;
            res*=pow(10,ceil(log10(s1+2)));
            res+=s1;
            res*=pow(10,ceil(log10(s2+2)));
            res+=s2;

            for(int ss=0; ss<INDIVIDUOS/2; ss++)
            {
                  if(cruzamentos[ss]==res)
                  {     flag_cruz = 1;    }
            }
            if(flag_cruz == 0)
            {     cruzamentos[q/2] = res; }

      } while(flag_cruz == 1);

      // Realiza o Cruzamento
```

```
                        for(d=0; d<DIM; d++)
                        {
                              pai1 = pop_dim[s1][d];
                              pai2 = pop_dim[s2][d];
                              rd =  rand()%2000;
                              beta1 = -0.5 + (rd/1000);
                              rd =  rand()%2000;
                              beta2 = -0.5 + (rd/1000);
                              filho1 = pai1 + beta1*(pai2-pai1);
                              filho2 = pai1 + beta2*(pai2-pai1);
                  //Faz mutaçao com uma probablilidade progressiva
                  //inicial de 0% ate 10% na ultima geraçao
                              rd_mutacao =  rand()%(taxa_mutacao);
                              if(rd_mutacao == 19)
                              {
                                    numero_rand = rand()%RAND_MAX;
                                    filho1 = (((limite_sup-
limite_inf)*numero_rand)/RAND_MAX)+limite_inf;
                                    mutacoes++;
                                    }
                              rd_mutacao =  rand()%(taxa_mutacao);
                              if(rd_mutacao == 36)
                                    {
                                    numero_rand = rand()%RAND_MAX;
                                    filho2 = (((limite_sup-
limite_inf)*numero_rand)/RAND_MAX)+limite_inf;
                                    mutacoes++;
                                    }
                              new_pop_dim[q][d]=filho1;
                              new_pop_dim[q+1][d]=filho2;
                              //Fim da geracao de 2 novos individuos
                        }     //FIM - For para Crossover nas dimensoes
                        //Testa restricoes Filho 1
                        restricao=0;
                        Restricao_Raio(q);
                        if(restricao!= 1)
                              Restricao_Sobreposicao(q);
                        int restricao_f1 = restricao;
                        //Testa restricoes Filho 2
                        restricao=0;
                        Restricao_Raio(q+1);
                        if(restricao!= 1)
                              Restricao_Sobreposicao(q+1);

                        int restricao_f2 = restricao;

                        if(restricao_f1 == 1 || restricao_f2 ==1)
                        {
//Se ocorrer restriçoes refaz a seleçao de pais e cruzamento
                              q = q-2;
                              cruzamentos[q/2] = initial_cross+1;
                        }
            }     // FIM FOR para crossover de toda populacao


      /*********************************************************/
```

```
            //Atualiza Populacao nova populacao-Com Elitismo
            //Move populaçao nova para calculo de parametros
            for(i=0; i<INDIVIDUOS; i++)
                        for(d=0; d<DIM; d++)
                                    pop_dim[i][d] = new_pop_dim[i][d];

            calculaInercia();
            calculaAptidao();
            bubbleSort(aptidao,pop_dim);

//Atualiza populaçao reaproveitando bons indivivuos da elite anterior
            int flag = 0;
            int substitui=INDIVIDUOS-1;
            for(i=0; i<INDIVIDUOS && flag<qtde_elite; i++)
            {
                  if(aptidao[i] > elite_aptidao[flag])
                  {
                        for(d=0; d<DIM; d++)
                              {
                              pop_dim[substitui][d] = elite[flag][d];
                              }
                        aptidao[substitui]=elite_aptidao[flag];
                        flag++;
                        substitui--;
                  }
            }

            calculaInercia();
            calculaAptidao();
            bubbleSort(aptidao,pop_dim);
            for(i=0; i<INDIVIDUOS; i++)
            {
                  if( (pop_parametros[i][10]/1000000 < 0.08 ) &&
(pop_parametros[i][12]/1000000 < 0.25) )
                        cont_aptos ++;
            }
      printf("%.6lf;%.6lf;%.6lf;%.3lf;%.6lf;%i\n",pop_parametros[0][9]
/1000000,pop_parametros[0][10]/1000000,pop_parametros[0][11]/1000000,p
op_parametros[0][12],aptidao[0],cont_aptos);

      } while( (geracao < GERACOES) && (aptidao[0] > aptidao_meta) );
/*************************************************************/

      end_t=clock();
      tempo = (end_t-start_t)/(CLOCKS_PER_SEC);
      printf("Individuos;Objetos;Tempo Total;Geracoes;Semente\n");
      printf("%i;%i;%.3lf;%i;%i\n",INDIVIDUOS,OBJETO,tempo,GERACOES,se
mente);
      printf("CGx = ;%.1lf\n",5234-pop_parametros[0][2]);

/********** Imprimir sequencia cilindro x,y\n raio\n  *************/
      alturaPratoCAD();
      /*
      for(i=0; i<1;i++)
      {
            int o=0;
```

```
            for(d=0; d<DIM; d=d+2)
            {

      printf("cilindro\n%.0lf,%.0lf,%.0lf\n%.0lf\n%.0lf\n",pop_dim[i][
d],pop_dim[i][d+1],objetos[o][3],objetos[o][0],objetos[o][1]);

                  o++;
            }
      }
      printf("\n");
      */
      //Libera o espaco reservado na memoria
      printf("Esvaziando MemÃ³ria\n");
      free(pop_dim);
      free(new_pop_dim);
      free(pop_parametros);
      free(elite);
      free(aptidao);
      free(new_aptidao);
      //free(elite_aptidao);
      //free(cruzamentos);

      printf("FIM\n");
}

float fuzzy(float cg, float in, float teta_t)
{
for(int i=0; i<qtde_regras; i++)
      {
      ativas_cgfuz[i] = 0; ativas_ifuz[i] = 0; ativas_tfuz[i] = 0;
      }
      cg_o=0.0; cg_r=0.0; cg_p=0.0;
      i_o=0.0; i_r=0.0; i_p=0.0;
      t_o=0.0; t_r=0.0; t_p=0.0;

      // AVALIANDO REGRAS ATIVADAS NO CG
      e = cg;
      if(e >= 0.0 && e <= 0.1)
            {
            ativas_cgfuz[0]=1; ativas_cgfuz[1]=1;ativas_cgfuz[2]=1;
            ativas_cgfuz[3]=1;ativas_cgfuz[4]=1;ativas_cgfuz[5]=1;
            ativas_cgfuz[6]=1;ativas_cgfuz[7]=1;ativas_cgfuz[8]=1;
            cg_o = (0.4-e)/0.4;
            }
      if(e > 0.1 && e < 0.4 )
            {
            ativas_cgfuz[0]=1; ativas_cgfuz[1]=1;ativas_cgfuz[2]=1;
            ativas_cgfuz[3]=1;ativas_cgfuz[4]=1;ativas_cgfuz[5]=1;
            ativas_cgfuz[6]=1;ativas_cgfuz[7]=1;ativas_cgfuz[8]=1;
            ativas_cgfuz[9]=1; ativas_cgfuz[10]=1;ativas_cgfuz[11]=1;
            ativas_cgfuz[12]=1; ativas_cgfuz[13]=1;ativas_cgfuz[14]=1;
            ativas_cgfuz[15]=1;ativas_cgfuz[16]=1;ativas_cgfuz[17]=1;
            cg_o = (0.4-e)/0.4;
            cg_r = (e-0.1)/0.4;
            }
      if(e >= 0.4 && e < 0.6 )
```

```
      {
      ativas_cgfuz[9]=1;ativas_cgfuz[10]=1;ativas_cgfuz[11]=1;
      ativas_cgfuz[12]=1; ativas_cgfuz[13]=1;ativas_cgfuz[14]=1;
      ativas_cgfuz[15]=1; ativas_cgfuz[16]=1;ativas_cgfuz[17]=1;
      cg_r = (e-0.1)/0.4;
      }
if(e >= 0.6 && e < 0.9 )
      {
      ativas_cgfuz[9]=1; ativas_cgfuz[10]=1;ativas_cgfuz[11]=1;
      ativas_cgfuz[12]=1; ativas_cgfuz[13]=1;ativas_cgfuz[14]=1;
      ativas_cgfuz[15]=1;ativas_cgfuz[16]=1;ativas_cgfuz[17]=1;
      ativas_cgfuz[18]=1; ativas_cgfuz[19]=1;ativas_cgfuz[20]=1;
      ativas_cgfuz[21]=1; ativas_cgfuz[22]=1;ativas_cgfuz[23]=1;
      ativas_cgfuz[24]=1; ativas_cgfuz[25]=1;ativas_cgfuz[26]=1;
      cg_r = (0.9-e)/0.4;
      cg_p = (e-0.6)/0.4;
      }
if(e >= 0.9 )
      {
      ativas_cgfuz[18]=1;ativas_cgfuz[19]=1;ativas_cgfuz[20]=1;
      ativas_cgfuz[21]=1; ativas_cgfuz[22]=1;ativas_cgfuz[23]=1;
      ativas_cgfuz[24]=1; ativas_cgfuz[25]=1;ativas_cgfuz[26]=1;
      cg_p = (e-0.6)/0.4;
      }

// Verifica regras ativadas pelo Produto de Inercia
e = in;
if(e >= 0.0 && e <= 0.1)
      {
      ativas_ifuz[0]=1; ativas_ifuz[1]=1;ativas_ifuz[2]=1;
      ativas_ifuz[9]=1; ativas_ifuz[10]=1; ativas_ifuz[11]=1;
      ativas_ifuz[18]=1; ativas_ifuz[19]=1;ativas_ifuz[20]=1;
      i_o = (0.4-e)/0.4;
      }
if(e > 0.1 && e < 0.4 )
      {
      ativas_ifuz[0]=1; ativas_ifuz[1]=1;ativas_ifuz[2]=1;
      ativas_ifuz[9]=1; ativas_ifuz[10]=1; ativas_ifuz[11]=1;
      ativas_ifuz[18]=1; ativas_ifuz[19]=1;ativas_ifuz[20]=1;
      ativas_ifuz[3]=1; ativas_ifuz[4]=1;ativas_ifuz[5]=1;
      ativas_ifuz[12]=1; ativas_ifuz[13]=1;ativas_ifuz[14]=1;
      ativas_ifuz[21]=1; ativas_ifuz[22]=1;ativas_ifuz[23]=1;
      i_o = (0.4-e)/0.4;
      i_r = (e-0.1)/0.4;
      }
if(e >= 0.4 && e < 0.6 )
      {
      ativas_ifuz[3]=1; ativas_ifuz[4]=1;ativas_ifuz[5]=1;
      ativas_ifuz[12]=1; ativas_ifuz[13]=1;ativas_ifuz[14]=1;
      ativas_ifuz[21]=1; ativas_ifuz[22]=1;ativas_ifuz[23]=1;
      i_r = (e-0.1)/0.4;
      }
if(e >= 0.6 && e < 0.9 )
      {
      ativas_ifuz[3]=1; ativas_ifuz[4]=1;ativas_ifuz[5]=1;
      ativas_ifuz[12]=1; ativas_ifuz[13]=1;ativas_ifuz[14]=1;
```

```
            ativas_ifuz[21]=1; ativas_ifuz[22]=1;ativas_ifuz[23]=1;
            ativas_ifuz[6]=1; ativas_ifuz[7]=1; ativas_ifuz[8]=1;
            ativas_ifuz[15]=1; ativas_ifuz[16]=1;ativas_ifuz[17]=1;
            ativas_ifuz[24]=1; ativas_ifuz[25]=1;ativas_ifuz[26]=1;
            i_r = (0.9-e)/0.4;
            i_p = (e-0.6)/0.4;
            }
      if(e >= 0.9 )
            {
            ativas_ifuz[6]=1; ativas_ifuz[7]=1; ativas_ifuz[8]=1;
            ativas_ifuz[15]=1; ativas_ifuz[16]=1;ativas_ifuz[17]=1;
            ativas_ifuz[24]=1; ativas_ifuz[25]=1;ativas_ifuz[26]=1;
            i_p = (e-0.6)/0.4;
            }
// Verifica regras ativadas pelo Ã‚ngulo do Eixo do foguete
      e = teta_f;
      if(e >= 0.0 && e <= 0.1)
            {
            ativas_tfuz[0]=1; ativas_tfuz[3]=1; ativas_tfuz[6]=1;
            ativas_tfuz[9]=1; ativas_tfuz[12]=1;ativas_tfuz[15]=1;
            ativas_tfuz[18]=1;ativas_tfuz[21]=1;ativas_tfuz[24]=1;
            t_o = (0.4-e)/0.4;
            }
      if(e > 0.1 && e < 0.4 )
            {
            ativas_tfuz[0]=1; ativas_tfuz[3]=1; ativas_tfuz[6]=1;
            ativas_tfuz[9]=1; ativas_tfuz[12]=1;ativas_tfuz[15]=1;
            ativas_tfuz[18]=1;ativas_tfuz[21]=1;ativas_tfuz[24]=1;
            ativas_tfuz[1]=1; ativas_tfuz[4]=1; ativas_tfuz[7]=1;
            ativas_tfuz[10]=1;ativas_tfuz[13]=1;ativas_tfuz[16]=1;
            ativas_tfuz[19]=1;ativas_tfuz[22]=1;ativas_tfuz[25]=1;
            t_o = (0.4-e)/0.4;
            t_r = (e-0.1)/0.4;
            }
      if(e >= 0.4 && e < 0.6 )
            {
            ativas_tfuz[1]=1; ativas_tfuz[4]=1; ativas_tfuz[7]=1;
            ativas_tfuz[10]=1;ativas_tfuz[13]=1;ativas_tfuz[16]=1;
            ativas_tfuz[19]=1;ativas_tfuz[22]=1;ativas_tfuz[25]=1;
            t_r = (e-0.1)/0.4;
            }
      if(e >= 0.6 && e < 0.9 )
            {
            ativas_tfuz[1]=1; ativas_tfuz[4]=1; ativas_tfuz[7]=1;
            ativas_tfuz[10]=1;ativas_tfuz[13]=1;ativas_tfuz[16]=1;
            ativas_tfuz[19]=1;ativas_tfuz[22]=1;ativas_tfuz[25]=1;
            ativas_tfuz[2]=1; ativas_tfuz[5]=1; ativas_tfuz[8]=1;
            ativas_tfuz[11]=1;ativas_tfuz[14]=1;ativas_tfuz[17]=1;
            ativas_tfuz[20]=1;ativas_tfuz[23]=1;ativas_tfuz[26]=1;
            t_r = (0.9-e)/0.4;
            t_p = (e-0.6)/0.4;
            }
      if(e >= 0.9 )
            {
            ativas_tfuz[2]=1; ativas_tfuz[5]=1; ativas_tfuz[8]=1;
            ativas_tfuz[11]=1;ativas_tfuz[14]=1;ativas_tfuz[17]=1;
```

```
            ativas_tfuz[20]=1;ativas_tfuz[23]=1;ativas_tfuz[26]=1;
            t_p = (e-0.6)/0.4;
            }

      for(int i=0; i<qtde_regras; i++)
      {
            regras_ativas[i] = ativas_cgfuz[i] * ativas_ifuz[i] *
      ativas_tfuz[i];
      }

      // CALCULO DO GRAU DE PERTENCIMENTO DE CADA REGRA
      int cont_regras =0;
      for(int i=0; i<qtde_regras; i++)
      {
            if(regras_ativas[i] == 1)
            {
                  int aplica = i;
                  switch (aplica)
                  {
                        case 0:
                              DOF[cont_regras] = cg_o;
                              if( i_o < DOF[cont_regras])
                                    DOF[cont_regras] = i_o;
                              if ( t_o < DOF[cont_regras])
                                    DOF[cont_regras] = t_o;
                              u[cont_regras] = ((-0.4 *
DOF[cont_regras]) + 0.4) * 0.0; //O
                              break;

                        case 1:
                        DOF[cont_regras] = cg_o;
                              if( i_o < DOF[cont_regras])
                                    DOF[cont_regras] = i_o;
                              if ( t_r < DOF[cont_regras])
                                    DOF[cont_regras] = t_r;

                              if(DOF[cont_regras] <= 0.5)
                                    u[cont_regras] = (DOF[cont_regras]-
0.1)/0.4;
                              else
                                    u[cont_regras] = (0.9-
DOF[cont_regras])/0.4;
                              u[cont_regras] = u[cont_regras]* 0.5; //R

                              break;

                        case 2:
                              DOF[cont_regras] = cg_o;
                              if( i_o < DOF[cont_regras])
                                    DOF[cont_regras] = i_o;
                              if ( t_p < DOF[cont_regras])
                                    DOF[cont_regras] = t_p;
                              u[cont_regras] = ((0.4 *
DOF[cont_regras]) + 0.6) * 1;
                        break;
```

```
                        case 3:
                              DOF[cont_regras] = cg_o;
                              if( i_r < DOF[cont_regras])
                                    DOF[cont_regras] = i_r;
                              if ( t_o < DOF[cont_regras])
                                    DOF[cont_regras] = t_o;
                              u[cont_regras] = ((-0.4 *
DOF[cont_regras]) + 0.4) * 0.0; //O
                              break;

                        case 4:
                              DOF[cont_regras] = cg_o;
                              if( i_r < DOF[cont_regras])
                                    DOF[cont_regras] = i_r;
                              if ( t_r < DOF[cont_regras])
                                    DOF[cont_regras] = t_r;

                              if(DOF[cont_regras] <= 0.5)
                                    u[cont_regras] = (DOF[cont_regras]-
0.1)/0.4;
                              else
                                    u[cont_regras] = (0.9-
DOF[cont_regras])/0.4;
                              u[cont_regras] = u[cont_regras]* 0.5; //R

                              break;

                        case 5:
                              DOF[cont_regras] = cg_o;
                              if( i_r < DOF[cont_regras])
                                    DOF[cont_regras] = i_r;
                              if ( t_p < DOF[cont_regras])
                                    DOF[cont_regras] = t_p;
                              u[cont_regras] = ((0.4 *
DOF[cont_regras]) + 0.6) * 1; //P
                              break;

                        case 6:
                              DOF[cont_regras] = cg_o;
                              if( i_p < DOF[cont_regras])
                                    DOF[cont_regras] = i_p;
                              if ( t_o < DOF[cont_regras])
                                    DOF[cont_regras] = t_o;
                              u[cont_regras] = ((0.4 *
DOF[cont_regras]) + 0.6) * 1; //P
                              break;

                        case 7:
                              DOF[cont_regras] = cg_o;
                              if( i_p < DOF[cont_regras])
                                    DOF[cont_regras] = i_p;
                              if ( t_r < DOF[cont_regras])
                                    DOF[cont_regras] = t_r;
                              u[cont_regras] = ((0.4 *
DOF[cont_regras]) + 0.6) * 1; //P
                              break;
```

```
                        case 8:
                              DOF[cont_regras] = cg_o;
                              if( i_p < DOF[cont_regras])
                                    DOF[cont_regras] = i_p;
                              if ( t_p < DOF[cont_regras])
                                    DOF[cont_regras] = t_p;
                              u[cont_regras] = ((0.4 *
DOF[cont_regras]) + 0.6) * 1; //P
                              break;

                        case 9:
                              DOF[cont_regras] = cg_r;
                              if( i_o < DOF[cont_regras])
                                    DOF[cont_regras] = i_o;
                              if ( t_o < DOF[cont_regras])
                                    DOF[cont_regras] = t_o;
                              u[cont_regras] = ((-0.4 *
DOF[cont_regras]) + 0.4) * 0.0; //O
                              break;

                        case 10:
                              DOF[cont_regras] = cg_r;
                              if( i_o < DOF[cont_regras])
                                    DOF[cont_regras] = i_o;
                              if ( t_r < DOF[cont_regras])
                                    DOF[cont_regras] = t_r;

                              if(DOF[cont_regras] <= 0.5)
                                    u[cont_regras] = (DOF[cont_regras]-
0.1)/0.4;
                              else
                                    u[cont_regras] = (0.9-
DOF[cont_regras])/0.4;
                              u[cont_regras] = u[cont_regras]* 0.5; //R


                        case 11:
                              DOF[cont_regras] = cg_r;
                              if( i_o < DOF[cont_regras])
                                    DOF[cont_regras] = i_o;
                              if ( t_p < DOF[cont_regras])
                                    DOF[cont_regras] = t_p;
                              u[cont_regras] = ((0.4 *
DOF[cont_regras]) + 0.6) * 1; //P
                              break;


                        case 12:
                              DOF[cont_regras] = cg_r;
                              if( i_r < DOF[cont_regras])
                                    DOF[cont_regras] = i_r;
                              if ( t_o < DOF[cont_regras])
                                    DOF[cont_regras] = t_o;
                              u[cont_regras] = ((-0.4 *
DOF[cont_regras]) + 0.4) * 0.0; //O
```

```
                        break;

                    case 13:
                        DOF[cont_regras] = cg_r;
                        if( i_r < DOF[cont_regras])
                            DOF[cont_regras] = i_r;
                        if ( t_r < DOF[cont_regras])
                            DOF[cont_regras] = t_r;

                        if(DOF[cont_regras] <= 0.5)
                            u[cont_regras] = (DOF[cont_regras]-
0.1)/0.4;
                        else
                            u[cont_regras] = (0.9-
DOF[cont_regras])/0.4;
                        u[cont_regras] = u[cont_regras]* 0.5; //R


                    case 14:
                        DOF[cont_regras] = cg_r;
                        if( i_r < DOF[cont_regras])
                            DOF[cont_regras] = i_r;
                        if ( t_p < DOF[cont_regras])
                            DOF[cont_regras] = t_p;
                        u[cont_regras] = ((0.4 *
DOF[cont_regras]) + 0.6) * 1; //P
                        break;

                    case 15:
                        DOF[cont_regras] = cg_r;
                        if( i_p < DOF[cont_regras])
                            DOF[cont_regras] = i_p;
                        if ( t_o < DOF[cont_regras])
                            DOF[cont_regras] = t_o;
                        u[cont_regras] = ((0.4 *
DOF[cont_regras]) + 0.6) * 1; //P
                        break;

                    case 16:
                        DOF[cont_regras] = cg_r;
                        if( i_p < DOF[cont_regras])
                            DOF[cont_regras] = i_p;
                        if ( t_r < DOF[cont_regras])
                            DOF[cont_regras] = t_r;
                        u[cont_regras] = ((0.4 *
DOF[cont_regras]) + 0.6) * 1; //P
                        break;

                    case 17:
                        DOF[cont_regras] = cg_r;
                        if( i_p < DOF[cont_regras])
                            DOF[cont_regras] = i_p;
                        if ( t_p < DOF[cont_regras])
                            DOF[cont_regras] = t_p;
                        u[cont_regras] = ((0.4 *
DOF[cont_regras]) + 0.6) * 1; //P
```

```
                        break;

                    case 18:
                        DOF[cont_regras] = cg_p;
                        if( i_o < DOF[cont_regras])
                            DOF[cont_regras] = i_o;
                        if ( t_o < DOF[cont_regras])
                            DOF[cont_regras] = t_o;

                        if(DOF[cont_regras] <= 0.5)
                            u[cont_regras] = (DOF[cont_regras]-
0.1)/0.4;
                        else
                            u[cont_regras] = (0.9-
DOF[cont_regras])/0.4;
                        u[cont_regras] = u[cont_regras]* 0.5; //R

                    case 19:
                        DOF[cont_regras] = cg_p;
                        if( i_o < DOF[cont_regras])
                            DOF[cont_regras] = i_o;
                        if ( t_r < DOF[cont_regras])
                            DOF[cont_regras] = t_r;
                        u[cont_regras] = ((0.4 *
DOF[cont_regras]) + 0.6) * 1; //P
                        break;

                    case 20:
                        DOF[cont_regras] = cg_p;
                        if( i_o < DOF[cont_regras])
                            DOF[cont_regras] = i_o;
                        if ( t_p < DOF[cont_regras])
                            DOF[cont_regras] = t_p;
                        u[cont_regras] = ((0.4 *
DOF[cont_regras]) + 0.6) * 1; //P
                        break;

                    case 21:
                        DOF[cont_regras] = cg_p;
                        if( i_r < DOF[cont_regras])
                            DOF[cont_regras] = i_r;
                        if ( t_o < DOF[cont_regras])
                            DOF[cont_regras] = t_o;

                        if(DOF[cont_regras] <= 0.5)
                            u[cont_regras] = (DOF[cont_regras]-
0.1)/0.4;
                        else
                            u[cont_regras] = (0.9-
DOF[cont_regras])/0.4;
                        u[cont_regras] = u[cont_regras]* 0.5; //R

                    case 22:
                        DOF[cont_regras] = cg_p;
                        if( i_r < DOF[cont_regras])
                            DOF[cont_regras] = i_r;
```

```
                              if ( t_r < DOF[cont_regras])
                                    DOF[cont_regras] = t_r;
                              u[cont_regras] = ((0.4 *
DOF[cont_regras]) + 0.6) * 1; //P
                        break;

                        case 23:
                              DOF[cont_regras] = cg_p;
                              if( i_r < DOF[cont_regras])
                                    DOF[cont_regras] = i_r;
                              if ( t_p < DOF[cont_regras])
                                    DOF[cont_regras] = t_p;
                              u[cont_regras] = ((0.4 *
DOF[cont_regras]) + 0.6) * 1; //P
                        break;

                        case 24:
                              DOF[cont_regras] = cg_p;
                              if( i_p < DOF[cont_regras])
                                    DOF[cont_regras] = i_p;
                              if ( t_o < DOF[cont_regras])
                                    DOF[cont_regras] = t_o;
                              u[cont_regras] = ((0.4 *
DOF[cont_regras]) + 0.6) * 1; //P
                              break;

                        case 25:
                              DOF[cont_regras] = cg_p;
                              if( i_p < DOF[cont_regras])
                                    DOF[cont_regras] = i_p;
                              if ( t_r < DOF[cont_regras])
                                    DOF[cont_regras] = t_r;
                              u[cont_regras] = ((0.4 *
DOF[cont_regras]) + 0.6) * 1; //P
                              break;

                        case 26:
                              DOF[cont_regras] = cg_p;
                              if( i_p < DOF[cont_regras])
                                    DOF[cont_regras] = i_p;
                              if ( t_p < DOF[cont_regras])
                                    DOF[cont_regras] = t_p;
                              u[cont_regras] = ((0.4 *
DOF[cont_regras]) + 0.6) * 1; //P
                              break;

                        default:
                              printf("");
                  }
                  cont_regras++;
            }
      }

      // CALCULO DA SAIDA COS
      float numerador=0.0, denominador=0.0;
      for(int i=0; i<cont_regras ; i++)
```

```
	{
		numerador = numerador + (u[i]*DOF[i]);
		denominador = denominador + DOF[i];
	}
	saida = numerador/denominador;
	return(saida);
}
```