OTIMIZAÇÃO DE ESTRUTURAS RETICULADAS UTILIZANDO ALGORITMOS GENÉTICOS

DIGNA ISABEL ARTEAGA VÉLEZ

DISSERTAÇÃO DE MESTRADO EM ESTRUTURAS E CONSTRUÇÃO CIVIL

DEPARTAMENTO DE ENGENHARIA CIVIL E AMBIENTAL

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA CIVIL E AMBIENTAL

OTIMIZAÇÃO DE ESTRUTURAS RETICULADAS UTILIZANDO ALGORITMOS GENÉTICOS

DIGNA ISABEL ARTEAGA VÉLEZ

ORIENTADOR: RAÚL DARÍO DURAND FARFÁN

DISSERTAÇÃO DE MESTRADO EM ESTRUTURAS E CONSTRUÇÃO CIVIL

PUBLICAÇÃO: E.DM-006A/15
BRASÍLIA/DF: ABRIL – 2015

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA CIVIL E AMBIENTAL

OTIMIZAÇÃO DE ESTRUTURAS RETICULADAS UTILIZANDO ALGORITMOS GENÉTICOS

DIGNA ISABEL ARTEAGA VÉLEZ

DISSERTAÇÃO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA CIVIL E AMBIENTAL DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM ESTRUTURAS E CONSTRUÇÃO CIVIL.

APROVADA POR:

_____
Prof. Raúl Darío Durand Farfán, D.Sc. (UnB)
(Orientador)

_____
Prof. Artur Portela, Ph. D. (UnB)
(Examinador Interno)

_____
Prof. Márcio Muniz de Farias, Ph. D. (UnB)
(Examinador Externo)

BRASÍLIA/DF, 20 DE ABRIL DE 2015.

FICHA CATALOGRÁFICA

ARTEAGA, DIGNA ISABEL.
Otimização de Estruturas Reticuladas Utilizando Algoritmos Genéticos. [Distrito Federal] 2015.

xvii, 97 p., 297mm (ENC/FT/UnB, Mestre, Estruturas e Construção Civil, 2015).
Dissertação de Mestrado – Universidade de Brasília. Faculdade de Tecnologia.

Departamento de Engenharia Civil e Ambiental.

1. Otimização Estrutural	2. Elementos Finitos
3. Algoritmos Genéticos	4. Estruturas Reticuladas
I. ENC/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

ARTEAGA, D. I.; Otimização de Estruturas Reticuladas Utilizando Algoritmos Genéticos. Dissertação de Mestrado, Publicação E.DM-006A/15, Departamento de Engenharia Civil e Ambiental. Universidade de Brasília. Brasília, DF, 97p.

CESSÃO DE DIREITOS

AUTOR: Digna Isabel Arteaga Vélez
TÍTULO: Otimização de Estruturas Reticuladas Utilizando Algoritmos Genéticos.
GRAU: Mestre ANO: 2015

Digna Isabel Arteaga Vélez
Calle 3 # 10-01. Barrio Centenario.
Ipiales–Colombia.
e-mail: isabelagaetra@gmail.com

En memoria de mi padre Alberto Arteaga.
Siempre estarás en mi mente y mi corazón.

Você não pode mudar o vento,
mas pode ajustar as velas do barco
para chegar onde quer.
Confúcio

AGRADECIMENTOS

Agradeço principalmente a Deus por mais uma conquista. A minha família, em especial a minha mãe Margarita, a meus irmãos Claudia, Martha e Carlos Alberto, a meus sobrinhos Sofia, Angela e Juan David, por serem minha motivação e por sempre me apoiarem incondicionalmente, para eles minha mais profunda e eterna gratidão.

Ao meu orientador, Raul Durand pelo acompanhamento, pelas ideias e pelo tempo dedicado ao longo da pesquisa.

A meus amigos, Sergio e Jairo, pelos risos, pelo apoio incondicional e por fazerem minha estadia no Brasil mais amena. Obrigada pela amizade e carinho de todos que me ajudaram nos momentos que mais precisei Johnny, Jader, Cristina, Amarillo, Janeth, David, Jaime, Giovanni, Maria Paula.

Ao PECC pela oportunidade e aprendizagem ao longo destes anos. A todos os companheiros do programa que me proporcionaram um ambiente de estudo e pesquisa agradável. Wilber, Damaris, Juan David, Pablo e Carmen. E a meus companheiros de sala Nelson, Marcus, Carlos, Vitor, Jéssica e Uchôa.

A Capes pelo apoio financeiro ao longo da pesquisa.

OTIMIZAÇÃO DE ESTRUTURAS RETICULADAS UTILIZANDO ALGORITMOS GENÉTICOS

Autor: Digna Isabel Arteaga Vélez
Orientador: Raul Durand
Programa de Pós-graduação em Estruturas e Construção Civil
Brasília, abril de 2015

RESUMO

Esta pesquisa apresenta um procedimento e uma aplicação de software para otimizar peso e deslocamento de estruturas reticuladas, por meio da mudança das áreas das seções transversais e da forma de treliças planas e espaciais. Para tanto, foi utilizada a combinação do Algoritmo Genético como método de busca heurística conjuntamente com Elementos Finitos lineares tipo barra para avaliação estática. Neste processo, foram considerados restrições de tensão e deslocamentos nodais. Através de processos cíclicos do Algoritmo Genético e utilizando os operadores genéticos probabilísticos de seleção, cruzamento, mutação e elitismo, determinou-se uma família de possíveis soluções que ao longo das gerações levou a uma solução ótima. Este trabalho utilizou como ferramenta de programação as linguagens *Python* e *Julia*. Para a validação da metodologia, foram utilizados exemplos de otimização de treliças bidimensionais e tridimensionais submetidas a carregamento estático e sujeitas a restrições de tensão e deslocamentos. Os resultados são comparados com os obtidos por outros autores. Esses resultados demonstram que a metodologia implementada permite a obtenção de estruturas que satisfazem às condições inicialmente impostas com uma evidente redução de peso e deslocamento.

OPTIMIZATION OF FRAME STRUCTURES USING GENETIC ALGORITHMS

Author: Digna Isabel Arteaga Vélez
Supervisor: Raul Durand
Programa de Pós-graduação em Estruturas e Construção Civil
Brasília, April of 2015

ABSTRACT

This research presents a procedure and a software application to optimize the weight and displacements of frame structures by changing the cross section areas and the shape of plane and spatial trusses. Therefore, the combination of genetic algorithms as a heuristic search method and linear finite elements was used. In this process, restrictions such as maximum stresses and nodal displacement were considered. Through the application of genetic algorithms and the use of probabilistic genetic operators such as selection, crossover, mutation and elitism, it was possible to find a family of appropriate solutions and, along several generations, to find an optimized solution. In this study, Python and Julia languages were used as programming tool. In order to validate the methodology, two and three-dimensional trusses subject to static loads together with stress and displacements constraints were analyzed. The results are compared with those obtained by other authors. These results demonstrate that the proposed methodology is able to provide structures that satisfy imposed initial conditions with an evident reduction in weight and displacements.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS, NOMENCLATURA E ABREVIAÇÕES

α, β	Coeficientes utilizados na otimização multiobjetivo
γ	Peso específico
δ	Deslocamento
$\delta_{máx}$	Deslocamento máximo
ε	Deformação
λ_1	Fator de alteração
μ	Número de indivíduos da população
ξ	Coordenadas locais
σ^2	Variância
$\Delta\sigma$	Acréscimo de tensão axial
F	Vetor de forças
A	Área
B	Matriz da relação deformação-deslocamento
cr_A	Cromossomo utilizado para otimização dimensional
cr_n	Cromossomo utilizado para otimização de forma
D	Escalar constitutivo
E	Modulo de Young
f	Função objetivo
g	Restrições comportamentais
J	Matriz Jacobiana
K	Matriz de rigidez
l	Número de fenótipos
L	Comprimento da barra
n_f	Posição de corte do cromossomo
n_g	Comprimento do cromossomo
n_{pop}	Tamanho da população
N	Função de forma
p_m	Probabilidade de mutação
p_s	Probabilidade de seleção

r	Número aleatório
$r_1,\ r_2\ , r_3$	Componentes da direção de um elemento de barra
u	Vetor de deslocamento nodal
V	Volume do elemento de barra
W	Peso total
W_l	Peso limite
x_i	Variável de projeto
x	Vetor de variáveis de projeto

1 INTRODUÇÃO

A otimização estrutural é um processo numérico/matemático que proporciona uma melhor configuração da estrutura como uma composição ótima em desempenho e forma, por exemplo, uma estrutura com menor peso, menor flambagem local ou global, menor tensão, máxima rigidez (Silva, 2011). De acordo com Pizzirani (2003), a área de otimização estrutural está dividida em três categorias: otimização dimensional, otimização de forma e a otimização topológica. A otimização dimensional busca uma melhor distribuição das áreas de seção visando a minimização ou maximização da função objetivo. A otimização de forma pretende encontrar o domínio espacial ótimo do problema, este domínio é variável, pois o objetivo é encontrar a melhor forma da estrutura para atender uma determinada solicitação. Por sua vez, na otimização topológica a variável do projeto está associada a distribuição do material. Por exemplo, no caso de problemas discretos (treliças) alguns dos elementos que compõem a estrutura podem ser subtraídos da composição. Em geral, para a aplicação dos três tipos de otimização existem vários métodos, desde os mais antigos baseados em soluções analíticas, passando pelos métodos iterativos e os mais recentes métodos de otimização evolutivos como são os algoritmos genéticos.

O algoritmo genético (AG) é um método populacional de pesquisa dirigida baseada em probabilidade. Assim como as técnicas heurísticas, os AGs têm alcançado grande popularidade, pelo fato de resolver problemas que são considerados complexos na aplicação de procedimentos matemáticos tradicionais. Atualmente os AGs são adotados frequentemente como métodos para simular a evolução natural em busca de soluções ótimas. Dessa forma são usados, com êxito, na solução de problemas de otimização combinatória, otimização de funções reais e, também, em mecanismos de aprendizado de máquina (Kuri e Galaviz, 2002).

Neste trabalho, pretende-se abordar a otimização de estruturas reticuladas do tipo treliça por meio da minimização das variáveis como: área das seções transversais, peso e comprimento de elementos. Para tanto, utilizou-se o Algoritmo Genético como método de busca heurística combinado com o Método dos Elementos Finitos (elementos tipo barra) para avaliação estática. Nesse processo, foram consideradas restrições de tensões e deslocamentos nodais.

Por meio de processos cíclicos do Algoritmo Genético juntamente com operadores genéticos probabilísticos de seleção, cruzamento, mutação e elitismo foi possível determinar uma família de possíveis soluções que ao longo das gerações levou, em tese, à melhor solução. Para isso, utilizou-se como ferramenta de programação as linguagens *Python* e *Julia* e as bibliotecas *Pyfem* e *FEMlab*.

1.1 OBJETIVOS

Este trabalho aplica AGs conjuntamente com o MEF para realizar otimizações dimensionais e de forma em estruturas do tipo treliça. Os objetivos deste trabalho são:

- Estudar parâmetros e tipos de seleção próprios dos AGs com a finalidade de encontrar aqueles que melhorem o processo de otimização de peso e deslocamento em estruturas reticuladas.

- Determinar critérios que facilitem os processos de otimização e que poderão constituir uma metodologia de análise para o uso de Algoritmos Genéticos na otimização estrutural.

- Introduzir critérios que facilitem o processo de convergência via AG.

Para atingir estes objetivos, foi necessária a realização das seguintes etapas:

- Estudo do Método de Elementos Finitos para o cálculo de forças internas em treliças conjuntamente com o estudo da técnica de otimização através Algoritmos Genéticos.

- Estudo de otimização dimensional de estruturas do tipo treliça através da variação dos parâmetros dos AGs, tais como, taxa de mutação, taxa de elitismo e fator de mutação.

- Estudo de otimização de forma de estruturas do tipo treliça utilizando dois tipos de seleção de indivíduos (treliças) no contexto dos AGs.

- Estudo da definição da aptidão de indivíduos (treliças) de forma a realizar otimizações multi-objetivo.

- Aplicação da metodologia proposta em casos amplamente estudados na literatura com a finalidade de verificar a qualidade dos resultados.

1.2 ESTRUTURA DA DISSERTAÇÃO

A dissertação é composta de 7 capítulos. Para fornecer uma visão geral do trabalho, a seguir, é apresentado um breve resumo de cada capítulo.

O primeiro capítulo apresenta a introdução, destacando o contexto do tema abordado. O capítulo dois apresenta uma revisão bibliográfica da formulação geral do processo de otimização estrutural. Contém a classificação dos métodos de otimização, assim como, os conceitos fundamentais das variáveis do projeto, restrições e função objetivo.

O capítulo três contém as origens e fundamentos teóricos sobre os algoritmos genéticos, sua classificação em relação a outros algoritmos evolucionários de otimização existentes. Ainda neste capitulo é mostrado alguns tópicos sobre algoritmos genéticos, como os parâmetros de configuração e seus operadores genéticos, além disso, são apresentadas as vantagens e desvantagens do uso deste método.

No capítulo quatro encontra-se a formulação generalizada para um elemento de barra utilizando o Método dos Elementos Finitos. Ainda neste capítulo são apresentadas aplicações do uso dos AGs e o MEF.

O capítulo cinco mostra a metodologia e os parâmetros utilizados no processo de otimização dimensional e de forma em treliças bidimensionais (2D) e tridimensionais (3D).

No capítulo seis é apresentada a análise de alguns casos clássicos de otimização estrutural em treliças 2D e 3D, com inclusão de variáveis contínuas. São utilizados, para alguns casos, restrições de tensões e deslocamento máximo. Os resultados são comparados aos da literatura.

Finalmente, no capítulo sete, apresentam-se as conclusões deste trabalho, as sugestões para futuros trabalhos em otimização estrutural através dos Algoritmos Genéticos.

2 OTIMIZAÇÃO ESTRUTURAL

O tema otimização estrutural é uma fusão das áreas de Engenharia, Matemática, Ciências e Tecnologia que tem como objetivo a obtenção do projeto (estrutura) com melhor desempenho. Esta área é muito ampla e tem muitos autores que fazem referência neste campo. Os primeiros registros datam dos anos de 1638 quando Galileu utilizou o conceito de otimização para melhorar a forma de uma estrutura baseada em sua resistência.

De acordo com Silva (2011), o cientista Maxwell, no final do século XVIII, utilizou a otimização estrutural com o objetivo de diminuir o uso de material na construção de pontes que suportassem as necessidades de uso. Depois de vários estudos, Maxwell sugeriu que a forma conceitual de uma estrutura ótima, que utilizasse menos material possível, seria constituída de elementos de treliça. Posteriormente, Michell decidiu aplicar essa teoria para o projeto de vários tipos de estruturas, visando utilizar o menor volume de material. Esses estudos naquela época foram considerados muito teóricos e sem aplicação prática. Somente mais tarde com o método dos elementos finitos e o surgimento dos computadores é que problemas práticos começaram a ser estudados. Já na década dos 70, algoritmos de otimização topológica foram desenvolvidos e na década seguinte, com a ajuda de softwares, os resultados de Michell passaram a ser aplicados na Engenharia Civil.

Vanderplaats (1993) faz uma introdução sobre otimização estrutural e apresenta o estado da arte. O autor referido descreve alguns exemplos como resultado do desenvolvimento dos diferentes métodos utilizados ao longo dos anos e a utilização da tecnologia como uma fonte prática para tal fim.

Segundo Lemonge (1999), a otimização estrutural busca desvincular a escolha da melhor solução da experiência do projetista, incorporando novos critérios que permitem uma avaliação matemática de qualidade da solução. Os critérios que ele menciona são, por exemplo: escolha do material, melhor combinação das peças estruturais disponíveis no mercado, topologia, durabilidade, confiabilidade, funcionalidade, eficiência e tempo de execução, recursos para análises, dentre outros.

Segundo Melchers e Hough (2007) a otimização estrutural é uma técnica computacional que substitui o método de tentativa e erro, do procedimento de concepção de estruturas da forma tradicional por um processo orientado a objetos. A Figura 2.1 mostra um fluxograma básico do processo de otimização estrutural apresentado pelos autores.

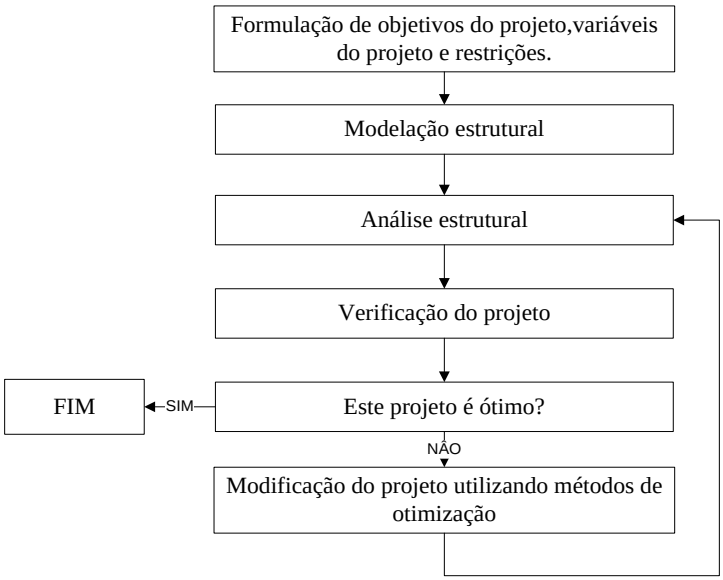


Figura 2.1 - Fluxograma de otimização estrutural. Modificado de Melchers e Hough (2007).

Silva (2011) indica que, geralmente, os problemas de otimização são resolvidos através de algoritmos de otimização determinísticos ou estocásticos. No método determinístico, os modelos mais comuns geralmente precisam da primeira derivada da função objetivo em relação às variáveis do projeto. Já para os algoritmos estocásticos, é avaliada diretamente a função objetivo e não precisa de derivadas sendo, portanto, conhecidos como métodos de ordem zero.

2.1 FORMULAÇÃO DE UM PROBLEMA DE OTIMIZAÇÃO

Para o entendimento do processo de otimização, é preciso conhecer algumas funções e variáveis que fazem parte do problema de otimização estrutural. De acordo com Christensen e Klarbling (2009), estes elementos são:

Função objetivo (*f (x)*): essa função é utilizada para classificar os projetos. Para cada possível projeto *f* retorna um número que traz informação sobre a qualidade do projeto. Frequentemente, a função objetivo a ser minimizada é o peso, deslocamento em uma dada direção, tensão efetiva ou até mesmo custo de produção, dentre outros.

Variável do projeto (*x*): a variável do projeto é uma função ou vetor que descreve o projeto. Durante o processo de otimização, esta função pode mudar. Representa por exemplo, a geometria ou a escolha do material. Quando se descreve a

geometria, a mesma pode estar relacionada com uma interpolação de forma ou pode ser simplesmente a área de uma barra ou a espessura de uma folha.

Variável fixa (*y*): a variável fixa é uma função ou um vetor que representa a resposta da estrutura. Para uma estrutura mecânica, a resposta pode ser o deslocamento, tensão, deformação ou força.

No dimensionamento de problemas de otimização estrutural, o objetivo é geralmente minimizar o peso da estrutura submetida a algumas restrições, tais como: as tensões, os deslocamentos e outras. Geralmente, um problema de otimização estrutural é apresentado da seguinte forma:

$$\text{Minimize } f(\boldsymbol{x})$$

$$\text{Sujeito a}$$

$$f(y(\boldsymbol{x})) \leq 0,\ i = 1, 2, \ldots, m \tag{2.1}$$

$$x_j \in \mathbb{R}, j = 1, 2, \ldots, n \tag{2.2}$$

onde $f(\boldsymbol{x})$ representa a função objetivo e $g(\boldsymbol{x})$ são as restrições comportamentais. m e n são o número de restrições e variáveis de desenho, respectivamente. $\boldsymbol{x} = [x_1, x_2, \ldots x_n]$ é o vetor das variáveis do projeto. Estas restrições podem ser classificadas em dois tipos: explícitas e implícitas. Restrições explicitas (Eq. 2.1) que são analisadas sem um sistema de simulação. Em contraste, as restrições implícitas (Eq. 2.2) requerem uma análise e verificação do projeto (Zuo *et al*. 2011).

2.2 TIPOS DE OTIMIZAÇÃO

Segundo Pizzirani (2003) e Fonseca (2007), a otimização estrutural pode ser dividida em três tipos: otimização dimensional, otimização de forma e otimização topológica. A seguir são descritos os três tipos de otimização aplicados a treliças.

2.2.1 Otimização Dimensional

Esta otimização é uma das mais simples no campo da otimização estrutural. Neste caso, a forma da estrutura é conhecida e o objetivo é otimizar a estrutura ajustando o tamanho dos componentes. Para uma estrutura de treliça, a variável do projeto é dada pela seção transversal das barras. O processo de otimização dimensional busca uma melhor distribuição das áreas, visando a minimização ou maximização da função objetivo. A

Figura 2.2 mostra um problema de otimização de dimensionamento para uma estrutura de treliça.

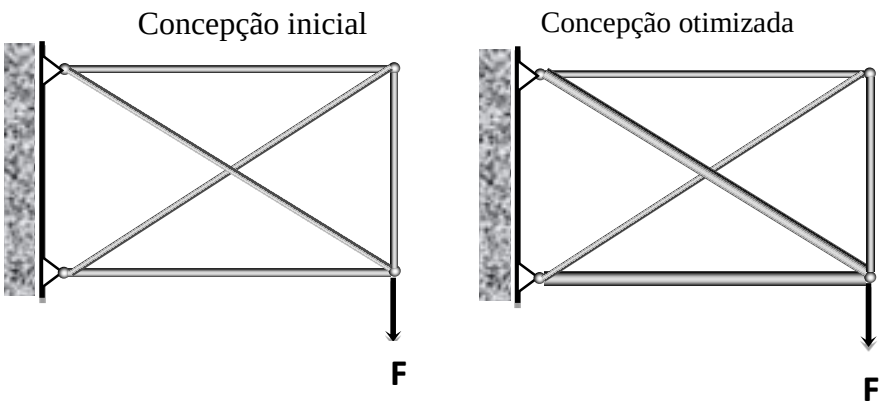


Figura 2.2 - Otimização da seção transversal dos elementos de uma treliça.

Em casos de treliças com poucos elementos é fatível a determinação das áreas de seção de forma analítica. A minimização do peso da treliça sujeita a restrições pode ser realizada, por exemplo, através da otimização utilizando multiplicadores de Lagrange. Em caso de treliças com vários elementos, este procedimento torna-se dispendioso e de difícil aplicação.

2.2.2 Otimização da forma

Na otimização de forma pretende-se encontrar o domínio ótimo do problema. Esse domínio é variável, pois o objetivo é encontrar a melhor forma da estrutura. Esse tipo de otimização é mais completo porque modifica a malha de elementos finitos levando a uma possível convergência da solução. Durante o processo de otimização, alguns pontos nodais podem mudar sua posição, gerando distorções no comprimento dos elementos. Nesse sentido, além das restrições usuais (deslocamento, tensões, frequências, cumprimento de flambagem, etc.) devem-se incorporar restrições que evitem soluções hipoestáticas. Na Figura 2.3, tem-se um exemplo deste tipo de otimização.

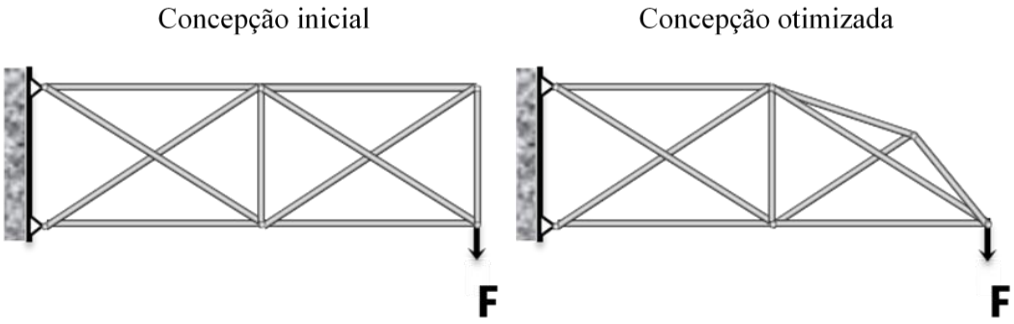


Figura 2.3 - Otimização de forma.

De maneira similar à otimização dimensional, a otimização de forma em treliças com poucos elementos pode ser realizada de forma analítica. Este procedimento pode ser de grande utilidade na verificação de procedimentos numéricos de otimização.

2.2.3 Otimização topológica

Este tipo de otimização é a mais geral dentro do campo da otimização estrutural. Com a otimização topológica não é possível ter uma noção da forma ou topologia resultante, número de buracos, elementos, etc. O propósito geral, para um dado domínio, é encontrar a distribuição ótima do material dentro de um elemento.

Neste tipo de otimização, a variável do projeto está associada à distribuição espacial do material. Para o caso de problemas discretos (treliças), alguns dos elementos que compõem a estrutura podem ser subtraídos ou adicionados da composição. Isto é possível desde que exista alta conectividade entre os nós de forma a evitar a obtenção de treliças hipoestáticas. Na Figura 2.4 é apresentada uma treliça na qual se fez uma mudança na topologia, onde após a otimização alguns elementos da treliça foram retirados. No caso de problemas contínuos, algumas regiões do domínio podem ser subtraídas, como pode ser visto na Figura 2.5. Isto pode ser realizado através do uso de elementos finitos sólidos conjuntamente com a técnica de programação linear.

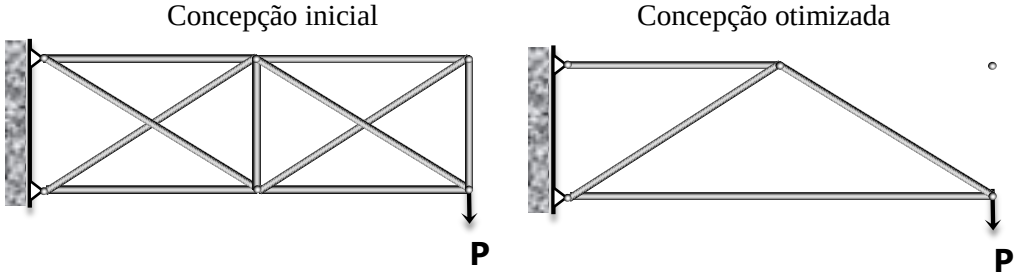


Figura 2.4 - Otimização topológica de uma treliça.

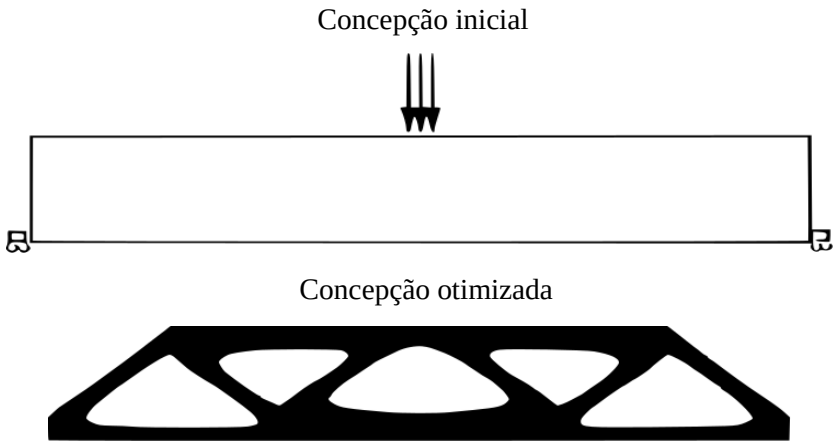


Figura 2.5 - Otimização topológica bidimensional (Christensen e Klarbling, 2009).

2.3 MÉTODOS DE OTIMIZAÇÃO

Apesar da individualidade de cada algoritmo, existem algumas similaridades que originam a concepção de grupos. Encontram-se normalmente na literatura três conjuntos principais dos métodos de procura: métodos determinísticos, métodos enumerativos e métodos estocásticos. Soares (1997) apresenta uma classificação das técnicas de procura (Tabela 2.1).

Tabela 2.1 - Técnicas de procura (Soares, 1997).

Técnicas de procura	Métodos Determinísticos	Programação Linear	Simplex	
		Programação Não-Linear	Sem Cálculo de Derivadas	Brent,Powell,Rosenbrock e outros
			Com Cálculo de Derivadas	Dbrent,Gradiente;Newton,Steepest,Descent
			Direções Conjugadas	BFGS,DFO,Fletcher & Reeves.
			Métodos das penalidades	Exterior,Interior,Interior Extendida
		Outros	Elipsóide, Grid	
	Métodos Enumerativos	Programação Dinâmica		
	Métodos Estocásticos	Computação evolucionária	Estratégias evolucionárias	
			Algoritmos Genéticos	
		Outros	Tabu	
			Reconhecimento Simulado	

O primeiro grupo constituído por os métodos determinísticos associam algoritmos que fazem uso do cálculo de derivadas e precisam de algum tipo de informação do gradiente na

procura do extremo global. Nele, o ponto inicial é o ponto de partida para a iteração seguinte, por tanto, a procura é local. Uma das desvantagens é que a solução encontrada tem uma grande possibilidade de ser um extremo local. Além desse problema, outro problema aparece quando a função a ser tratada não é contínua e gera uma derivação complicada. Por outro lado, estes métodos possuem grande rapidez e funcionam bem para problemas unimodais contínuos.

O método enumerativo como técnica de procura é um método muito mais simples. Neste caso, o algoritmo procura todas as combinações possíveis de soluções em um espaço finito de procura ou um espaço contínuo e discreto. Esta técnica é utilizada quando a implementação não seja complicada, uma vez que este pode tornar-se inviável para regiões muito grandes e, consequentemente, a eficiência fica prejudicada.

O último grupo, dos métodos estocásticos, tem ganhado popularidade nos últimos anos. Eles procuram em vários pontos do espaço por meio de regras de probabilidade e não precisam de cálculos de derivadas. Entretanto, não garantem a obtenção da melhor solução possível.

Uma das maiores preocupações em um algoritmo de otimização é a robustez, que faz a diferença entre eficiência e eficácia. Neste caso, se o método é considerado robusto, sua solução é mais confiável. Dentro do grupo de métodos estocásticos, encontram-se, por exemplo, os métodos de colônia de abelha, colônia de formigas e de Algoritmos Genéticos o qual é descrito no Capítulo 3.

No Brasil, autores como Afonso e Vaz 2000a, 2000b, tem desenvolvido muitos estudos na área da otimização de estruturas. Os autores desenvolveram trabalhos na área de Engenharia Civil, especialmente em Estruturas, com ênfase em métodos de otimização principalmente em temas de metodologias de aproximação, otimização de forma e topológica, otimização multiobjetivo e, também, em metodologias de otimização em problemas de Engenharia de reservatórios de petróleo. Outras referências dos trabalhos desenvolvidos por estes e outros autores, podem ser encontradas em Afonso *et al.* (2005), Afonso *et al.* (2004), Afonso *et al.* (2001).

3 ALGORITMOS GENÉTICOS

Os AGs são métodos heurísticos de busca inspirados na teoria da evolução natural. Esses métodos são baseados na teoria da evolução proposta por Charles Darwin. Segundo Fonseca (2007), o estudo dos Algoritmos Genéticos originou-se com os demais algoritmos tidos como evolucionistas, citando-se: a programação evolucionista e as estratégias evolucionistas. Ambos se baseiam no conceito de população de candidatos para obter a solução do problema. Esses indivíduos são modificados por processos de seleção, recombinação e mutação genética, de forma a promover a evolução dos indivíduos.

Goldberg (1989) descreve o funcionamento dos Algoritmos Genéticos. Nestes algoritmos, a troca de informação gera um procedimento de busca capaz de encontrar a melhor solução para uma série de problemas por meio da combinação de conceitos de seleção natural e operadores genéticos.

Outra definição de Algoritmo Genético foi feita por Koza (1994). Para o autor, o Algoritmo Genético é um algoritmo matemático fortemente paralelizável. Este algoritmo permite transformar uma população (conjunto de possíveis soluções), cujos indivíduos têm uma aptidão associada, em uma nova população (seguinte geração). Essa transformação é feita utilizando operações baseadas nos princípios darwinianos de reprodução e sobrevivência dos melhores indivíduos (os mais aptos).

Segundo Lemonge (1999), existem cinco aspectos fundamentais, os quais são usados para resolver um problema de otimização utilizando Algoritmos Genéticos:

1. Codificação genética de soluções para o problema;
2. Procedimento para criar uma população inicial de soluções;
3. Função de avaliação que proporciona a aptidão de cada indivíduo;
4. Operadores genéticos que manipulam a codificação dos pais durante o processo de reprodução, dando origem a novos indivíduos; e
5. Parâmetros a serem utilizados no algoritmo durante o processo de cruzamento e mutação.

Segundo Ignízio e Cavalier (1994), as heurísticas têm se destacado como as abordagens mais promissoras para solução de problemas de otimização. Enquanto os algoritmos exatos garantem uma solução ótima para certos tipos desses problemas, os métodos heurísticos

não podem provar a otimização das suas soluções, mas oferecem soluções aceitáveis, inclusive para problemas complexos e de grande porte, com baixo custo computacional.

3.1 TERMINOLOGIA EM ALGORITMOS GENÉTICOS

Dado que o método de AGs está baseado em fenômenos da biologia, exatamente na teoria da evolução, muitos termos são originados dela. A seguir é apresentada a terminologia convencional, utilizada no estudo de otimização através do uso dos AGs:

Gene: é um parâmetro codificado no cromossomo, ou seja, um elemento de vetor que representa o cromossomo. A quantidade de valores que pode tomar um parâmetro pode ser discreta e deve-se ter em conta que quanto maior o número mais complexo tende a ser o espaço de busca. A estrutura do gene depende da função de codificação;

Alelo: representa os valores que o gene pode assumir. Por exemplo, um gene que representa o parâmetro de forma geométrica da seção transversal de uma barra, poderia ter os alelos de parâmetros circulares, retangulares, etc;

Cromossomo: está formado por um conjunto de genes (cadeia de caracteres) que faz referência ao espaço de busca genotípico. Nos AGs, cada cromossomo representa uma estrutura de dados que codifica uma solução para um problema;

Espaço Fenotípico: este espaço é formado por um conjunto das variáveis do projeto; por exemplo, áreas da seção transversal, coordenadas nodais, etc;

Espaço Genotípico: representa a informação contida no cromossomo. Este espaço codifica ao espaço fenotípico atribuindo nele uma faixa de valores. Por exemplo, dentro um espaço fenotípico que contém as áreas da seção transversal, o espaço genotípico são os valores das áreas que adota um gene dentro de um cromossomo;

Indivíduo: representa uma das soluções do problema proposto. Em geral, existem dois tipos de soluções, a genotípica e fenotípica. A solução genotípica é representada pelos cromossomas, e na fenotípica, pelas variáveis do projeto;

População: é um conjunto de indivíduos que representam todas as possíveis soluções avaliadas durante a geração (iteração). Idealmente, a primeira população

deveria ser formada por indivíduos que estejam contidos no espaço de busca. Por esta razão, a população inicial é aleatória para evitar uma convergência prematura. A determinação do tamanho da população necessária em um processo de otimização não é uma tarefa fácil. Esta depende da complexidade do problema, de modo que quando maior o espaço de busca, maior tem que ser o tamanho da população para evitar um resultado prematuro o para que a convergência obtida não derive em uma solução ótima; e

Geração: é o número da iteração que o AG executa. As gerações dizem respeito à evolução que as populações vão tendo ao longo do tempo em função da aplicação dos operadores genéticos de reprodução.

3.2 OPERADORES GENÉTICOS

Os Algoritmos Genéticos precisam de parâmetros de controle em sua estrutura de funcionamento. De acordo com Silva (2011), a eficiência do funcionamento do algoritmo genético depende diretamente destes parâmetros, destacando-se: o tamanho da população e os operadores genéticos, com suas respectivas probabilidades de aplicação.

O princípio básico dos operadores genéticos é transformar a população através de sucessivas gerações, assim como acontece na natureza. Os operadores genéticos são necessários para que a população se diversifique e mantenha as características de adaptação adquiridas nas gerações anteriores (Goldberg, 1989). Segundo Sanchez (2012), existem os seguintes operadores: iniciação, reprodução (seleção + recombinação) e mutação.

3.2.1 Configuração

A escolha correta dos parâmetros que influenciam o comportamento dos AGs é um dos aspectos mais importantes dentro da estratégia de configuração. Para uma boa escolha dos parâmetros, têm-se algumas recomendações encontradas na literatura descrita nas subseções seguintes:

Codificação: no AG é preciso codificar cada indivíduo através de um cromossomo. A codificação feita para o indivíduo e a avaliação da aptidão são os principais problemas na aplicação do AG. A estrutura parametrizada, chamado de fenótipo, precisa ser codificada no genótipo. Assim, cada indivíduo (possível solução) é

representado no AG por seu cromossoma, um vetor de genes, cada um contendo o valor de um parâmetro;

Tamanho da população (n_{pop})**:** indica o número de indivíduos que tem a população. O desempenho do AG depende em grande medida deste parâmetro. Com uma população pequena o desempenho pode cair, pois esta representaria somente uma pequena parte do espaço de busca do problema. Já com grandes populações apresentará mais diversidade de soluções, mas precisa de ferramentas computacionais de maior capacidade. Portanto, o desempenho global e a eficiência dos AGs estão influenciados por este parâmetro. Segundo Lemonge (1999), ainda não existem critérios para definir o tamanho da população. Portanto, a escolha vai depender de cada problema e da experiência adquirida ao longo do processo.

3.2.2 Operador de iniciação

Este operador é responsável pela geração da população inicial e de reinicializar uma população em determinadas condições. Esta inicialização geralmente é feita através da geração de um conjunto de genes de cada indivíduo, de forma aleatória, dentro do domínio para o qual foi definido cada gene.

3.2.3 Operadores de reprodução

Os AGs estão fundamentados no processo cíclico de reprodução, onde se gera indivíduos novos e melhores. Estes são os operadores que dominam o processo de busca. O ciclo reprodutivo consta de três passos: a seleção dos pais, o cruzamento (*crossover*) dos pais para gerar novos indivíduos (filhos) e a mutação que altera em maior ou menor medida o cromossomo dos novos filhos. Cada um destes passos é implementado dentro do algoritmo como um operador diferente. A Figura 3.1 apresenta um fluxograma de otimização, utilizando Algoritmos Genéticos nos três procedimentos citados anteriormente.

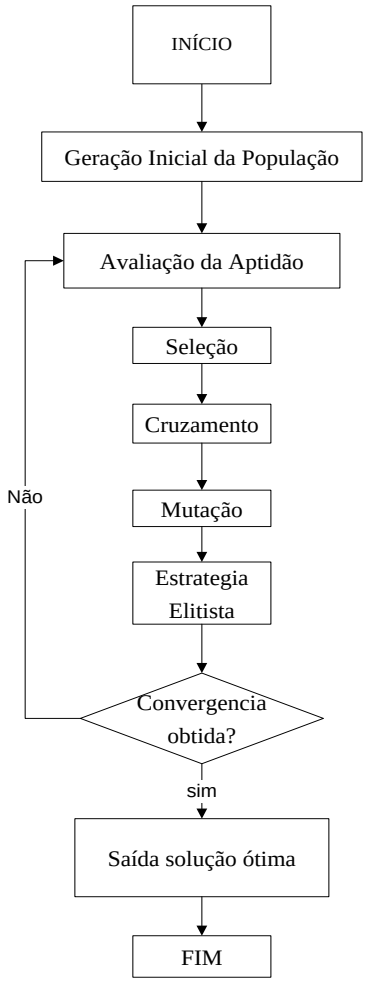


Figura 3.1 - Fluxograma de otimização utilizando Algoritmos Genéticos. Modificado de Zuo *et al*. (2011).

3.2.4 Operador de seleção

Por meio deste operador, selecionam-se, por diferentes procedimentos, indivíduos da população (pais) para participar no processo de reprodução. O conjunto dos indivíduos selecionados dará origem aos filhos, depois de aplicar-lhe os operadores de cruzamento e mutação.

Goldberg e Deb (1991) e Miller e Goldberg (1996) fazem referência aos diferentes operadores de seleção, os quais são:

> **Seleção aleatória:** esta técnica seleciona aleatoriamente os pais da nova população. Ela também apresenta a vantagem de manter a diversidade da população e o inconveniente de realizar o processo de convergência com maior lentidão;

Seleção por roleta: este operador de seleção é conhecido como seleção estocástica de substituição. A implementação é feita com uma roleta de cassino e é dividida em partes proporcionais à aptidão de cada indivíduo de modo que cada parte coincide com a probabilidade de seleção p_s do mesmo. A seguir a roleta é lançada μ vezes, onde μ é o número de indivíduos da população, selecionando em cada tirada um progenitor formando deste modo os pares que gerarão os filhos;

Seleção por grupos: também conhecida como seleção por blocos. A implementação é feita dividindo a população em k grupos, atribuindo-se uma probabilidade de seleção a cada grupo. A probabilidade de seleção de um indivíduo de um grupo dado se obtém dividendo a probabilidade de seleção do grupo pôr o número de indivíduos desse grupo; e

Seleção por torneio: foi popularizada por Goldberg e Deb (1989) sendo uma das mais simples de implementar. Dois ou mais indivíduos da população são selecionados de modo que eles competem entre si para ser um dos pais da nova geração. O indivíduo com maior aptidão será vencedor. Este processo se repete até escolher todos os pais.

3.2.5 Operador e taxa de cruzamento

A taxa de cruzamento indica a probabilidade de um indivíduo de ser recombinado com outro. Segundo Castro (2005), quanto maior a probabilidade de cruzamento (p_c), novos indivíduos são introduzidos mais rapidamente na população. Mas no caso que a probabilidade seja muito alta, os indivíduos com uma boa qualificação poderão ser retidos mais rapidamente da população.

O operador de cruzamento, também chamado operador de recombinação ou *crossover*, é responsável de realizar a combinação de dois indivíduos para gerar dois ou mais filhos. O mecanismo de cruzamento permite a combinação de dois indivíduos com certas características diferentes e gera filhos que combinam as duas características. Este é um dos principais operadores de busca nos Algoritmos Genéticos. A eficiência da busca depende em grande parte da diversidade da população e da necessidade seletiva. Quanto mais diferentes são os pais, maior será a influência do operador no funcionamento do algoritmo.

Usualmente, os operadores de cruzamento dividem os cromossomos dos pais em duas ou mais partes, após definirem pontos de corte. Posteriormente, as partes resultantes são recombinadas para gerar novos indivíduos. Existem duas formas de realizar o corte: corte por genótipo, cortando por qualquer ponto do cromossomo e por fenótipo. Nesse segundo caso, os genes se agrupam em função do fenótipo que eles definem e o corte é feito entre os blocos de genes. Existem diferentes métodos para definir a localização do ponto (ou os pontos) de corte. Para definir estes pontos de corte existem os diferentes operadores de cruzamento mencionados a continuação.

a) Operadores de cruzamento determinísticos

Dentro destes operadores de cruzamento determinísticos estão aqueles operadores onde os cromossomos dos filhos são obtidos misturando os genes dos pais.

Cruzamento por um ponto: este operador de cruzamento é um dos mais simples. Foi proposto por Holland (1975). No caso de que o corte seja realizado por genótipo, a implementação do algoritmo se realiza do seguinte modo: inicialmente é gerado um número inteiro aleatório $r \in [1, n_g - 1]$, *onde* n_g é o comprimento do cromossomo. A seguir se cortam dois cromossomos, ou seja, os dois pais são cortados no ponto r e se permutam, gerando os filhos como mostra a Figura 3.2.

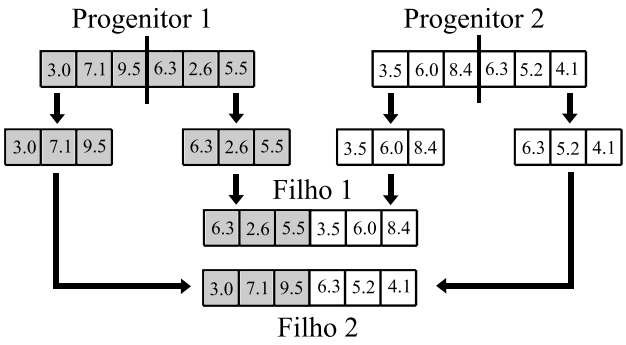


Figura 3.2 - Cruzamento por um ponto.

Cruzamento por *n* pontos: é possível realizar esta técnica por genótipo, onde o número máximo de cortes possível é $n_g - 1$, sendo n_g é o número de genes que possui o cromossomo. Quando o corte é feito por fenótipo, é possível realizar tantos pontos de corte $n_f - 1$ quantos grupos de genes (fenótipo) possui o cromossomo. Na Figura 3.3 apresenta um exemplo do funcionamento deste operador.

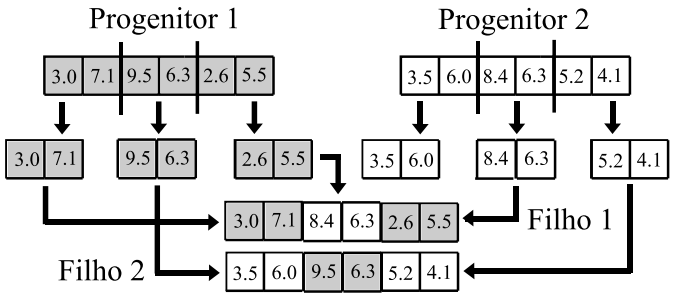


Figura 3.3 - Cruzamento por dois pontos.

b) Operadores de cruzamento aritmético

Dependendo dos números de genes combinados, se definem três tipos de recombinação aritmética, a recombinação simples, recombinação aritmética simples e recombinação aritmética completa.

Recombinação simples: nesta implementação, seleciona-se um gene aleatório i do cromossomo (o do fenótipo) e, para o primeiro filho, se copia o cromossomo do primeiro pai até o gene i formando o resto do cromossomo por meio de algum dos operadores anteriormente descritos. Para o segundo filho é gerado do mesmo modo, mas usando o cromossomo do segundo pai. A seguinte equação apresenta a implementação deste operador. Por exemplo, a Figura 3.4 mostra o uso deste operador para um valor de $\lambda_1 = 0{,}5$ e um valor de $i = 4$.

$$
\begin{aligned}
&S_{p_1} = \{p_1, p_2, \ldots, p_i, \ldots, p_n\} \\
&S_{p_2} = \{q_1, q_2, \ldots, q_i, \ldots, q_n\} \\
&S_{h_1} = \{p_1, p_2, \ldots, p_i,\ \lambda_1 q_{i+1} + (1-\lambda_1) p_{i+1}, \ldots, \lambda_1 q_n + (1-\lambda_1) p_n\} \\
&S_{h_2} = \{q_1, q_2, \ldots, q_i,\ \lambda_1 p_{i+1} + (1-\lambda_1) q_{i+1}, \ldots, \lambda_1 p_n + (1-\lambda_1) q_n\}
\end{aligned}
\tag{3.1}
$$

onde, S_{p_1} e S_{p_2}são os cromossomos dos pais, S_{h_2} e S_{h_1} são os cromossomos dos filhos, p e q representam os genes dos cromossomos, λ_1 é o fator de alteração que modifica ao cromossomos dos pais, ($0 < \lambda_1 < 1$), e i é a posição do gene que vai ser modificado.

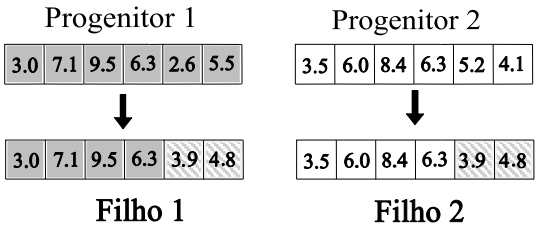


Figura 3.4 - Cruzamento aritmético lineal com recombinação simples.

Recombinação aritmética simples: para esta implementação seleciona-se um ponto aleatório i do genótipo (ou do fenótipo). A continuação, para o primeiro filho, se copia todo o cromossomo do primeiro pai com exceção do gene i, o qual será formado por meio de um dos operadores anteriormente descritos. Para o segundo filho é gerado do mesmo modo, mas invertendo os cromossomos dos pais. A seguinte equação e a Figura 3.5 apresentam a implementação deste operador com valores de $\lambda_1 = 0{,}5$ e um valor de $i = 5$.

$$
\begin{aligned}
S_{p_1} &= \{p_1, p_2, \ldots, p_i, \ldots, p_n\} \\
S_{p_2} &= \{q_1, q_2, \ldots, q_i, \ldots, q_n\} \\
S_{h_1} &= \{p_1, p_2, \ldots, p_{i-1}, \lambda_1 q_i + (1 - \lambda_1) p_i, p_{i+1}, \ldots, p_n\} \\
S_{h_2} &= \{q_1, q_2, \ldots, q_{i-1}, \lambda_1 p_i + (1 - \lambda_1) q_i, q_{i+1}, \ldots, q_n\}
\end{aligned}
\tag{3.2}
$$

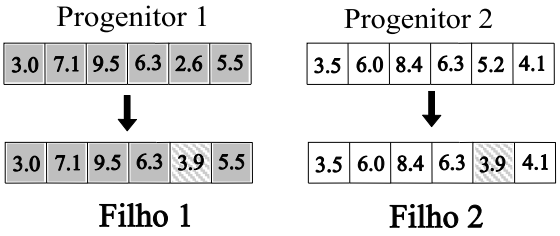


Figura 3.5 - Cruzamento aritmético lineal com recombinação aritmética simples

Recombinação aritmética completa: neste caso, todos os genes dos filhos formam-se por meio da combinação dos pais. Deve-se ter cuidado em não realizar um cruzamento intermédio $(\lambda_1 = 0,5)$ porque neste caso os filhos ficam iguais, podendo provocar a perda da diversidade na população. A seguinte equação apresenta a implementação deste operador. Por exemplo, a Figura 3.6 apresenta o uso de este operador para um valor de $\lambda_1 = 0{,}3$.

$$
\begin{aligned}
S_{p_1} &= \{p_1, p_2, \ldots, p_i, \ldots, p_n\} \\
S_{p_2} &= \{q_1, q_2, \ldots, q_i, \ldots, q_n\} \\
S_{h_1} &= \{\lambda_1 q_1 + (1 - \lambda_1) p_i, \ldots, \lambda_1 q_n + (1 - \lambda_1) p_n\}
\end{aligned}
\tag{3.3}
$$

$$S_{h_2} = \{\lambda_1 p_1 + (1 - \lambda_1) q_1, \ldots, \lambda_1 p_n + (1 - \lambda_1) q_n\}$$

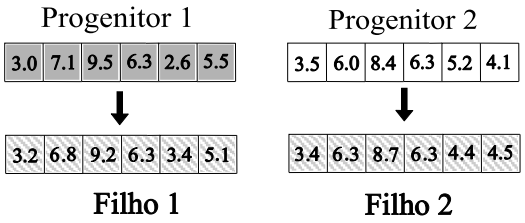


Figura 3.6 - Cruzamento aritmético lineal com recombinação aritmética completa.

3.2.6 Operador de mutação

Este operador efetua uma alteração randômica no gene do cromossomo depois de ter sofrido o processo de cruzamento. Assim como os outros parâmetros são importantes, uma boa escolha deste fator determinara um bom resultado da aplicação a ser resolvida. Dentro deste operador é importante definir a taxa e fator de mutação empregado.

Taxa e fator de mutação: segundo Fonseca (2007), a mutação é considerada um operador secundário por se limitar a recuperar a diversidade genética da população e controlar os mínimos locais, devido à aplicação do crossover. Segundo Silva (2001), a probabilidade de mutação (p_m) é utilizada para garantir a diversidade genética, e assim, evitar que o algoritmo fique estagnado em regiões do espaço de busca por falta de diversidade. Isto ocorre quando a probabilidade é muito baixa, e quando a taxa de aplicação é excessivamente alta, pode ocorrer que o algoritmo assuma características de busca aleatória.

Para a implementação deste operador, é utilizado uma probabilidade de mutação dentro da população e dentro do cromossomo. Os indivíduos eleitos neste processo são modificados com uma taxa de mutação, que indica a probabilidade que o conteúdo de uma determinada posição dentro do cromossomo seja alterado por um determinado fator. A Figura 3.7 ilustra o procedimento básico adotado por este operador.

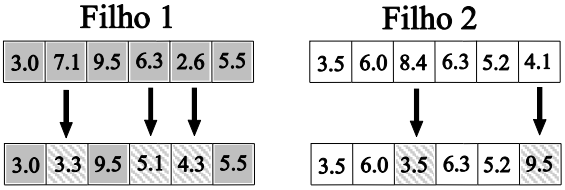


Figura 3.7 - Operador de mutação.

Para codificações inteiras ou reais, empregam-se habitualmente os seguintes operadores de mutação:

Mutação aleatória uniforme: foi descrita por Michalewicz (1992). Um gene i é escolhido para mutação e depois se gera um número aleatório dentro do domínio do gene, isto é, $r \in [l_i, u_i]$, onde l_i e u_i representam os valores mínimo e máximo do gene que pode ter;

Mutação de convolução gaussiana: segundo Sanchez (2012), o operador de mutação de convolação gaussiana é um dos mais empregados na atualidade. Neste caso, a mutação é realizada por meio da adição do ruído gaussiano ao gene que se deseja alterar. Este operador modifica o gene com valores próximos ao gene inicial. Para isto é utilizada uma distribuição normal $N(\hat{\mu}, \sigma^2)$ onde o valor de $\hat{\mu}$ geralmente é igual a zero e a variância σ^2 controla os genes distantes à média. Por ser possível gerar valores fora dos limites do domínio do gene é necessário rejeitar ou truncar estes valores.

3.2.7 Operador de substituição

Existem duas classes fundamentais de algoritmos genéticos dependendo da estratégia de substituição empregada: os AGs com substituição geracional e os AGs com brecha geracional. Na primeira classe, a população de pais é completamente substituída pela população de filhos uma vez sofrida mutação. Na segunda classe, existe um grupo de pais que sobrevivem.

Estas diferentes estratégias adotadas para os AGs com brecha geracional resultam em diferentes operadores de substituição:

Substituição dos menos aptos: foi definido por De Jong e Sarma (1992) dentro de um modelo de algoritmo genético. Com este operador os filhos gerados substituem aos indivíduos menos aptos da população;

Substituição aleatória: com este operador os filhos gerados substituem de forma aleatória aos indivíduos da população;

Torneio a morte: com este operador se seleciona aleatoriamente um conjunto de indivíduos da população, onde o menos apto é substituído por um filho;

Substituição do indivíduo mais velho: com este operador se substitui o indivíduo mais velho; aquele que sobreviveu durante várias gerações. Para evitar eliminar o indivíduo mais apto é necessário incorporar um operador de elitismo;

Seleção conservativa: foi definido por Smith (2007). Este operador combina a estratégia de substituição do mais velho com um torneio de seleção binária. Este operador realiza um torneio entre o indivíduo mais velho e outro selecionado aleatoriamente dentro da população. Se o mais velho é mais apto então sobrevive, e o outro é substituído por um filho novo. No caso contrário é o velho quem é substituído. Esta estratégia tem a vantagem de não substituir o indivíduo mais velho se também for o mais apto;

Substituição dos progenitores: este operador por médio de uma estratégia de supervivência predefinida decide se o filho substitui algum dos pais ou bem os dois progenitores sobrevivem. A estratégia pode estar baseada na aptidão, distância entre cromossomos, etc;

Elitismo: segundo Coley (1999), este operador de substituição é um dos mais utilizados atualmente. Tem por objeto evitar a eliminação do indivíduo ou grupo de indivíduos mais aptos da população. Para alguns problemas, o elitismo é vantajoso porque retarda o algoritmo, permitindo-lhe que explore mais o espaço de busca antes da convergência.

3.3 VANTAGENS E DESVANTAGENS DOS ALGORITMOS GENÉTICOS

Os AGs, apesar de terem-se destacado como as abordagens mais promissoras para solução de problemas de otimização, possuem vantagens e desvantagens no uso.

3.3.1 Vantagens

A seguir, são resumidas as principais vantagens dos AGs descritas por Castro (2001), Guerra (2008) e Castro (2005), quando comparados a outras metodologias de otimização:

- Apresentam um bom desempenho numérico para uma grande escala de problemas;
- São de fácil implementação numérica e proporcionam maior flexibilidade no tratamento do problema a ser resolvido;
- Funcionam tanto com parâmetros contínuos como discretos ou uma combinação

deles;

- São mais robustos e menos suscetíveis a obter ótimos locais devido aos operadores genéticos serem probabilísticos;
- Otimizam simultaneamente um grande número de variáveis;
- Não precisa de computadores com processadores de grande desempenho;
- Realizam buscas simultâneas em várias regiões do espaço de busca, devido que trabalhar com uma população e não sobre simples indivíduos;
- São flexíveis a trabalhar com restrições arbitrárias;
- O campo de aplicação é em problemas complexos, com múltiplos mínimos e/ou máximos;
- Permitem a implementação utilizando computação paralela, o que permite reduzir o tempo gasto na otimização.

3.3.2 Desvantagens

Assim como tem muitas vantagens, os AGs ainda não são eficientes para muitos problemas. Em alguns casos são considerados lentos a comparação de muitos métodos determinísticos. Além disso, Fogel (1997), em seu trabalho *"The Advantages of Evolutionary Computation",* descreve algumas das seguintes desvantagens:

- Tarda em achar o ótimo global exato (se possível);
- Caso não seja adequadamente configurado, pode levar prematuramente à convergência do problema em um ótimo local;
- Pode-se requerer um grande número de avaliações da função de aptidão no processo de busca;
- As configurações das variáveis e a escolha dos operadores geram um número grande de combinações de parâmetros a serem investigados.

4 ELEMENTOS FINITOS DE BARRA

O Método dos Elementos Finitos é uma técnica numérica para análise de estruturas e meios contínuos e está baseado no conceito de discretização. Neste método, os diversos fenômenos físicos que ocorrem em meios contínuos são descritos através de equações diferenciais parciais, com determinadas condições de contorno. O uso deste método permite transformar um problema complexo na soma de vários problemas simples.

A ideia principal do MEF consiste em dividir o domínio do problema (meio contínuo) em sub-regiões de geometria simples. Estas subdivisões são chamadas de elementos finitos. Advém daí o nome de Método dos Elementos Finitos, estabelecido por Ray Clough, na década de 50. Os elementos finitos utilizados na discretização do domínio do problema são conectados entre si através de determinados pontos, denominados nós ou pontos nodais. Ao conjunto de elementos finitos, dá-se usualmente o nome de malha de elementos finitos.

Diversos tipos de elementos finitos têm sido desenvolvidos. Estes apresentam formas geométricas diversas em função do tipo e da dimensão do problema. Para este estudo, que trata de otimização de treliças, o tipo de elemento finito utilizado é o elemento de barra com dois nós. Segundo Liu e Quek (2013), um elemento de treliça é um dos membros estruturais mais simples e mais amplamente utilizados. Este elemento é dado por uma barra reta que é projetada para levar apenas forças axiais, pois deforma apenas na sua direção axial. A seção transversal da barra pode ser arbitrária, mas as dimensões da seção transversal devem ser muito menores do que na direção axial.

Em treliças tridimensionais existem três componentes nas direções x, y e z para os deslocamentos e para as forças. Conceitualmente, os elementos de barra que compõem as treliças são unidos por pinos ou dobradiças (não por soldagem), de modo que não existem momentos transmissíveis entre as barras.

No livro de Rao (2004), o autor desenvolve os elementos finitos desde os conceitos básicos, procedimentos e formulações a partir de um elemento de barra de dois nós. Neste trabalho, com a finalidade de analisar as treliças estaticamente, é utilizada uma formulação generalizada, a mesma que é apresentada em termos de Jacobiano.

4.1 FORMULAÇÃO GENERALIZADA PARA UM ELEMENTO DE BARRA.

A formulação a seguir está baseada em Durand (2008), a qual pode ser aplicada a elementos de barra com dois ou mais nós. A posição dos elementos de barra na malha de elementos finitos é tal que seus nós são sempre coincidentes com os nós dos elementos circundantes.

Para a análise de equilíbrio, é necessária a determinação da matriz de rigidez de uma barra e para a formulação desta matriz é preciso determinar uma expressão para a deformação axial no domínio da barra em função dos seus deslocamentos nodais. Desta forma, para um ponto qualquer que pertença a uma barra de comprimento L, é possível associar ao mesmo um comprimento infinitesimal $\mathrm{d}L$. A deformação axial neste ponto é dada por $\frac{\Delta \mathrm{d}L}{\mathrm{d}L}$. Considerando que os componentes de deslocamento no ponto, nas três direções cartesianas, são dadas por $\mathbf{u} = [u\ v\ w]^T$, as componentes da variação do comprimento com relação a $\mathrm{d}L$ são dadas por $\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}L} = \left[\frac{\partial u}{\partial L} \quad \frac{\partial v}{\partial L} \quad \frac{\partial w}{\partial L}\right]^T$. Projetando o vetor $\frac{d\mathbf{u}}{dL}$ na direção $[r_1\ r_2\ r_3]$ da barra é possível obter a deformação axial no ponto em questão por meio de:

$$\varepsilon = [r_1\ r_2\ r_3]\left[\frac{\partial u}{\partial L} \quad \frac{\partial v}{\partial L} \quad \frac{\partial w}{\partial L}\right]^T \tag{4.1}$$

Considerando que a localização do ponto na barra é dada pela coordenada local ξ, os deslocamentos neste ponto podem ser expressos em função dos deslocamentos nodais e utilizando as funções de forma N que, por exemplo, para um elemento de barra de dois nós são dadas por:

$$\begin{aligned} N_1(\xi) &= \frac{1}{2} - \frac{\xi}{2} \\ N_2(\xi) &= \frac{1}{2} + \frac{\xi}{2} \end{aligned} \tag{4.2}$$

Desta forma e utilizando a regra da cadeia, é possível expressar a deformação axial como:

$$\varepsilon = [r_1\ r_2\ r_3]\left[\sum_{i=1}^{n} \frac{\partial N_i}{\partial \xi}\frac{\partial \xi}{\partial L}u_i \quad \sum_{i=1}^{n} \frac{\partial N_i}{\partial \xi}\frac{\partial \xi}{\partial L}v_i \ \sum_{i=1}^{n} \frac{\partial N_i}{\partial \xi}\frac{\partial \xi}{\partial L}w_i\right]^T \tag{4.3}$$

Considerando a matriz Jacobiana **J**, que para o caso de um elemento unidimensional localizado no espaço 3D é dado por $\mathbf{J} = \left[\frac{\partial x}{\partial \xi} \quad \frac{\partial y}{\partial \xi} \quad \frac{\partial z}{\partial \xi}\right]$ e sua correspondente norma $J = \sqrt{\left(\frac{\partial x}{\partial \xi}\right)^2 + \quad \left(\frac{\partial y}{\partial \xi}\right)^2 + \left(\frac{\partial z}{\partial \xi}\right)^2}$ avaliados em ξ, é possível substituir os termos $\frac{\partial \xi}{\partial L}$ por $\frac{1}{J}$ e $[r_1\, r_2\, r_3]$ por $\frac{\mathbf{J}}{J}$ na Eq. (4.3). Realizando estas substituições obtém-se:

$$\varepsilon = \frac{\mathbf{J}}{J^2}\left[\sum_{i=1}^{n} \frac{\partial N_i}{\partial \xi} u_i \quad \sum_{i=1}^{n} \frac{\partial N_i}{\partial \xi} v_i \quad \sum_{i=1}^{n} \frac{\partial N_i}{\partial \xi} \mathrm{w}_i\right]^T \tag{4.4}$$

Organizando esta equação de forma conveniente é possível reescrevê-la em forma matricial como:

$$\varepsilon = \underbrace{\frac{\mathbf{J}}{J^2}\begin{bmatrix} \frac{\partial N_1}{\partial \xi} & 0 & 0 & \frac{\partial N_2}{\partial \xi} & 0 & 0 & \cdots & \frac{\partial N_n}{\partial \xi} & 0 & 0 \\ 0 & \frac{\partial N_1}{\partial \xi} & 0 & 0 & \frac{\partial N_2}{\partial \xi} & 0 & \cdots & 0 & \frac{\partial N_n}{\partial \xi} & 0 \\ 0 & 0 & \frac{\partial N_1}{\partial \xi} & 0 & 0 & \frac{\partial N_2}{\partial \xi} & \cdots & 0 & 0 & \frac{\partial N_n}{\partial \xi} \end{bmatrix}}_{\mathbf{B}_{(3\times 3n)}} \begin{Bmatrix} u_1 \\ v_1 \\ w_1 \\ u_2 \\ v_2 \\ w_2 \\ \vdots \\ u_n \\ v_n \\ w_n \end{Bmatrix} \tag{4.5}$$

Esta equação associa a deformação axial com o vetor de deslocamentos nodais por meio da matriz deformação-deslocamento **B**. Uma vez obtida a matriz **B**, a matriz de rigidez de um elemento de barra pode ser calculada por meio da formulação convencional do MEF como:

$$\mathbf{K} = \int_V \mathbf{B}^T D \mathbf{B} \mathrm{d}V = AE \int_L \boldsymbol{B}^T \boldsymbol{B} dL \tag{4.6}$$

onde D representa o escalar constitutivo que relaciona as taxas de tensão e deformação axial ($\dot{\sigma} = D\dot{\varepsilon}$). Para uma barra linear elástica tem-se que $D = E$, em que E é o modulo de Young. Adicionalmente, V representa o volume do elemento de barra. O diferencial dV pode ser substituído por $dV = AdL$, em que L o comprimento e A é a área da secção transversal.

Para os casos de elementos de barra de dois nós, a aplicação desta formulação resulta em expressões sucintas para a matriz de rigidez. Por exemplo, para um elemento de barra no espaço 2D com representado na Figura 4.1, ela é dada por:

$$\boldsymbol{K} = \frac{EA}{L}\begin{bmatrix} l^2 & lm & -m^2 & -lm \\ lm & m^2 & -lm & -m^2 \\ -m^2 & -lm & l^2 & lm \\ -lm & -m^2 & lm & m^2 \end{bmatrix} \tag{4.7}$$

onde $l = cos\theta_1$ e $m = cos\theta_2$.

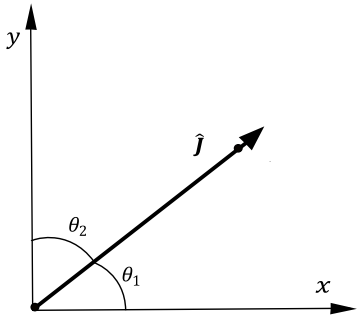


Figura 4.1 - Direção do elemento de barra no espaço 2D

Por sua vez, para um elemento de barra no espaço 3D, como apresentado na Figura 4.2, tem-se:

$$\boldsymbol{K} = \frac{EA}{L}\begin{bmatrix} l^2 & lm & ln & -l^2 & -lm & -ln \\ lm & m^2 & mn & -lm & -m^2 & -mn \\ ln & mn & n^2 & -ln & -mn & -n^2 \\ -l^2 & -lm & -ln & l^2 & lm & ln \\ -lm & -m^2 & -mn & lm & m^2 & mn \\ -ln & -mn & -n^2 & ln & mn & n^2 \end{bmatrix} \tag{4.8}$$

onde $l = cos\theta_1$, $m = cos\theta_2$ e $n = cos\theta_3$ são os cossenos diretores.

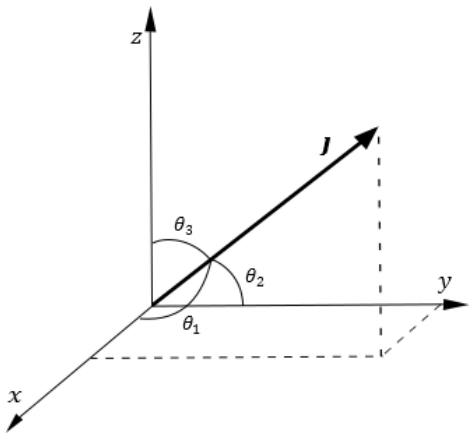


Figura 4.2 - Direção do elemento de barra no espaço 3D

Após a obtenção dos deslocamentos nodais e do incremento de deformações, as tensões axiais são obtidas por meio da relação constitutiva. Finalmente, o incremento de forças internas correspondente para um acréscimo de tensão axial $\Delta\sigma$ pode ser calculado através de:

$$\Delta\mathbf{F} = \int_V \mathbf{B}^T \Delta\sigma \mathrm{d}V = A \int_L \mathbf{B}^T \Delta\sigma \mathrm{d}L \tag{4.9}$$

4.2 APLICAÇÕES DA OTIMIZAÇÃO USANDO AG E MEF

A propagação dos Algoritmos Genéticos despertou grande interesse na área da Engenharia e nas ciências afins como uma ferramenta para a otimização. Por esta razão, muitos estudos, desde a Aeronáutica até análises de estruturas de proteínas, são feitos com as ajudas dos AGs. Autores como Goldberg *et al*. (1997) e Soares (1997) apresentam um resumo de exemplos das aplicações dos AGs e a evolução.

Na dissertação desenvolvida por Castro (2005), o autor apresenta um processo de otimização utilizando algoritmos genéticos do tipo geracional com codificação binária e estratégia elitista, associado ao Método dos Elementos Finitos. Os problemas tratados nesta dissertação foram de otimização do peso de estruturas reticuladas submetidas a restrições de tensão e deslocamento. A Figura 4.3 mostra um dos exemplos de otimização de peso constituído por uma treliça de 10 barras com uma carga P de 450kN. A Figura 4.4 mostra a convergência de peso próprio, de acordo com o número de gerações. O resultado obtido por Castro, foi muito próximo das soluções encontradas na literatura, mas precisou um número muito maior de gerações.

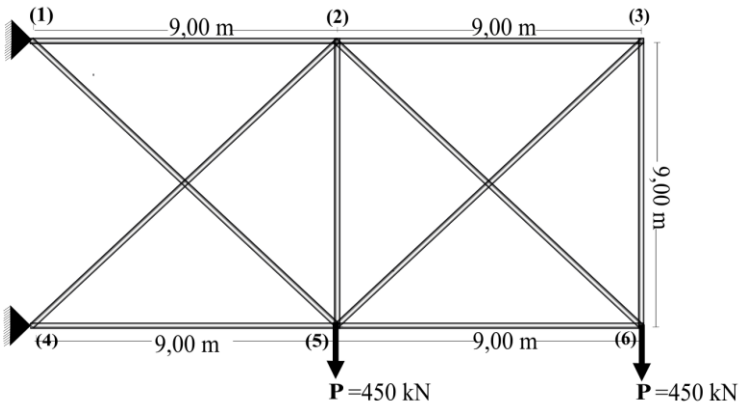


Figura 4.3 - Treliça bidimensional de 10 barras

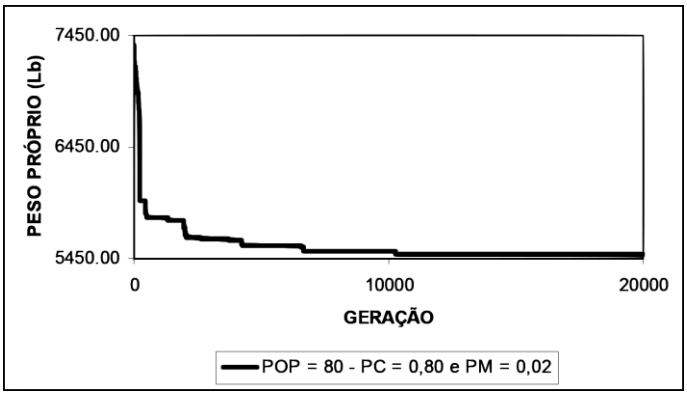


Figura 4.4 - Minimização do peso. Treliça de 10 barras (Castro, 2005).

Deb *et al.* (2001) utilizam os Algoritmos Genéticos, com codificação real, para otimizar as seções transversais, forma e topologia de treliças em 2D e 3D. O AG proposto utiliza um vetor que contém o comprimento, as áreas e as mudanças nodais, quando se tira algum elemento da treliça no processo de otimização topológica. Além disso, consideraram a inclusão de nós importantes na estrutura, usando um conceito de nós básicos e não básicos dentro da configuração da mesma. A técnica proposta permite encontrar uma solução intuitivamente ótima da configuração da treliça com o objetivo de diminuir o peso final. Foram estudados alguns casos comuns encontrados na literatura. O mesmo exemplo apresentado anteriormente e analisado por Castro (2005) foi estudado por Deb *et al.* (2001) com o objetivo de otimizar o peso da treliça por meio da otimização topológica. A configuração final da treliça depois da otimização é apresentada na Figura 4.5 e a Figura 4.6 e representa o comportamento da melhor solução ao longo das gerações para dois tamanhos de população diferentes.

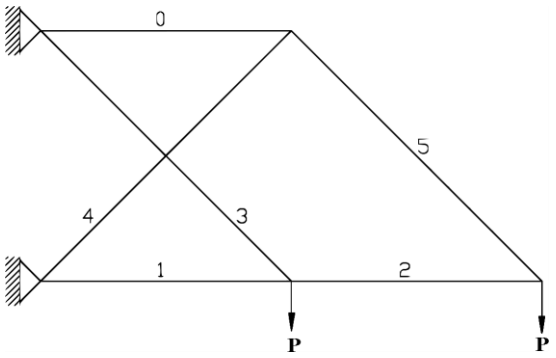


Figura 4.5 - Configuração final. Otimização topológica (Deb *et al*. 2001).

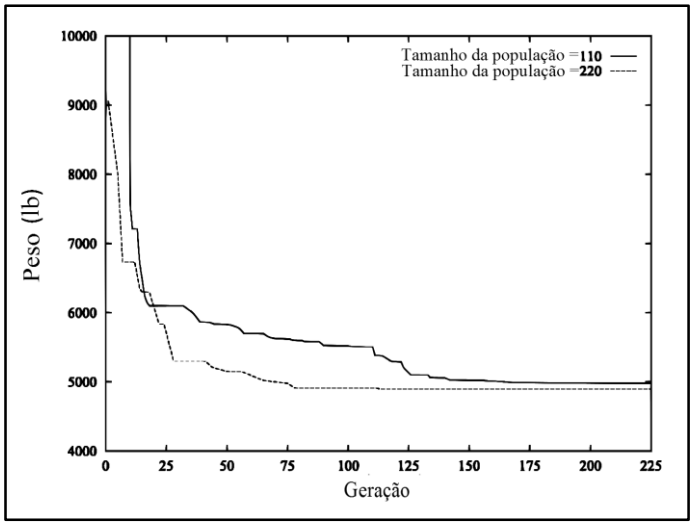


Figura 4.6 - Variação do peso ao longo das gerações (Deb *et al.* 2001).

Fadel *et al*. (2013) apresentam os algoritmos *firefly* como um método utilizado para otimizar simultaneamente o tamanho, forma e topologia de estruturas treliçadas. Este método, está inspirado no comportamento intermitente dos vaga-lumes. Basicamente vai comparando a luminosidade das soluções geradas aproximando as soluções que emitem menos luz para as que emitem um maior brilho, ou seja, trazer soluções de qualidade inferior a soluções de melhor qualidade. No trabalho, a minimização de peso estrutural é uma possível solução à otimização estrutural incluindo também sua função objetivo e as restrições de estabilidade, deslocamento, tensão e rigidez. Com o objetivo de fornecer uma base estatística para posterior comparação, este trabalho apresenta os resultados de mais de 100 gerações para cada exemplo. Desta forma, os valores médios e os desvios padrão são apresentados juntamente com os resultados ótimos. Na Figura 4.7, tem-se o histórico de evolução de uma treliça plana de 15 barras onde foi modificado o tamanho, forma, e a topologia. Neste exemplo, não se consideraram restrições de flambagem. A Figura 4.8 mostra a convergência do peso, do melhor individuo em 800 gerações.

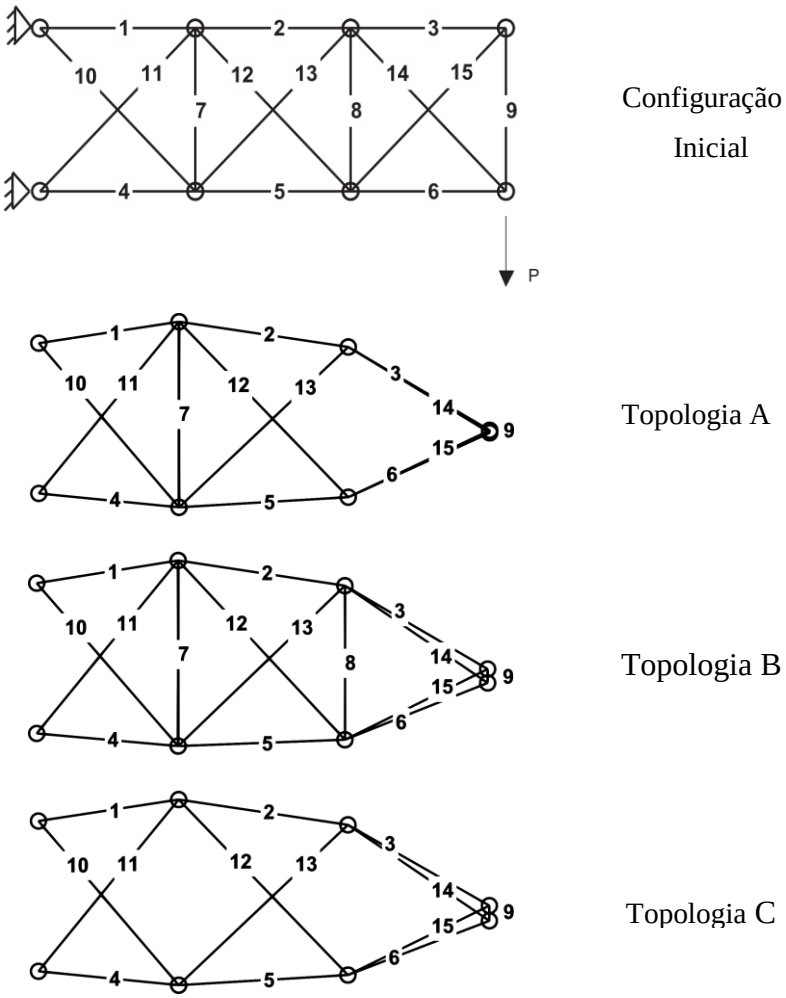


Figura 4.7 - Amostras de topologias selecionadas na otimização de tamanho, forma, e topologia da treliça plana de 15 barras (Fadel *et al.* 2013).

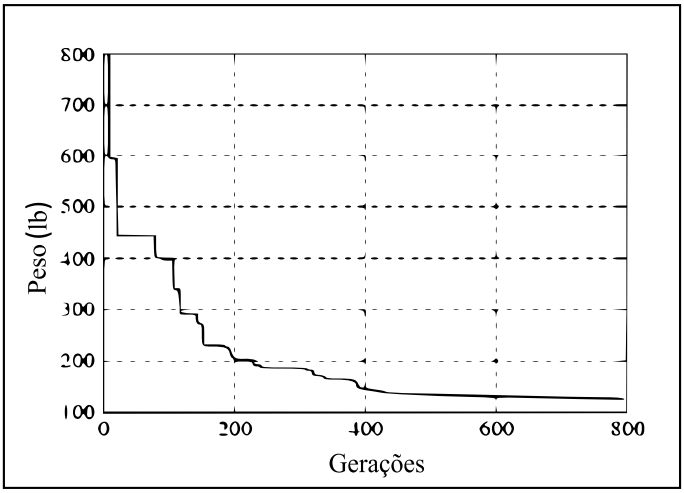


Figura 4.8 - Histórico de convergência do peso da treliça de 15 barras em 800 gerações (Fadel *et al.* 2013).

O trabalho desenvolvido por Krishnamoorthy *et al.* (2002) discute um projeto orientado a objetos com a implementação de uma biblioteca central composta por todos os operadores

genéticos. No trabalho é mostrado como as bibliotecas podem ser usadas para a otimização de tamanho de grandes treliças espaciais. As bibliotecas implementadas são testadas em um grande número de treliças espaciais previamente desenhadas e os resultados são comparados com os valores reportados no desenho inicial. Na Figura 4.9 mostra-se uma treliça na qual se realizou o processo de otimização de peso mudando as seções transversais das barras. A Figura 4.10 mostra a convergência do peso depois de 100 gerações de esta treliça com três mudanças de seção.

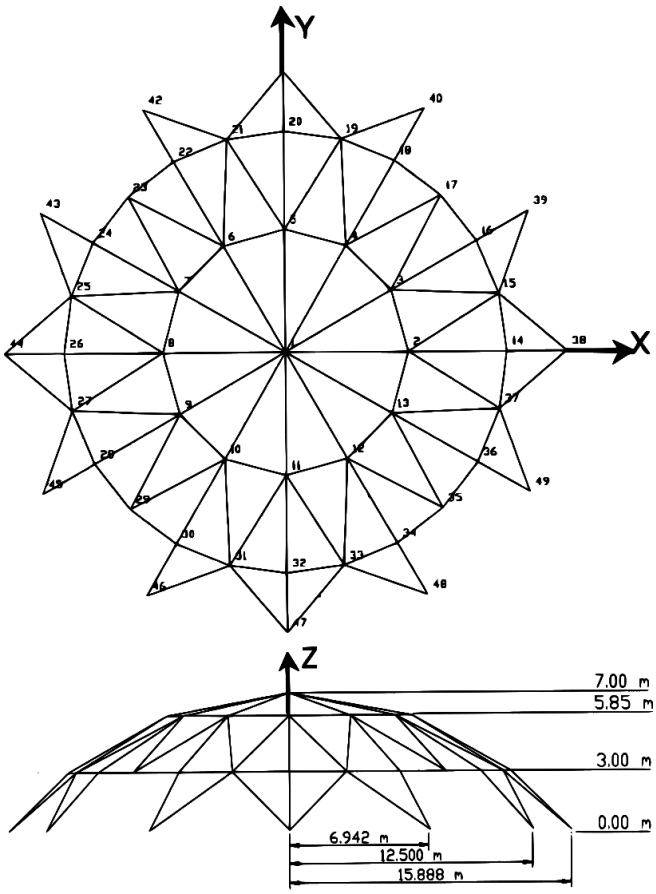


Figura 4.9 - Topologia da treliça espacial em forma de cúpula de 112 barras.(Krishnamoorthy *et al*. 2002)

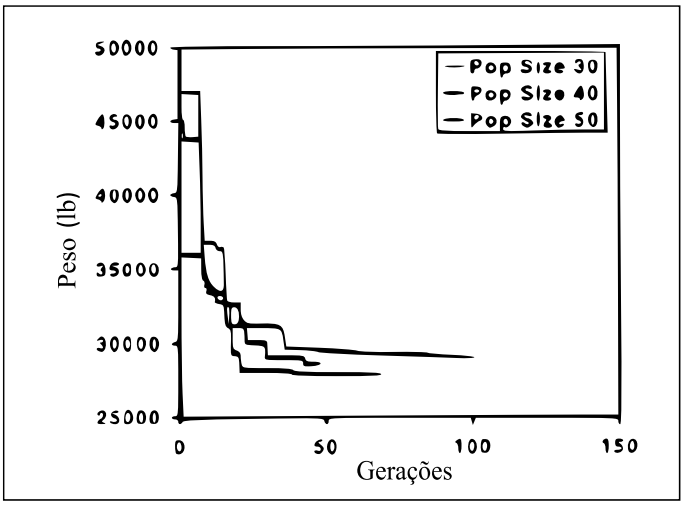


Figura 4.10 - Resultado do peso obtido depois de 100 gerações (Krishnamoorthy *et al.* 2002).

Kaveh e Shahrouzi (2008), em seu trabalho, propõem o uso de Algoritmos Genéticos combinado com a estratégia de colônia de formigas, a fim de provar indivíduos mais competitivos a partir de vários subespaços dentro do espaço de busca. O método proposto é aplicado na otimização simultânea como a dimensional e topológica. A codificação proposta por os autores, é inteira. São testados três exemplos onde o objetivo é a redução do peso final. Um destes exemplos é apresentado na Figura 4.11 composto de 6 andares e 2 vãos. Neste exemplo, otimiza-se as seções transversais. O edifício está submetido a carregamentos por cada andar e cargas de vento laterais. As restrições são de tensão e escoamento. A Figura 4.12 apresenta o comportamento do *fitness* ao longo de 100 gerações.

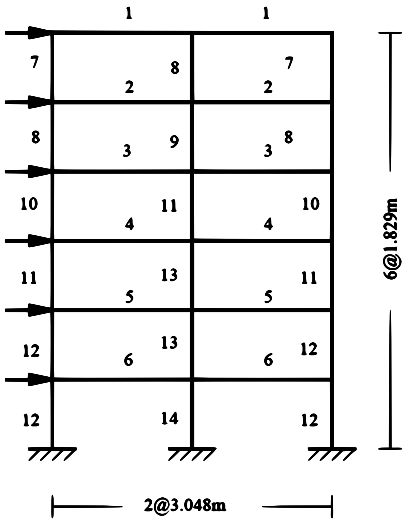


Figura 4.11 - Esquema do edifício de 6 andares e 2 vãos com carregamento lateral (Kaveh e Shahrouzi. 2008).

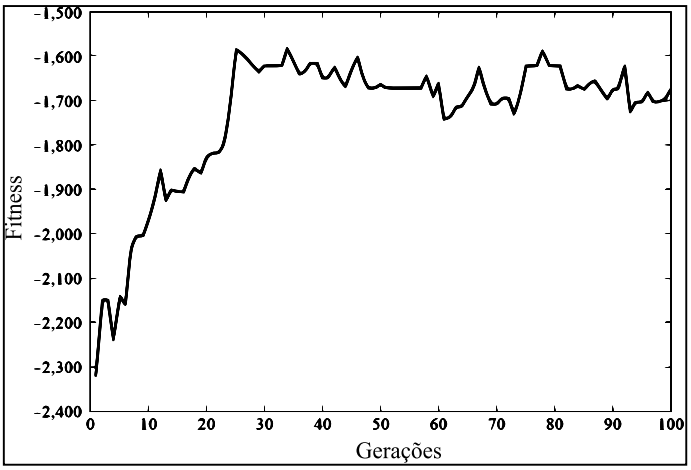


Figura 4.12 - Histórico do *fitness* para a otimização do edifício (Kaveh e Shahrouzi. 2008).

Ebadi *et al.* (2011) apresenta uma otimização simultânea, dimensional e de forma, para estruturas conectadas com pinos via Algoritmos Genéticos com codificação real. O objetivo desta otimização é encontrar o menor peso das estruturas considerando como varáveis as coordenadas dos nós e das seções transversais das barras. A metodologia empregada neste trabalho é testada com estruturas planas e espaciais de 18 e 25 barras conectadas por pinos. Neste trabalho, utilizou-se um novo operador de mutação chamado de *class mutation*. Esse operador associa as variáveis do projeto com as mesmas características dentro de uma classe. No processo de mutação, os grupos com a mesma classe serão mutados mantendo assim, as características impostas inicialmente. Por exemplo, no grupo de seções transversais, no processo de mutação, não poderá gerar genes com conteúdo das coordenadas dos nós, somente gera genes mutados que contem valores das áreas das seções. A Figura 4.13 apresenta a configuração inicial e final da estrutura, de um dos exemplos depois da otimização, onde se obteve uma redução de peso comparada com outros autores que estudaram esta mesma estrutura.

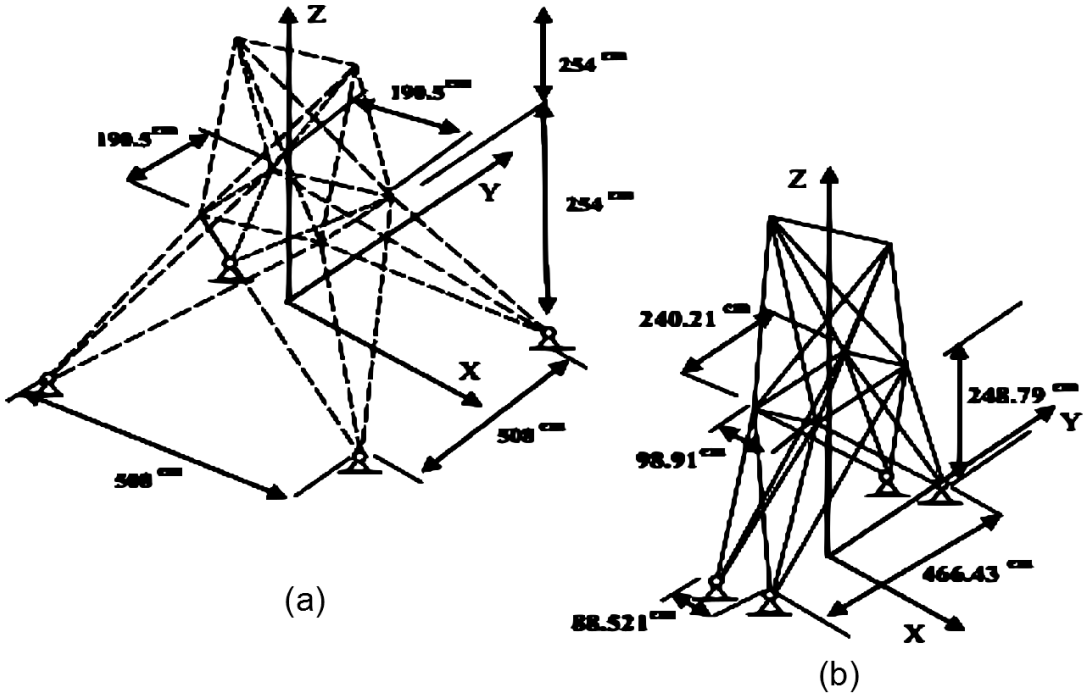


Figura 4.13 - (a) Configuração inicial da estrutura, (b) Configuração da estrutura depois da otimização (Ebadi *et al.* 2011).

5 OTIMIZAÇÃO VIA ALGORITMOS GENÉTICOS

Tendo em consideração os cinco aspectos fundamentais mencionados por Lemonge (1999), introduzidos no capítulo 3, apresenta-se na Figura 5.1 um esquema do princípio geral do algoritmo utilizado. Para dar início à otimização, o algoritmo começa gerando uma população inicial aleatória de possíveis soluções dentro do domínio das variáveis do projeto. Cada possível solução é dada por uma treliça. A partir da população inicial são realizados vários ciclos de evolução (gerações) visando obter melhores indivíduos. Durante este processo evolutivo, cada um dos indivíduos da população é avaliado em cada geração. Neste processo, atribui-se a cada indivíduo uma nota (*fitness*). Neste caso, quanto maior o valor do *fitness* melhor é a aptidão. A avaliação estrutural da treliça é realizada utilizando o MEF. A partir dos resultados obtidos, se comprovou quais são os valores de tensão e o deslocamento máximo.

No processo de seleção uma porcentagem dos indivíduos mais aptos é mantida, enquanto os outros são descartados. Os indivíduos mantidos podem sofrer modificações durante o processo evolutivo através dos operadores genéticos de mutação e cruzamento gerando descendentes para a próxima geração. Os critérios de parada, tais como o número máximo de gerações ou a satisfação da função objetivo, determinam se o algoritmo continua ou pelo contrário chega a seu fim, obtendo assim uma solução ótima.

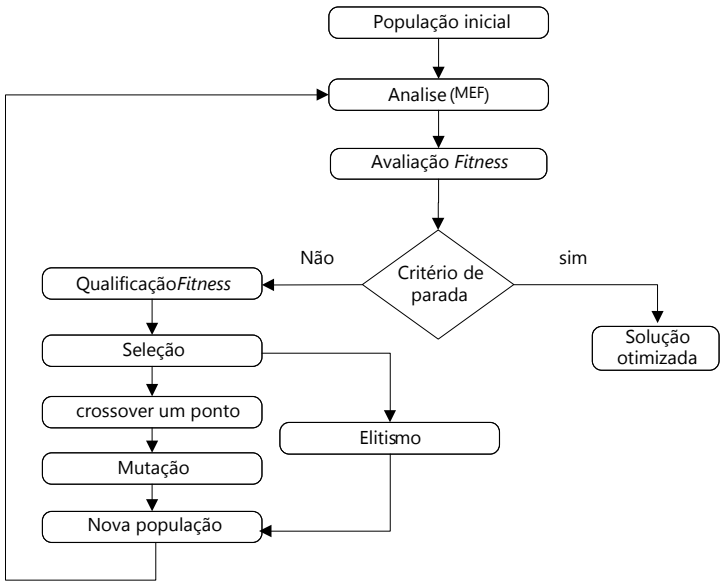


Figura 5.1 - Esquema de trabalho geral do AG e MEF.

A metodologia anteriormente descrita foi utilizada para dois tipos de otimização, a dimensional e de forma.

5.1 OTIMIZAÇÃO DIMENSIONAL

Neste tipo de otimização, buscou-se a modificação das seções transversais como objetivo final. A seguir é apresentada a implementação dos parâmetros para este tipo de otimização.

5.1.1 Codificação

Nesta etapa, o objetivo é a otimização de peso, através da modificação dimensional. Cada indivíduo é constituído por uma treliça. Cada treliça possui seu respectivo cromossomo e está composto por genes que tem a informação da área de cada uma das barras que compõem a estrutura. A equação apresentada a seguir mostra o vetor que representa o cromossomo das áreas de uma treliça (cr_a), onde A_i são as áreas das seções transversais dos elementos. Durante as análises, as áreas foram consideradas como variáveis reais contínuas sem nenhuma limitação exceto $A_i > 0$.

$$cr_A = [A_1, A_2, A_3, A_4, \cdots A_n] \tag{5.1}$$

5.1.2 Aptidão (*fitness*)

O objetivo da otimização é minimizar o peso total da estrutura satisfazendo as restrições iniciais do projeto como os deslocamentos e tensões máximas admissíveis. As tensões e deslocamentos fazem parte da avaliação estática das possíveis soluções. A aptidão (*fitness*) de cada indivíduo é determinada utilizando a função objetivo. Está função avalia os deslocamentos nodais e as tensões nas barras obtidos da análise estrutural da treliça. Para isto, o software de elementos finitos é utilizado. A seguinte equação apresenta a aptidão de cada indivíduo que é dada pelo inverso do peso da treliça. Desta forma, quanto menor o peso (W), maior a aptidão do indivíduo (f).

$$f = \frac{1}{W} \tag{5.2}$$

O valor do peso é dado por:

$$W = \sum_{i=1}^{n} \gamma A_i L_i \tag{5.3}$$

onde γ é o peso específico, A é a área das barras e L é o comprimento das barras.

Caso um indivíduo não atenda às restrições de tensão ou deslocamento o mesmo é penalizado com um valor de aptidão igual a zero.

5.1.3 Tamanho da população

Neste trabalho, foram testados diversos tamanhos das populações. Para os casos estudados considerou-se um valor de 90 indivíduos, por ser um valor próximo da média dos valores utilizados na bibliografia.

5.1.4 Operador de substituição

A fim de preservar os melhores indivíduos, utilizou-se o operador de elitismo. Isto significa que uma parte deles sobrevive inalterada para a próxima geração. Neste estudo, utilizou-se uma taxa de elitismo de $0{,}10$, ou seja, somente 10% dos indivíduos (os mais aptos) são escolhidos para continuar no processo. Neste caso, estes indivíduos são aqueles que vão a gerar novos indivíduos por cruzamento.

5.1.5 Tipo de cruzamento

Neste tipo de otimização, somente os indivíduos da elite foram utilizados para gerar novos indivíduos por cruzamento. Esta análise utilizou 10% do elitismo para gerar novos filhos até completar 100% da população. A escolha dos pais, dentro dos indivíduos do elitismo, foi aleatória. O tipo de cruzamento que foi utilizado é chamado de cruzamento por um ponto (ver Figura 3.2).

5.1.6 Taxa e fator de mutação

Neste estudo, utilizam-se duas taxas de mutação: uma taxa para a população e outra para o indivíduo. Tanto para a população como para o indivíduo utilizou-se a mutação uniforme, ou seja, existe uma probabilidade p_m de ser mutado. Assim, para o caso da população, o valor de p_m escolhido foi de 0.3, ou seja, somente 30% da população atual (10% de elitismo e 90% gerados por cruzamento) são mutados e essa mutação não inclui os indivíduos de elitismo. Para o caso do indivíduo escolhido aleatoriamente dentro dessa população, a taxa p_m foi de 0.5, ou seja, a metade dos genes dentro do cromossomo será mutado. Por exemplo, se um cromossomo possui 6 genes então somente 3 são modificados em cada geração. A Figura 5.2 mostra uma representação de um cromossomo onde apenas

três genes foram mutados. Uma vez que os genes a serem mutados são escolhidos, novos valores são atribuídos pela multiplicação de um fator *z* obtido de acordo com:

$$q = a + b \times r \tag{5.4}$$

onde a e b são valores reais de forma que $(a + b/2) = 1$ e r é um número aleatório que varia de zero a um. Isto garante que o valor médio de q seja igual a 1 e que seus valores extremos estejam entre $1 - b/2$ e $1 + b/2$.

Ao longo do estudo foram consideradas as seguintes expressões para o cálculo do fator de mutação baseadas na Eq. (5.4) são dadas por as seguintes equações:

$$q_1 = 0{,}3 + 1{,}4 \times r \tag{5.5}$$
$$q_2 = 0{,}4 + 1{,}2 \times r \tag{5.6}$$
$$q_3 = 0{,}5 + 1{,}0 \times r \tag{5.7}$$
$$q_4 = 0{,}6 + 0{,}8 \times r \tag{5.8}$$
$$q_5 = 0{,}7 + 0{,}6 \times r \tag{5.9}$$
$$q_6 = 0{,}8 + 0{,}4 \times r \tag{5.10}$$
$$q_7 = 0{,}9 + 0{,}2 \times r \tag{5.11}$$

Pode se observar que na Eq. (5.7) a variação de um gene pode ser de até 50% enquanto que na Eq. (5.11) a variação pode ser de até 10%. A Figura 5.3 apresenta um exemplo onde foi utilizada a Eq. (5.7), onde, por sua vez, se observa a mudança do 50% dos genes do cromossomo.

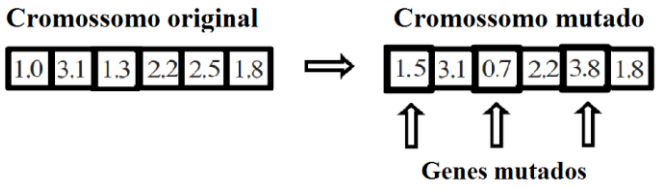


Figura 5.2 - Mutação uniforme dentro do cromossomo.

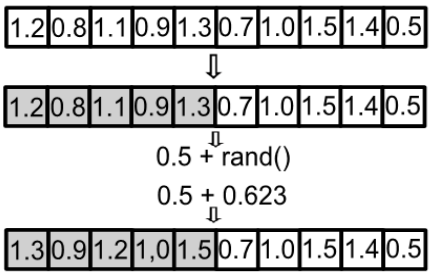


Figura 5.3 - Exemplo aplicando o fator de mutação.

5.2 OTIMIZAÇÃO DE FORMA

Neste tipo de otimização, busca-se a melhor configuração da estrutura, tendo como variável de projeto as coordenadas dos pontos nodais. A estratégia implementada no processo de otimização é a mesma abordada na otimização dimensional, salvo algumas modificações. A configuração dos parâmetros utilizados neste tipo de otimização é apresentada a seguir.

5.2.1 Codificação

Para a codificação no caso da otimização de forma se considerou um cromossomo (cr_n), composto por números reais que contém a posição dos nós nas treliças como o representado na seguinte equação.

$$cr_n = [\, x_1, y_1, z_1,\ x_2, y_2, z_2, \cdots\ x_i, y_i, z_i] \tag{5.12}$$

Onde $x_1, y_1, z_1,\ x_2, y_2, z_2, \cdots\ x_i, y_i, z_i$ representam as coordenadas nodais da treliça nas três direções.

5.2.2 Aptidão (*fitness*)

Para o valor do *fitness*, neste tipo de otimização, foram consideradas as seguintes equações:

$$f_1 = \frac{1}{\delta_{máx}} \tag{5.13}$$

$$f_2 = \frac{\alpha}{\frac{W}{W_l}} + \frac{\beta}{\frac{\delta_{máx}}{\delta_l}} \tag{5.14}$$

onde $\alpha + \beta = 1$, $\delta_{máx}$ é o deslocamento máximo,W é o peso da treliça, W_l é o peso limite e δ_l é o deslocamento limite.

A primeira expressão dada pela Eq. (5.13) permite encontrar o menor deslocamento ao longo das gerações. O uso de α e β na Eq. (5.14) permite parametrizar e dar à formulação uma importância relativa. Por exemplo, se o objetivo é otimizar o peso da estrutura, atribui-se um valor de β igual a zero e α igual a um, e pelo contrário, se a otimização é o deslocamento final da estrutura, atribui-se para α um valor de zero e β um valor de um. Também é possível utilizar esta equação para otimização simultânea, atribuindo valores α e β tais que o somatório deles seja um, por exemplo, para valores de α e β iguais a 0,5 a otimização será feita em igual proporção para o peso e deslocamento final da estrutura.

5.2.3 Tamanho da população

O tamanho da população variou dependendo do exemplo testado. Foram testadas diversas populações até se obter a convergência do *fitness*. Na maioria dos casos adotou-se uma população de 90 indivíduos, por ser um valor próximo da média dos valores usados na bibliografia.

5.2.4 Operador de substituição

Neste estudo, utilizou-se o operador de elitismo. Foram feitas duas análises com uma taxa de elitismo de 0,10. Em uma das análises somente 10% dos indivíduos com melhor qualificação *fitness* são escolhidos para gerar novos filhos por cruzamento e na outra análise 10% da população com melhor aptidão continua sem ser modificada.

5.2.5 Tipo de cruzamento

Utilizou-se cruzamento por um ponto para todos os casos. Para alguns casos, foram feitas duas análises com diferentes formas de seleção, assim:

> **Seleção tipo 1:** esta análise utilizou 10% do elitismo para gerar novos filhos até completar 100% da população. A escolha dos pais foi aleatória com distribuição de probabilidade uniforme dentro da população constituída por os indivíduos de elitismo;

Seleção tipo 2: nesta análise, utilizou-se toda a população, sem incluir aos indivíduos de elitismo, para gerar novos indivíduos no processo de cruzamento. Para a escolha dos indivíduos, utilizou-se o operador de seleção por roleta explicado no item 3.2.4.

5.2.6 Taxa e fator de mutação

Neste ponto, é escolhida uma taxa de mutação de 0,3 (30% da nova população), depois do cruzamento, que não inclui 10% dos indivíduos de elitismo. Os indivíduos escolhidos randomicamente são mutados como explicado a seguir:

Uma vez escolhidos os cromossomos somente 50% dos genes que contém os cromossomos são modificados, ou seja, a metade dos genes do cromossomo e multiplicado por um fator. O fator considerado ao longo do estudo é baseado na Eq. (5.4) e é dado por:

$$q = (-1{,}0 + 2{,}0 \times r) \times \delta \tag{5.15}$$

onde δ é o comprimento máximo a ser adicionado ou subtraído a uma coordenada, r é um número aleatório que varia de zero a um. Para todos os casos adoptou-se um valor δ de 0,02 m.

Finalmente, os novos indivíduos que sofreram mutação são restituídos à 90% da população e, posteriormente, são adicionados à elite (10%), formando, assim, 100% da nova população.

6 ESTUDOS DE CASOS

Neste capítulo, serão apresentados os estudos realizados para calibração dos parâmetros e exemplos básicos resolvidos. Os exemplos escolhidos são clássicos da literatura, a fim de validar a metodologia. Os exemplos têm como critério a minimização da função objetivo obedecendo às restrições iniciais impostas como são deslocamentos máximos nos nós e, em alguns casos, as tensões máximas nas barras. Para alguns exemplos, as implementações foram realizadas com a linguagem de programação *Python* e *Julia*. A seguir se descreve as características destas linguagens.

6.1 SOFWARE DE ANÁLISE

A implementação do AG e do MEF foi feita com *Python* e *Julia*. A primeira é uma linguagem computacional, com uma sintaxe clara e simples de alto nível criada em meados dos anos 90. Esta linguagem é multiparadigma, já que suporta orientação a objetos e programação imperativa. Dispõe de uma licença de código aberto, denominada *Python Software Foundation License.* Esta linguagem tem uma grande quantidade de bibliotecas de uso geral. Por outra parte, linguagem *Julia* é multiparadigma e combina características de programação imperativa, funcional e orientada a objetos. *Julia* tem licença de software livre. Esta linguagem mantém a facilidade e expressividade do MATLAB para computação numérica de alto nível, mas ultrapassa as limitações comparadas a uma linguagem de programação de propósito geral.

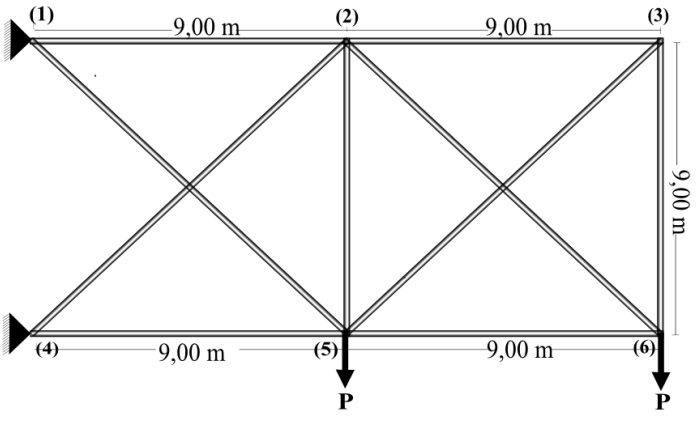
Foram realizadas duas implementações utilizando as linguagens descritas anteriormente. A implementação inicial foi feita em *Python* e demandou longos tempos de execução. Por esta razão, foi feita uma segunda implementação na linguagem *Julia.* Para cada implementação, foi utilizada uma biblioteca de elementos finitos correspondente, *Pyfem* e *FEMlab*, ambas utilizadas para resolver estaticamente a estrutura.

O pós-processamento dos resultados obtidos depois da otimização foi feito no *software Paraview*® desenvolvido pela empresa *Kitware, Inc*. Este é um software livre (*open-source),* multiplataforma para análise de dados, visualização e exploração de dados de forma interativa em 3D.

6.2 OTIMIZAÇÃO DIMENSIONAL

O objetivo desde tipo de otimização, é encontrar o conjunto de áreas que minimize o peso da estrutura. A seguir é apresentado um caso, estudado por muitos autores, implementado com as duas linguagens *Python* e *Julia.*

A fim de testar o procedimento de otimização foi utilizada uma treliça plana amplamente analisada na bibliografia. Este exemplo é simples, todavia representativo e devido ao seu baixo custo computacional, permitiu realizar alguns experimentos numéricos visando o comportamento e calibração dos parâmetros do algoritmo. Autores como Deb e Gulati (2001), Kaveh e Shahrouzi (2008), Nanakorn *et al.* (2001) e Razvan e Lucian (2014) fazem uma análise desta treliça. Nesta estrutura, foram aplicados os operadores genéticos descritos no capítulo anterior, em uma população de 90 indivíduos, otimizados ao longo de um máximo de 800 gerações. O esquema do problema está apresentado na Figura 6.1 onde as dimensões da treliça são $18\ \mathrm{m} \times 9\ \mathrm{m}$. As barras são de alumínio com peso específico de $25{,}9\ \mathrm{kN/m^3}$. A treliça foi submetida à restrição de deslocamento vertical máxima do nó 6 de $50{,}8\ mm$ e a tensão axial foi limitada a $130\ \mathrm{N/mm^2}$, de acordo com os autores que realizaram o mesmo estudo.



c

Figura 6.1 - Esquema estático. Treliça 10 barras. 6 nós.

Tabela 6.1 - Propriedades do material e restrições (Treliça de 10 barras)

Propriedades	Valores
Modulo de Young (E)	$6{,}89\ x\ 10^7$kPa
Peso específico (γ)	$25{,}9$ kN/m^3
Carga (P)	450 kN
Tensão axial admissível ($\sigma_{máx}$)	130 N/mm^2
Deslocamento nodal admissível nó 6 ($d_{máx}$)	$50{,}8$ mm

Tabela 6.2 - Parâmetros para a treliça de 10 barras (Variáveis contínuas)

Parâmetros	
População (n_{pop})	90
Probabilidade de mutação (p_c)	0,3
Probabilidade de elitismo (p_m)	0,1
Operador de cruzamento	1 (um) ponto
Critério de parada	Menor peso encontrado em 800 gerações

6.3 ESTUDO DE CASO - IMPLEMENTAÇÃO EM *PYTHON*

Para dar início à otimização, utilizando Algoritmos Genéticos, a população inicial foi constituída através de cromossomos com genes aleatórios. Três estudos foram realizados para avaliar a convergência. Na primeira, duas expressões para o fator de mutação foram testadas. O segundo estudo analisa a variação da taxa de mutação dos indivíduos na população. Finalmente o terceiro estudo analisa a variação na taxa de elitismo de indivíduos entre gerações.

6.3.1 Estudo do fator de mutação

Neste estudo, foi considerada uma taxa de elitismo de 0,1 e uma taxa de mutação constante de 0,3, i.e. 10% dos indivíduos são mantidos e 30% dos indivíduos estão sujeitos à mutação. Foram realizadas duas análises de otimização variando o fator de mutação de acordo com as Eqs. (5.7) e (5.11). A Figura 6.2, mostra os resultados das duas análises onde a aptidão do melhor indivíduo é ilustrada ao longo das gerações. Pode-se observar que o uso da Eq. (5.7) apresenta melhores valores de aptidão mostrando uma evolução mais rápida nas primeiras gerações. A evolução utilizando esta equação tem mais regiões de estagnação onde a variação da aptidão é mínima; mas também existem pontos onde a

aptidão melhora subitamente. Já para a Eq. (5.11), a evolução é menos rápida nas primeiras gerações, mas apresenta uma tendência sempre crescente. Verifica-se que o resultado obtido com a Eq. (5.7) alcança um valor de *fitness* maior em menos gerações, enquanto que com a Eq. (5.11) a evolução é um pouco mais lenta e precisa de mais gerações para obter valores semelhantes para o *fitness*.

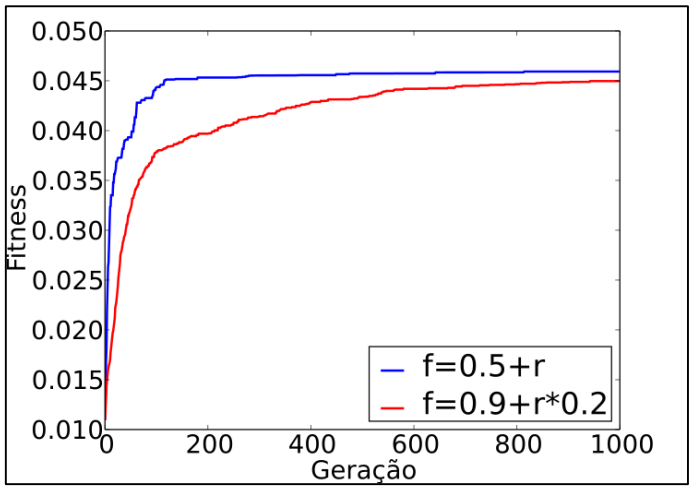


Figura 6.2 - Variação do valor do fitness para diferentes fatores de mutação.

6.3.2 Estudo da variação da taxa de elitismo

Para este estudo, foram consideradas diferentes taxas de elitismo para uma população de 90 indivíduos durante 100 e 800 gerações. A taxa de mutação foi deixada constante e igual a 30%. As taxas de elitismo utilizadas foram de 5%, 10%, 20%, 30% e 50%. Neste estudo em particular, dado que os indivíduos de elitismo são os únicos que geram novos filhos, não se considerou uma taxa de elitismo de 0%. A Figura 6.3, mostra o processo de evolução para os quatro casos em 800 gerações, por outro lado a Figura 6.4, apresenta o comportamento da evolução apenas nas primeiras 100 gerações. Pode-se observar que a taxa de elitismo não altera consideravelmente os resultados exceto para valores muito altos. Contudo, a melhora do *fitness* foi um pouco mais rápida para uma taxa de 10%. O melhor comportamento também foi obtido para 10% de elitismo ao final das 800 gerações.

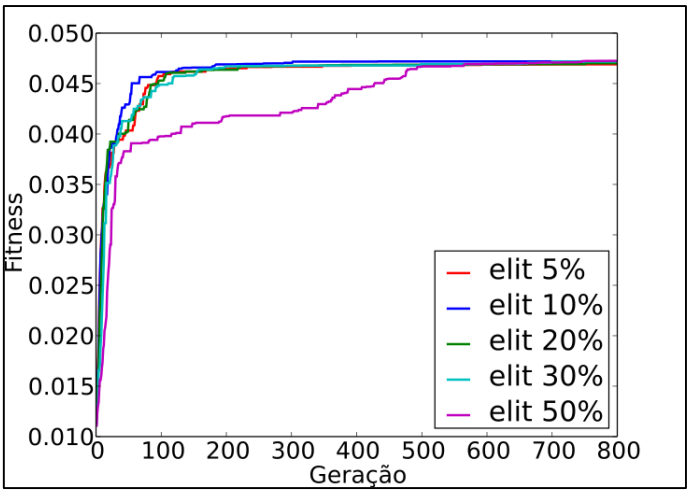


Figura 6.3 - Valores do *fitness* para diferentes porcentagens de elitismo com mutação constante de 30% em 800 gerações.

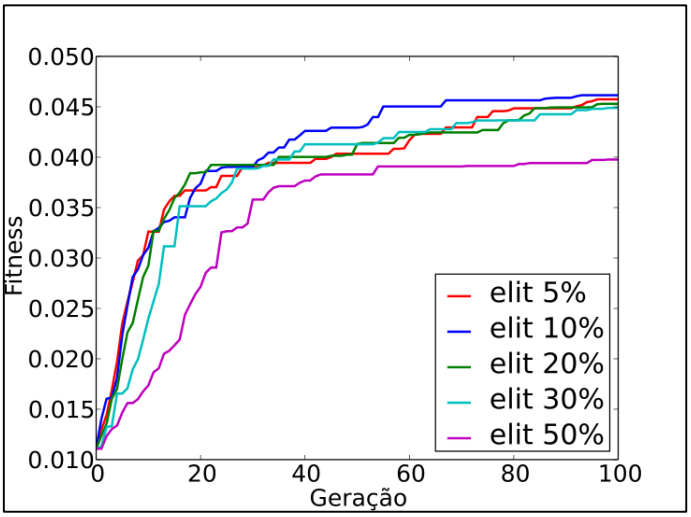


Figura 6.4 - Valores do *fitness* para diferentes porcentagens de elitismo com mutação constante de 30% em 100 gerações.

6.3.3 Estudo da variação da taxa de mutação

Este estudo considerou taxas de mutação de 10%, 20%, 30% e 40% em uma população de 90 indivíduos durante 100 e 800 gerações. A taxa de elitismo foi mantida constante e igual a 10%. Os resultados obtidos são apresentados na Figura 6.5, onde é apresentado o comportamento do melhor indivíduo em 800 gerações. A Figura 6.6, apresenta apenas as primeiras 100 gerações. Nos quatro casos, nota-se uma variação leve na velocidade de evolução, aparentemente mais rápida quanto maior a taxa de mutação. Ao final das gerações, o melhor indivíduo foi obtido para uma taxa de mutação de 30%.

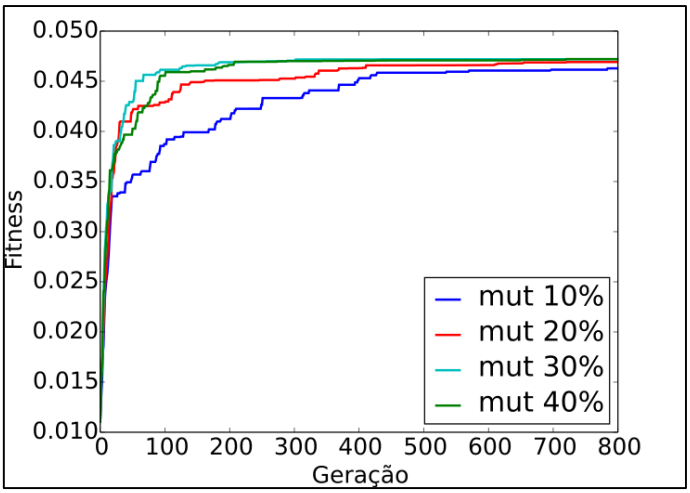


Figura 6.5 - Valores do *fitness* para diferentes porcentagens de mutação com elitismo constante de 10% em 800 gerações.

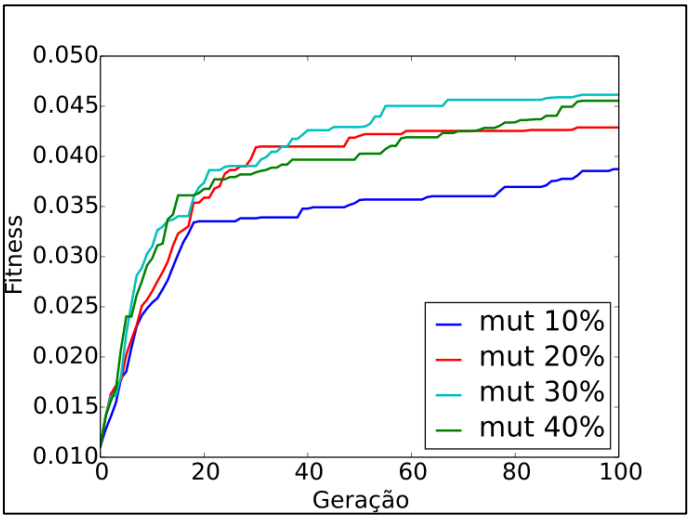


Figura 6.6 - Valores do *fitness* para diferentes porcentagens de mutação com elitismo constante de 10% em 100 gerações.

6.3.4 Análise comparativa com outros autores

Com os resultados obtidos dos três estudos apresentados, foi realizada uma análise de otimização utilizando as taxas que forneceram melhores resultados. Desta forma, foi utilizado a Eq. (5.7) para o fator de mutação, e taxas de 10% e 30% de elitismo e mutação, respectivamente. A Figura 6.7 mostra o comportamento do *fitness* do melhor indivíduo ao longo das gerações de acordo com a Eq. (5.2). Já a Figura 6.8, apresenta o valor do peso do melhor indivíduo ao longo das gerações sendo este valor o inverso da Eq. (5.2). Os resultados obtidos estão em concordância como os obtidos na literatura uma vez que o peso final é próximo ao obtido por outros pesquisadores que utilizaram AGs, como mostra a Tabela 6.7. Somente os autores Deb e Gulati (2001) apresentam um peso menor que os

obtidos neste estudo, entretanto eles utilizaram uma população muito maior e também um número maior de gerações.

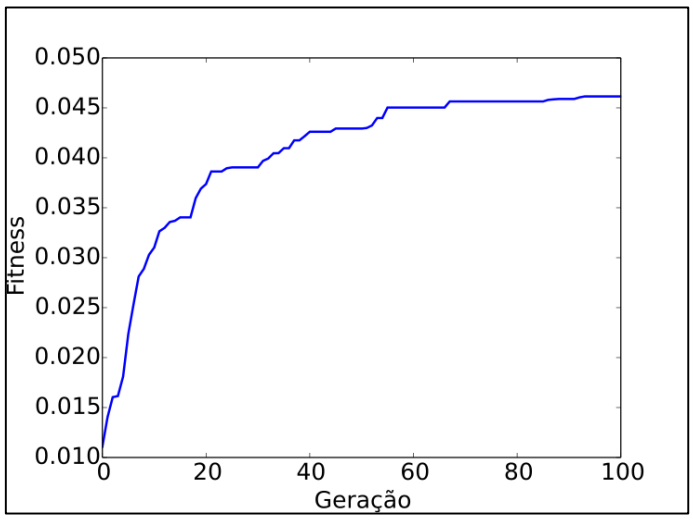


Figura 6.7 - Variação do valor do fitness para o melhor indivíduo (elitismo de 10% e mutação de 30%).

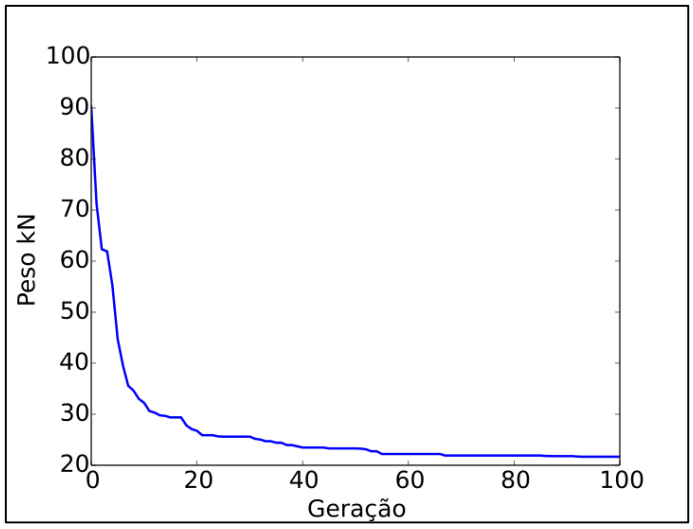


Figura 6.8 - Minimização do peso da treliça de 10 barras.

Tabela 6.3 - Melhor solução encontrada para o problema de referência utilizando Python.

Artigo	População	Gerações	Peso total
Deb e Gulati (2001)	450	189	21.06 kN
Este estudo (elit. 10%, mut. 30%)	90	100	21.67 kN
Kaveh e Shahrouzi (2008)	50	100	22.06 kN
Nanakorn *et al.* (2001)	70	100	22.08 kN
Razvan e Lucian (2014)	100	100	22.57 kN

Os resultados encontrados em este estudo apresentam diversos comportamentos na evolução de acordo com os diferentes parâmetros utilizados. Isto pode ser visto especialmente nos gráficos que mostram resultados até as 100 gerações. Foi observado que taxas de 10% e 30% de elitismo e mutação, respectivamente, favorecem o processo de evolução especialmente nas primeiras gerações. Os resultados obtidos no exemplo de referência estiverem muito próximos à comparação com outros autores, devido a que permitiu obter melhores resultados com o mesmo número de gerações.

6.4 ESTUDO DE CASO - IMPLEMENTAÇÃO EM *JULIA*

O mesmo exemplo anterior foi testado com os mesmos parâmetros e características da Tabela 6.1 e Tabela 6.2, mas neste caso, utilizando a linguagem de programação *Julia*.

6.4.1 Estudo do fator de mutação

Ao longo deste estudo, foram consideradas mais expressões para o cálculo do fator de mutação baseadas na Eq. (5.4) que estão dadas pelas equações (5.5) à (5.11). Os resultados obtidos são apresentados na Figura 6.9 e na Tabela 6.4, onde se mostra o comportamento do melhor indivíduo em 100 gerações com os diferentes fatores de mutação.

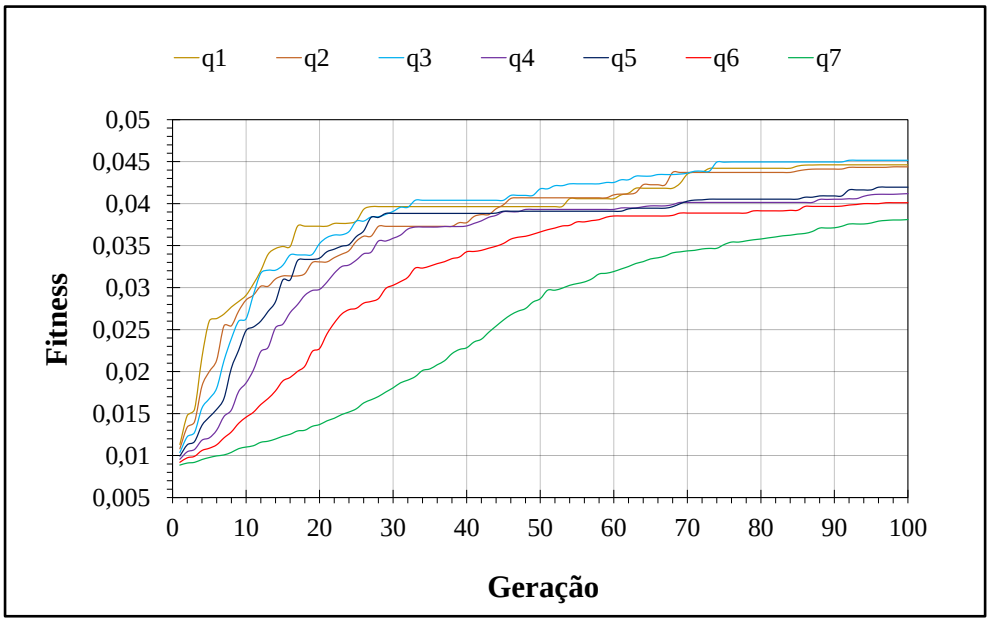


Figura 6.9 - Variação dos fatores de mutação.

Tabela 6.4 - Resultados obtidos no estudo do fator de mutação

	Fatores						
Geração	q_1	q_2	q_3	q_4	q_5	q_6	q_7
100	22,4117	22,5325	22,1431	24,2782	23,8336	24,9329	26,2426

O melhor resultado foi obtido com o fator $q_3 = 0{,}5 + 1{,}0 \times r$. Onde se observa que todos os fatores tem um comportamento similar ao longo das gerações à exceção do fator q_7, onde o comportamento segue a mesma tendência, mas não se obtém valores bons do *fitness*. Ao longo das gerações, para o fator q_3, o comportamento melhora a partir da geração 34 comparado com os outros fatores, e finalmente, obtém a melhor nota de aptidão.

6.4.2 Estudo da variação da taxa de mutação

Neste estudo, consideraram-se mutações de 10%, 20%, 30%, 40%, 50% e 60% numa população de 90 indivíduos durante 100 e 800 gerações. A taxa de elitismo foi mantida constante e igual a 10%. A Figura 6.10 apresenta o comportamento do *fitness* do melhor indivíduo com as diferentes porcentagens de mutação para 800 gerações. Para uma melhor visualização do comportamento só nas primeiras 100 gerações e apresentada a Figura 6.11. Finalmente os resultados obtidos neste estudo são apresentados na Tabela 6.5.

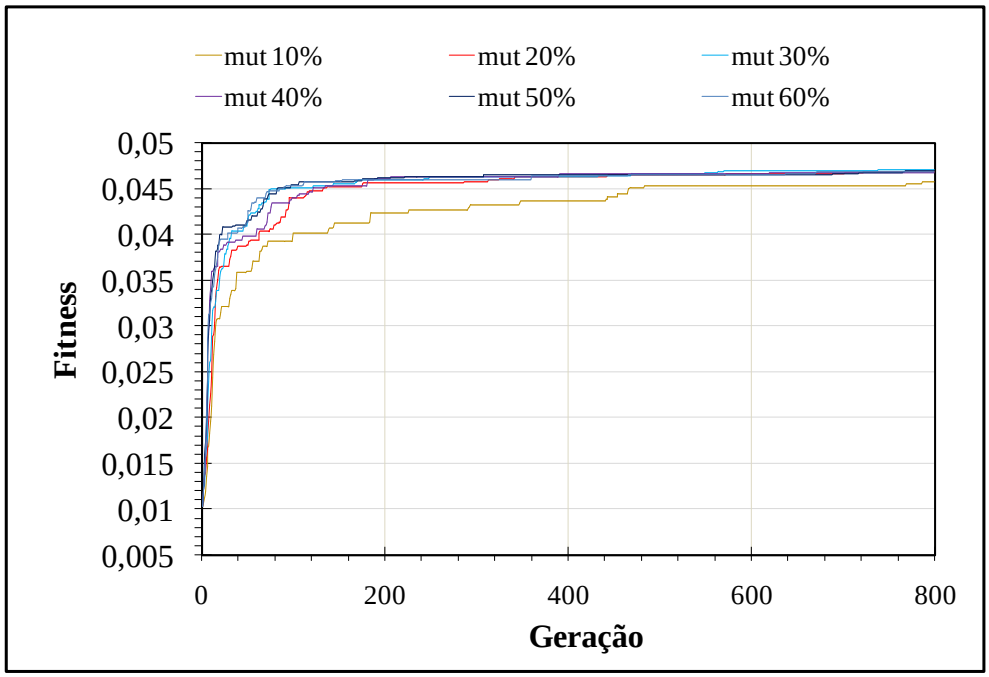


Figura 6.10 Variação da taxa de mutação em 800 gerações

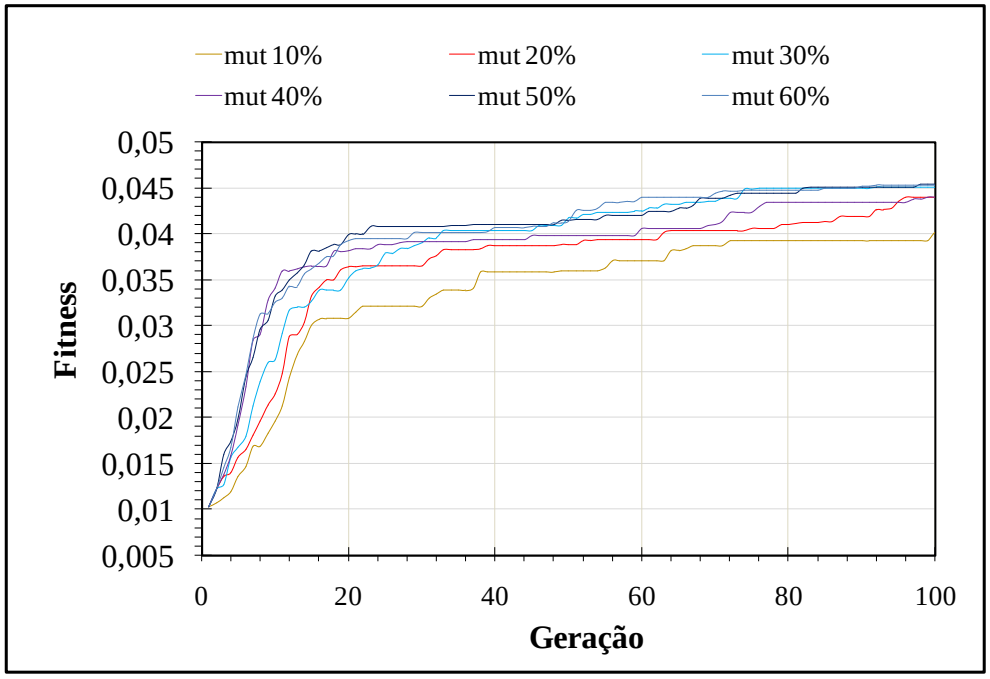


Figura 6.11 - Variação da taxa de mutação em 100 gerações

Tabela 6.5 - Resultados obtidos no estudo da taxa de mutação

	Peso					
Gerações	mut 10%	mut 20%	mut 30%	mut 40%	mut 50%	mut 60%
800	21,8487	21,3628	21,2591	21,3814	21,2817	21,3601
100	24,9161	22,7010	22,1431	22,7092	22,0322	22,0380

Nos seis casos, nota-se uma variação rápida do *fitness* nas primeiras 20 gerações, mas com o passo das gerações a velocidade de evolução fica mais leve; aparentemente esta evolução é mais rápida quanto maior a taxa de mutação. Ao fim das gerações, o melhor indivíduo obtido em 800 gerações foi para uma taxa de mutação de 30%, mas para 100 gerações o melhor indivíduo foi obtido com um fator de mutação 50%.

6.4.3 Estudo da variação da taxa de elitismo

Para este estudo, foram consideradas diferentes taxas de elitismo para uma população de 90 indivíduos durante 800 gerações. A taxa de mutação foi deixada constante e igual a 30%. As taxas de elitismo utilizadas foram de 5%, 10%, 20%, 30%, 40% e 50%. A Figura 6.12 mostra o processo de evolução do melhor indivíduo para os quatro casos em 800 gerações, por outro lado a Figura 6.13 apresenta o comportamento da evolução apenas nas primeiras 100 gerações. A Tabela 6.6 apresenta os resultados do peso para este estudo.

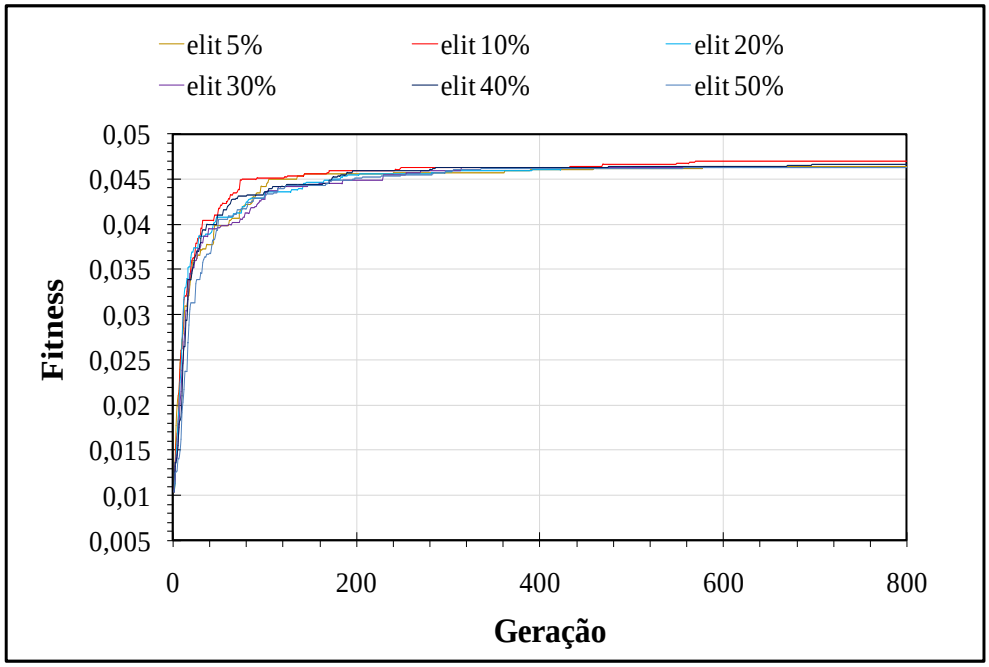


Figura 6.12 - Variação da taxa de elitismo em 800 gerações

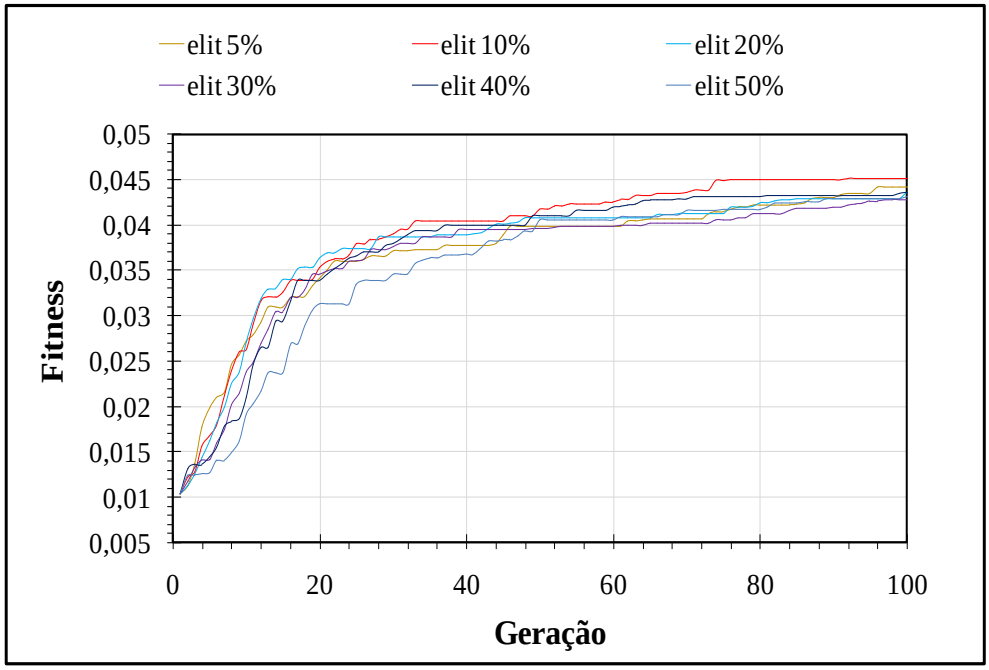


Figura 6.13 - Variação da taxa de elitismo em 100 gerações

Tabela 6.6 - Resultados obtidos no estudo da taxa de elitismo

	Peso					
Gerações	elit 5%	elit 10%	elit 20%	elit 30%	elit 40%	elit 50%
800	21,5321	21,2591	21,6216	21,5768	21,4617	21,5898
100	22,6043	22,1431	22,9477	23,3426	22,9332	23,1772

Pode-se observar que a taxa de elitismo não altera consideravelmente os resultados. Esta tem pouco efeito nos resultados após as primeiras 100 gerações. Na Figura 6.12 onde se observa o comportamento do melhor indivíduo ao longo de 100 gerações pode-se ver que o fitness do melhor indivíduo para uma taxa de elitismo de 10% obtém a melhor qualificação, mas na Figura 6.13 pode-se observar que este indivíduo ainda pode melhorar com o passo das gerações. Contudo, a melhora do *fitness* foi um pouco mais rápida para uma taxa de 10%.

Com os resultados obtidos dos três estudos apresentados na linguagem *Julia* foi realizada uma análise de otimização utilizando as taxas que forneceram os melhores resultados nas primeiras 100 gerações. Desta forma, foi utilizado o fator q_3 apresentado na Eq. (5.7), e taxas de 10% e 30% de elitismo e mutação, respectivamente. Na Figura 6.14 é apresentada a configuração final da treliça com suas respectivas áreas depois da otimização. Pode-se observar que as barras com maior seção transversal caracterizam uma treliça mais simples, sugerindo que as outras barras com seções transversais menores possam ser descartadas em um estudo de otimização topológica. A Figura 6.15 mostra o comportamento do *fitness* do melhor indivíduo ao longo das gerações de acordo com a Eq. (5.2). Já na Figura 6.16 é apresentado o valor do peso do melhor indivíduo ao longo das gerações sendo este valor o inverso da Eq. (5.2). A Tabela 6.7 apresenta a comparação feita com outros autores e com o resultado obtido no estudo anterior, feito na linguagem *Python.* Mesmo assim, os autores Deb e Gulati (2001) continuam apresentando um peso menor com uma população muito maior e também com um número maior de gerações.

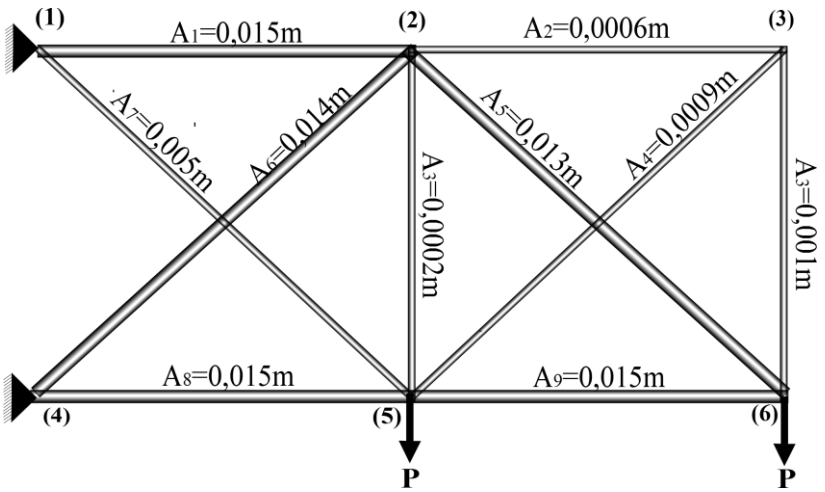


Figura 6.14 – Áreas finais das seções transversais da treliça de 10 barras.

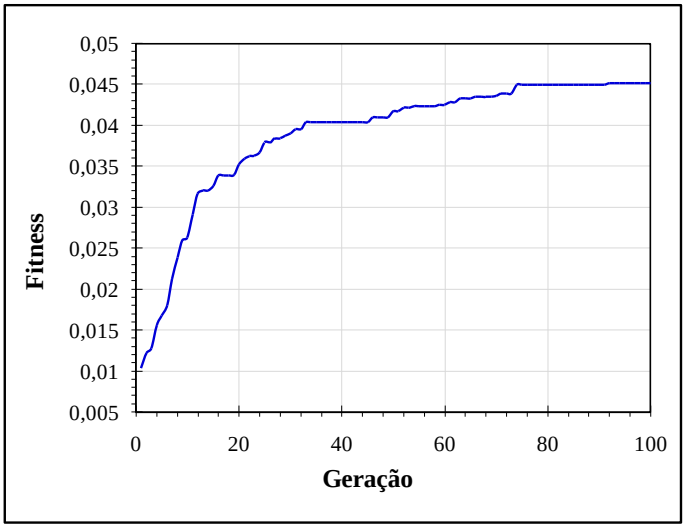


Figura 6.15 - Variação do valor do *fitness* para o melhor indivíduo (elitismo de 10% e mutação de 30%).

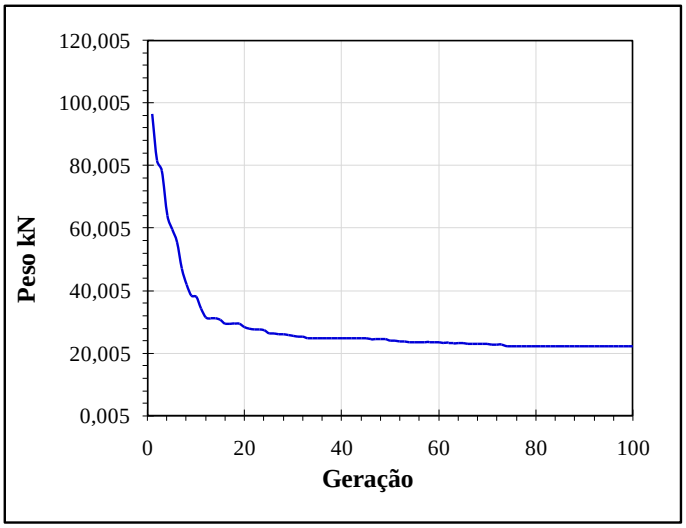


Figura 6.16 - Minimização do peso da treliça de 10 barras

Tabela 6.7 - Melhor solução encontrada para o problema de referência utilizando *Julia*.

Artigo	População	Gerações	Peso total
Deb e Gulati (2001)	450	189	21,06 kN
Este estudo (elit. 10%, mut. 30%)	90	100	21,14 kN
Estudo *Python* (elit. 10%, mut. 30%)	90	100	21,67 kN
Kaveh e Shahrouzi (2008)	50	100	22,06 kN
Nanakorn *et al.* (2001)	70	100	22,08 kN
Razvan e Lucian (2014)	100	100	22,57 kN

Foi observado que com as mesmas taxas de elitismo, mutação e as mesmas características do exemplo estudado com *Python*, os resultados obtidos com a implementação na

linguagem *Julia* permitiu obter melhores resultados. Isso se deve as diferenças das funções randômicas que utilizam estas linguagens.

6.5 OTIMIZAÇÃO DE FORMA

Neste tipo de otimização, foram testados 4 exemplos. Para os três primeiros testes foram usadas duas análises, aplicando a seleção tipo 1 e seleção tipo 2, explicados no item 5.2.5. A implementação destes exemplos foi feita na linguagem *Julia*. Os valores dos parâmetros apresentados na Tabela 6.8 são mantidos constantes no processo de otimização. Os cromossomos foram codificados como indicado no item 5.2.1.

Tabela 6.8 - Parâmetros utilizados na otimização de forma

Parâmetros	
Probabilidade de mutação (p_c)	0,3
Probabilidade de elitismo (p_m)	0,1
Operador de cruzamento	1 (um) ponto
Número de indivíduos	90

6.5.1 Caso 1: treliça plana de 10 barras

Foi testado o exemplo apresentado na otimização dimensional, mas esta vez, para otimização de forma com o objetivo de minimizar o peso da estrutura. Neste estudo considerou-se o *fitness* expressado na Equação (5.14), os valores de α e β foram de um e zero respectivamente. A área é mantida constante durante o estudo, sendo $A = 10\ cm^2$. Na Figura 6.17 é apresentado o *fitness* dos melhores indivíduos durante o processo de evolução em 100 gerações para as duas análises, usando a seleção tipo 1 e a seleção tipo 2. Pode-se observar que o uso da seleção tipo 1 apresenta melhores valores de aptidão mostrando uma evolução mais rápida nas primeiras gerações. Já para a seleção tipo 2, a evolução é menos rápida e tem mais regiões de estagnação durante a evolução. Verifica-se que os dois tipos de seleção apresentam uma tendência sempre crescente e uma em relação à outra estão muito perto de obter o mesmo valor final de *fitness*. Pode-se observar na Figura 6.18 a evolução do peso dos melhores indivíduos nas 100 gerações para as duas análises de otimização. Por último, na Figura 6.19 é apresentada a evolução da treliça de 10 barras em 100 gerações para a análise de seleção tipo1.

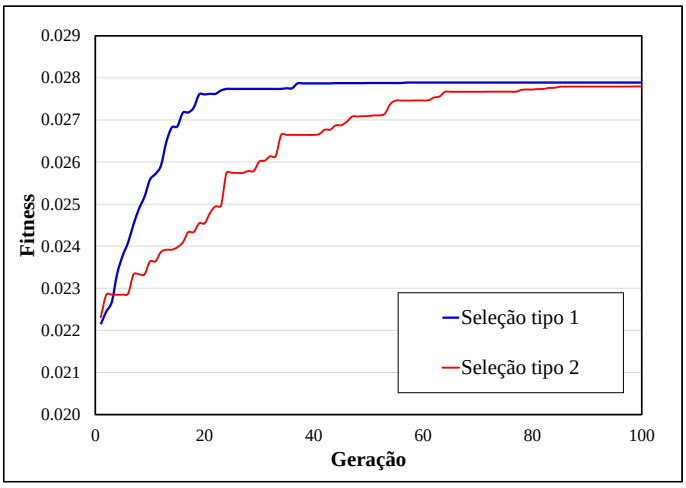


Figura 6.17 - Variação do *fitness* do melhor indivíduo em 100 gerações

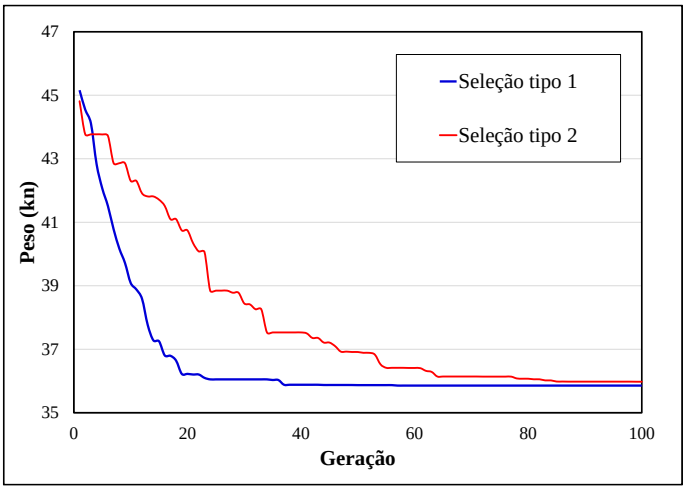


Figura 6.18 - Variação do peso do melhor indivíduo em 100 gerações

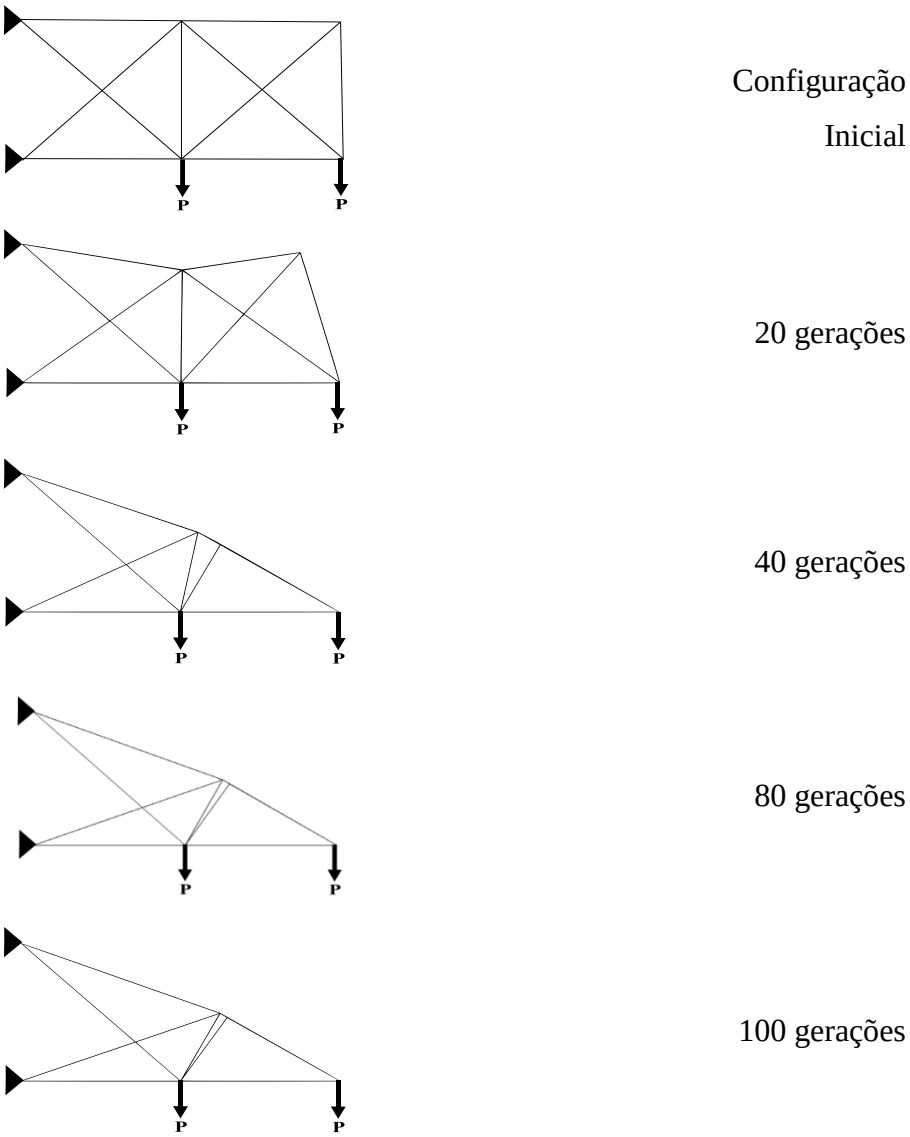


Figura 6.19 - Evolução de forma treliça de 10 barras.

O melhor individuo foi obtido por a análise de seleção tipo 1 onde, a nova configuração da estrutura proporcionou uma redução de 17,7 % do peso inicial da estrutura. Observou-se que, ao fim da otimização, dois nós ficaram em posições muito próximas sugerindo a fusão dos mesmos. Entretanto, dado que esta análise trata de otimização de forma, a junção de nós foi desconsiderada.

6.5.2 Caso 2: treliça arco, otimização de deslocamento.

Segundo Wang *et al.* (2002), o projeto do arco tem sido frequentemente atingido por otimização topológica. Neste trabalho, essa configuração tipo arco viria a ser adquirida por otimização da forma, com uma restrição de deslocamento. A concepção inicial da estrutura está apresentada na Figura 6.20. As características e restrições estão apresentadas na

Tabela 6.9 e a Tabela 6.10 respectivamente. As dimensões da treliça são $2{,}0\ \text{m} \times 0{,}5\ \text{m}$ e a seções transversais das barras é de $A = 10\ \text{cm}^2$.

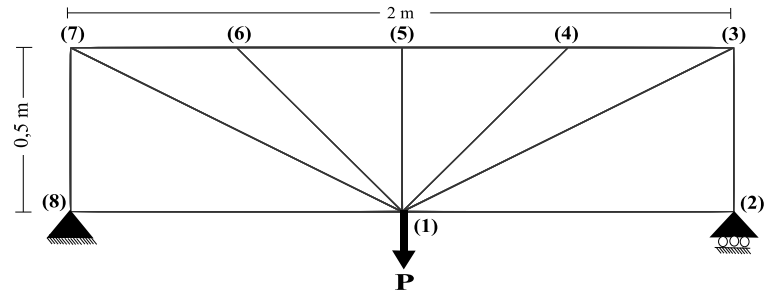


Figura 6.20 - Projeto inicial da treliça arco

Tabela 6.9 - Propriedades do material e restrições (Arco)

Propriedades	Valores
Modulo de Young (E)	$2{,}1\ x\ 10^8$ Kpa
Peso especifico (γ)	$76{,}44\ \text{Kn/m}^3$
Carga (P)	200 Kn
Deslocamento nodal admissível nó 1 ($d_{máx}$)	1,168 mm

Tabela 6.10 - Parâmetros para a treliça arco (Variáveis contínuas)

Parâmetros	
População (n_{pop})	90
Probabilidade de mutação (p_c)	0,3
Probabilidade de elitismo (p_m)	0,1
Operador de cruzamento	1 (um) ponto
Critério de parada	Menor peso encontrado em 200 gerações

Para este tipo de otimização, definiu-se a aptidão (*fitness*) pela expressão apresentada na Eq. (5.13). Neste caso, o deslocamento ao fim da otimização foi o menor possível. Para este exemplo, não se considerou restrição de peso nem de deslocamento. O comportamento do deslocamento para os melhores indivíduos em 200 gerações é apresentado na Figura 6.21 para seleção tipo 1 e seleção tipo 2. Pode-se ver que, ao igual que o caso 1, o comportamento do melhor individuo utilizando a seleção tipo 1 teve um melhor comportamento ao longo das gerações. A configuração final da treliça ao passo das 200 gerações com seleção tipo 1 é apresentada na Figura 6.22.

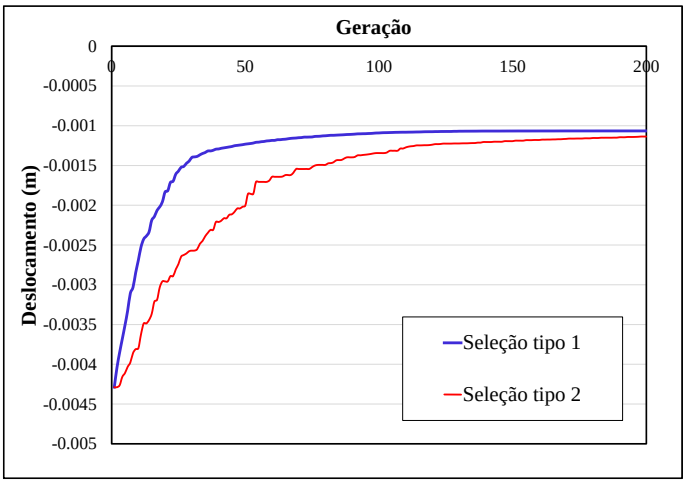
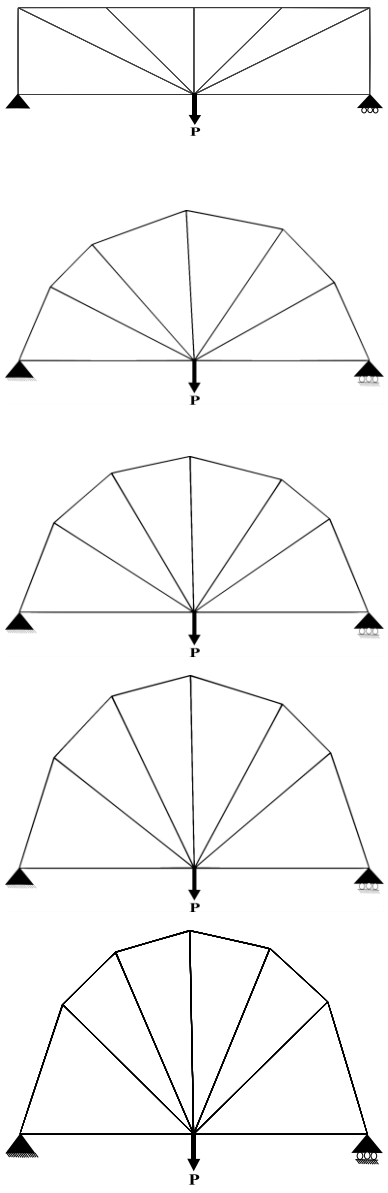


Figura 6.21 - Variação do deslocamento do melhor indivíduo em 200 gerações



Configuração Inicial

50 gerações

100 gerações

150 gerações

200 gerações

Figura 6.22 - Configuração da treliça no passo das gerações.

Dando cumprimento as restrições impostas, ao final do processo de otimização com seleção tipo 1, o peso da treliça aumento em um 20%, de 0,701 kN para 0,844 kN. O deslocamento no nó 1 foi reduzido em 76%, de 4,48 mm para 1,06 mm.

Estes resultados são comparados com os resultados obtidos por (Wang *et al.* 2002) onde, depois da otimização, apresenta um aumento do peso em 10% e uma redução de deslocamento de 75%. Neste caso, um aumento no peso proporcionou uma redução no deslocamento.

6.5.3 Caso 2: treliça arco, otimização de peso

O mesmo caso apresentado anteriormente foi utilizado para otimizar o peso final da treliça. Para este caso, foi considerada a expressão para o *fitness* apresentada na Eq. (5.14). Os valores de α e β foram de um e zero respectivamente. Neste caso, considerou-se como valor do peso limite um valor muito baixo para garantir que a Eq. (5.14) não tenha valores de zero. Este valor foi de 0,1, ou seja, o peso mínimo que pode ter a treliça é 10% do peso inicial, $0{,}07\ kN$. Neste exemplo, considerou-se que o deslocamento no nó 1 não pode ser maior que o deslocamento da configuração inicial. O comportamento do *fitness* do melhor indivíduo, para os estudos de seleção tipo 1 e tipo 2, é apresentado na Figura 6.23. Pode-se observar que, ao igual que os anteriores casos, o comportamento do *fitness* foi melhor para a análise com seleção tipo 1. A configuração da treliça ao longo das gerações, para a seleção tipo 1, é apresentada na Figura 6.24.

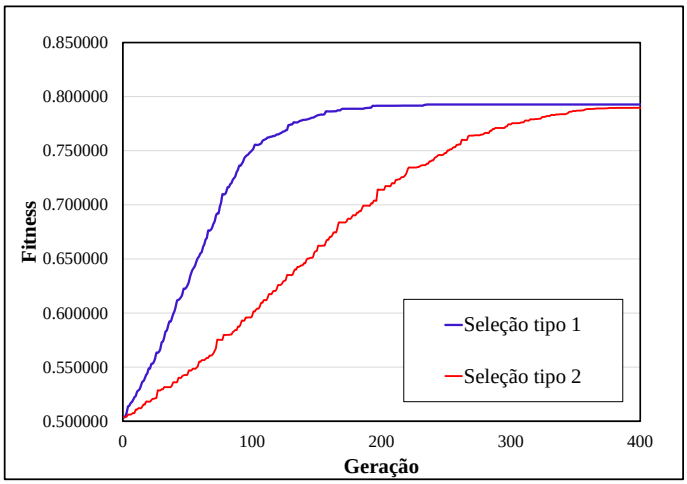


Figura 6.23 - Variação do *fitness* do melhor indivíduo em 400 gerações

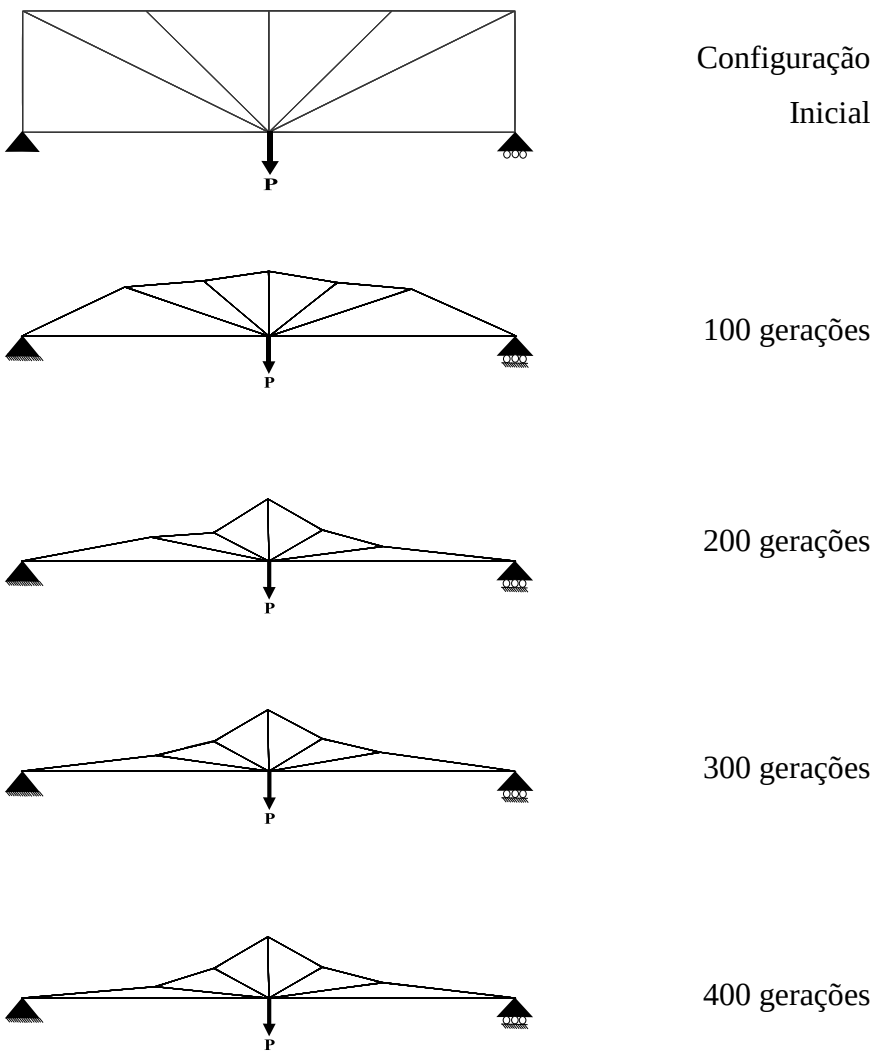


Figura 6.24 - Evolução da treliça em 400 gerações.

O resultado deste teste apresentou uma redução do peso final em um 37%, de 0,701 kN para 0,441 kN. Neste estudo, o deslocamento ficou igual a o inicial (4,48 mm). Pode-se observar na Figura 6.24 que a forma da treliça depois da otimização de peso foi muito diferente da configuração final depois da otimização de deslocamento da Figura 6.22. No caso da otimização de peso, a configuração da treliça ficou em forma de telhado oriental (pagode), mas na otimização de deslocamento, a forma de arco proporcionou mais rigidez na estrutura o que gerou menor deslocamento.

6.5.4 Caso 2: treliça arco, otimização simultânea de deslocamento e peso.

Para otimizar simultaneamente o deslocamento e peso da treliça apresentada no caso anterior considerou-se o *fitness* apresentado na Eq. (5.14). Os valores de α e β são de 0,5 para cada. O deslocamento nodal limite foi de 1,168 mm para o nó 1. O peso total limite

foi de $(1{,}1 \times W_i)$, ou seja, um aumento máximo de 10% do peso inicial da treliça. O comportamento dos *fitness* para os dois tipos de seleção, do melhor indivíduo nas 200 gerações, é apresentado na Figura 6.25. Pode-se observar que o melhor *fitness* foi obtido para a seleção tipo 1. Na Figura 6.26 é apresentado o comportamento do peso e do deslocamento ao longo das gerações para o melhor indivíduo. Pode-se observar que o comportamento do peso oscila por que eventualmente uma treliça com maior peso pode fornecer menor deslocamento e consequentemente melhor aptidão que a do melhor indivíduo da iteração anterior. A configuração da estrutura depois da otimização, com a seleção tipo 1, é apresentada na Figura 6.27.

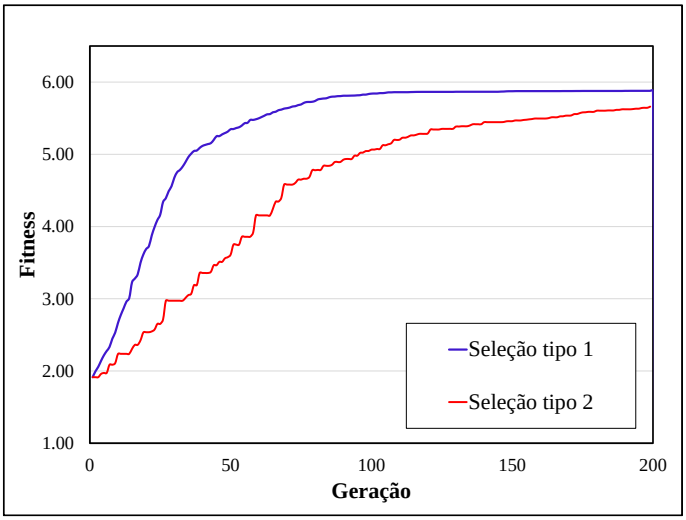


Figura 6.25 - Variação do *fitness* otimização simultânea em 200 gerações.

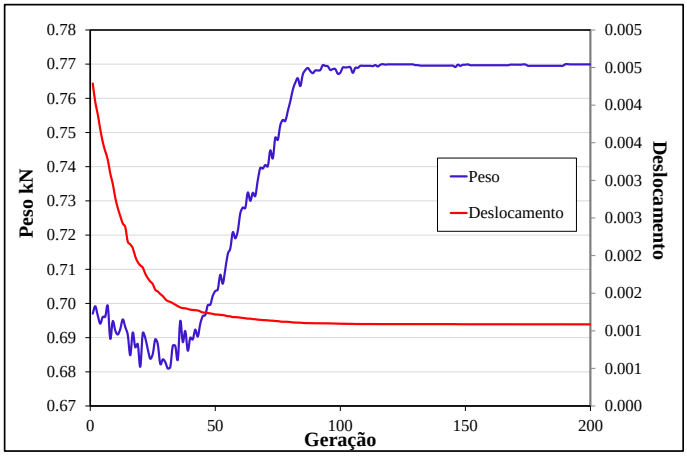
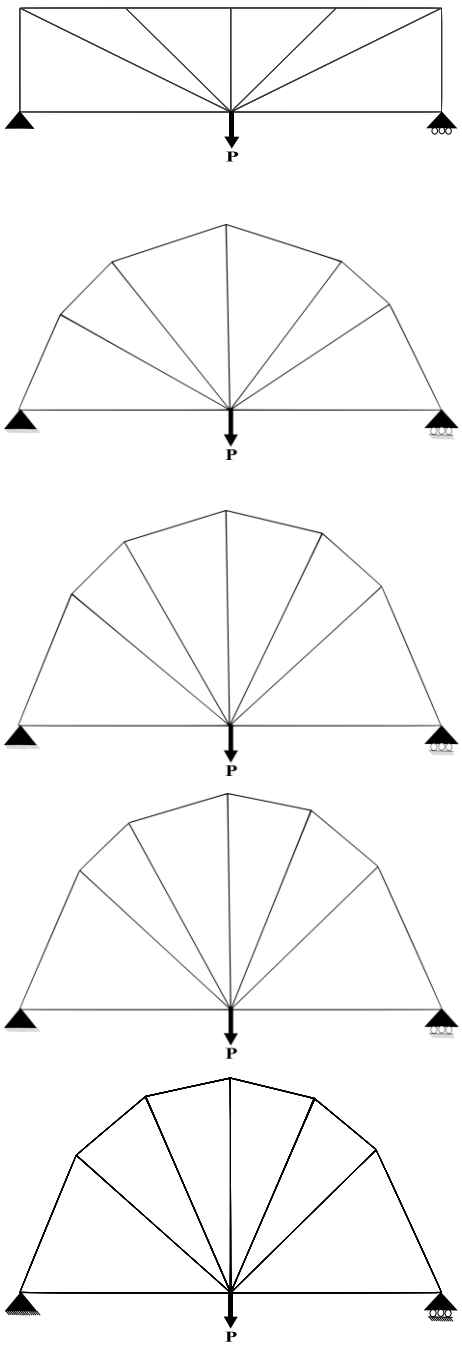


Figura 6.26 - Variação do peso e deslocamento para o melhor individuo da seleção tipo 1.



Configuração Inicial

50 gerações

100 gerações

150 gerações

200 gerações

Figura 6.27 - Configuração da treliça em 200 gerações.

O resultado apresentou um incremento no peso de 10%, passou de 0,701 kN para 0,771 kN. O deslocamento da estrutura diminuiu em um 76%, de 4,48 mm para 1,08 mm. Estes resultados comparados com os obtidos por (Wang *et al.* 2002), onde se aumento do peso em 10% e uma redução de deslocamento em um 75%, estão muito próximos, o que significam que a metodologia proposta demostrou eficiência no encontro da solução do problema.

Depois deste estudo, onde se consideraram diferentes critérios de otimização dentro da otimização de forma de uma treliça bidimensional em forma de arco, se apresenta a Figura 6.28 onde se tem um resumo das três configurações finais com seus respectivos resultados de peso, deslocamento e suas coordenadas nodais finais em metros.

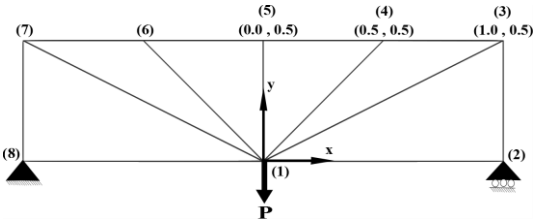
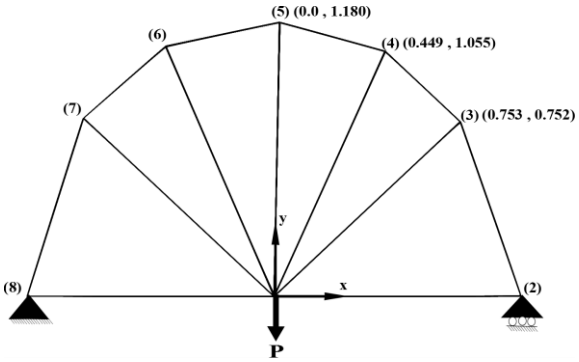
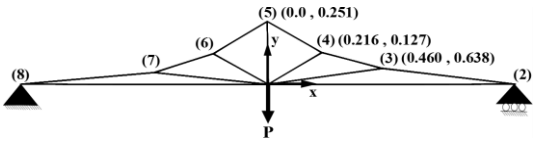
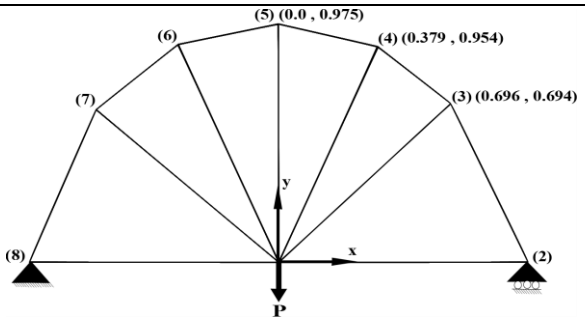
Configuração Inicial (CI) Peso: 0,701 kN Deslocamento: 4,48 mm	
Otimização de deslocamento sem restrições (OD-SR) Peso: 0,844 kN Deslocamento: 1,06 mm	
Otimização de peso com restrição de deslocamento (OP-RD) Peso: 0,441 kN Deslocamento: 4,48 mm	
Otimização simultânea de peso e deslocamento (OSPD-RP-RD) Peso: 0,771 kN Deslocamento: 1,08 mm	

Figura 6.28 - Resultados dos diferentes tipos de otimização de treliça tipo arco.

Para uma visualização mais detalhada das mudanças de peso e deslocamento nos diferentes processos de otimização, apresenta a Figura 6.29 com os resultados depois da otimização de peso e a Figura 6.30, os resultados finais de otimização de deslocamento.

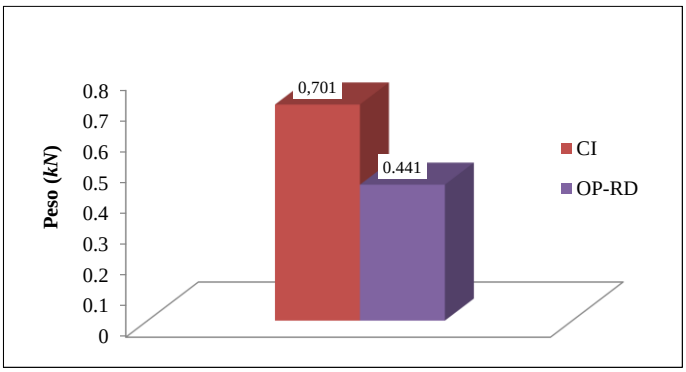


Figura 6.29 - Resultados de peso CI e OP-RD.

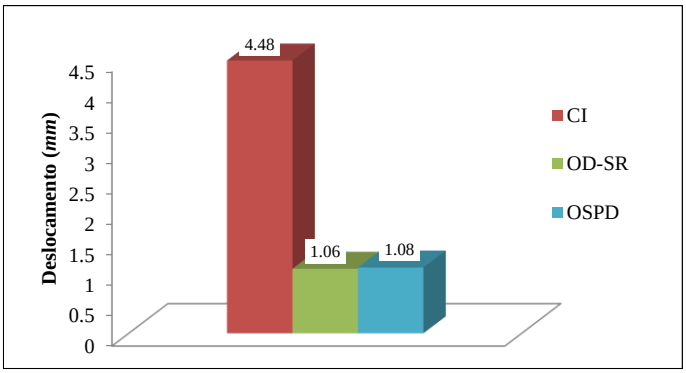


Figura 6.30 - Resultados de deslocamento da CI, OD-SR e OSPD.

Pode-se ver que na OP-RD apresenta-se uma redução do peso em um 37%, mas neste caso o deslocamento ficou igual ao inicial. Para os casos OD-SR e OSPD diminuiu consideravelmente o deslocamento. Isso é devido ao incremento da rigidez na estrutura o que gerou um incremento no peso final da estrutura. Das otimizações apresentadas, foi escolhida como a melhor otimização a OSPD por que cumpre com as restrições iniciais impostas e é aquela que tem o menor aumento do peso, o que proporciona um projeto com menor custo na hora de construído.

6.5.5 Caso 3: ponte simplesmente apoiada

Uma ponte simplesmente apoiada foi utilizada para otimização de forma (Wang *et al.* 2002). A concepção inicial da estrutura está representada na Figura 6.31. Os membros do grupo 1 da treliça tem uma seção transversal de $B = 8$ cm e $H = 5$ cm. Os elementos do grupo 2 tem uma área $A = 5\ \text{cm}^2$. O material tem um peso específico de

$\gamma = 76{,}51\,\mathrm{kN/m^3}$. Um modulo de Young de $2{,}1\,x\,10^8$ kPa. Os deslocamentos verticais máximos são limitados a 1 cm. A ponte tem um carregamento P = 10 kN em todos os nós do banzo inferior da treliça.

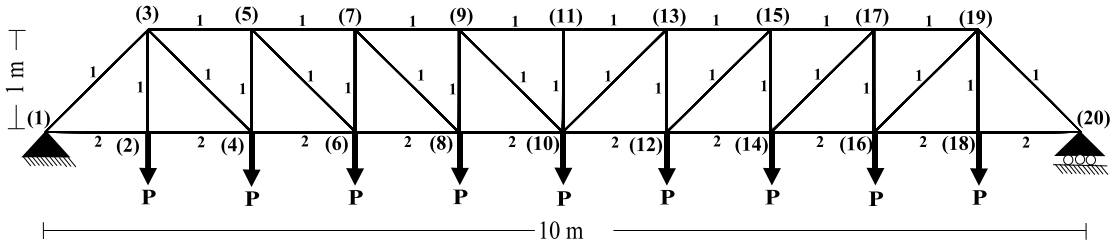


Figura 6.31 - Projeto inicial da ponte simplesmente apoiada.

Para este caso, o valor do *fitness* foi obtido pela Eq. (5.13). O resultado da variação do *fitness* é apresentado na Figura 6.32 durante as primeiras 200 gerações do processo de evolução para as duas análises de seleção a tipo 1 e a tipo 2. Pode-se observar que o comportamento do *fitness* para a seleção tipo 1 apresenta melhores resultados e que comparado com o comportamento de seleção tipo 2 esta alcança a convergir nas 200 gerações enquanto que a seleção tipo 2 ainda apresenta um comportamento crescente. Finalmente é apesentada na Figura 6.33 a evolução da treliça ao longo das gerações para a análise de seleção tipo1.

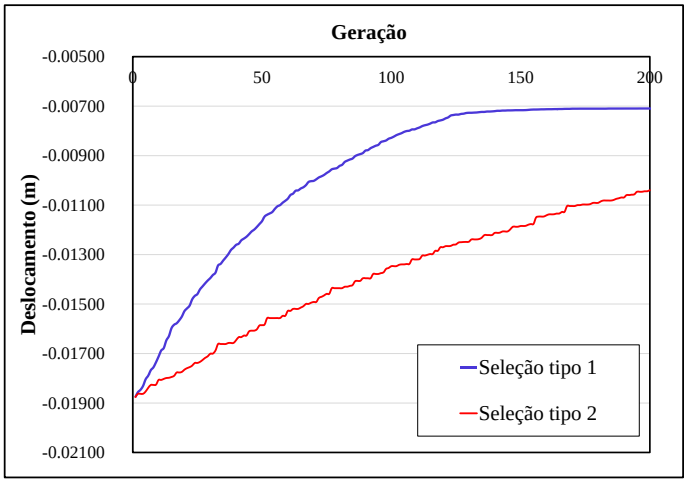


Figura 6.32 - Variação do deslocamento do melhor indivíduo em 200 gerações.

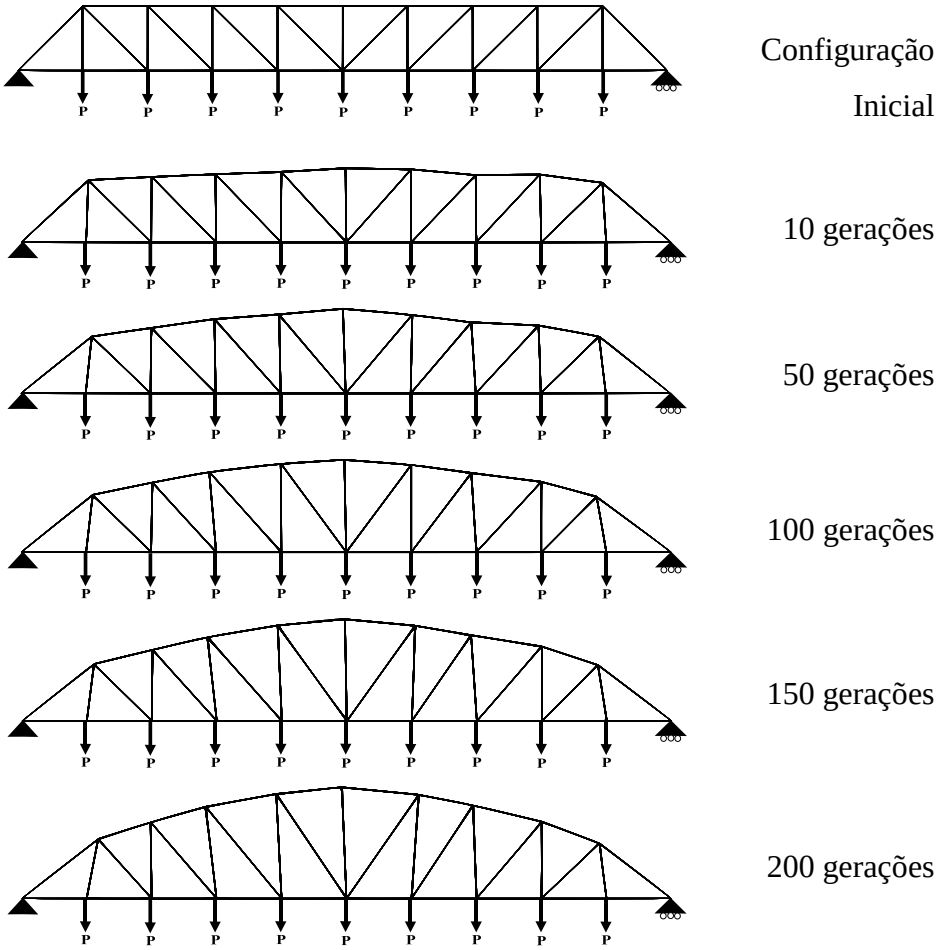


Figura 6.33 - Evolução da ponte durante 200 gerações

O peso final da treliça apresentou um aumento em um 5,5%, passou de 4,25 kN para 4,50 kN. Comparados com os resultados obtidos por (Wang *et al.* 2002) onde o resultado do peso apresentou um aumento de 11,5%. O deslocamento máximo da treliça diminuiu, passou de 0,019 m para 0,007 m.

Pode-se observar que para os casos estudados que o uso da seleção tipo 1, apresenta melhores valores de aptidão mostrando uma evolução mais rápida nas primeiras gerações. A evolução utilizando a elite como gerador de novos filhos por cruzamento apresentou uma evolução mais rápida nos casos estudados. Já para a otimização com seleção tipo 2, a evolução é menos rápida nas primeiras gerações, mas apresenta uma tendência sempre crescente em todos os casos onde foi analisado. Considera-se ainda que a seleção tipo 1 pode ser prejudicial à diversidade de indivíduos; o que pode levar à convergência em direção de um ótimo local.

6.5.6 Caso 4: cúpula

Wang *et al.* (2002) utilizou a estrutura apresentada na Figura 6.35 e na Figura 6.35 em forma de cúpula para otimizar simultaneamente a forma e as seções transversais das barras. Neste trabalho, e neste exemplo em particular, somente será realizada a otimização de forma. Desta maneira todos os elementos têm as mesmas secções transversais $A = 10\ \text{cm}^2$. Dado que os resultados apresentados nos anteriores casos foram melhores para a análise com seleção tipo 1, neste estudo, só é utilizado este tipo de seleção para a minimização de deslocamento máximo da treliça. A Tabela 6.11 apresenta as coordenadas da treliça. O movimento dos nós 14 até o 21 são restritos em todas as direções. A restrição de deslocamentos verticais do nó 1 não deve exceder 1 centímetro nos quatro casos de carga, apresentados na Tabela 6.12. O módulo de Young é $E = 2.1 \times 10^{11}\text{Pa}$ e o peso específico do material é de $77{,}00\ \text{kN/m}^3$. O *fitness* para este exemplo está dado pela Eq. (5.13). A Figura 6.36 apresenta a variação do deslocamento máximo com os quatro casos de carregamento dados. Pode-se observar que para os quatro casos de carregamento o comportamento do deslocamento diminui com o passo das gerações.

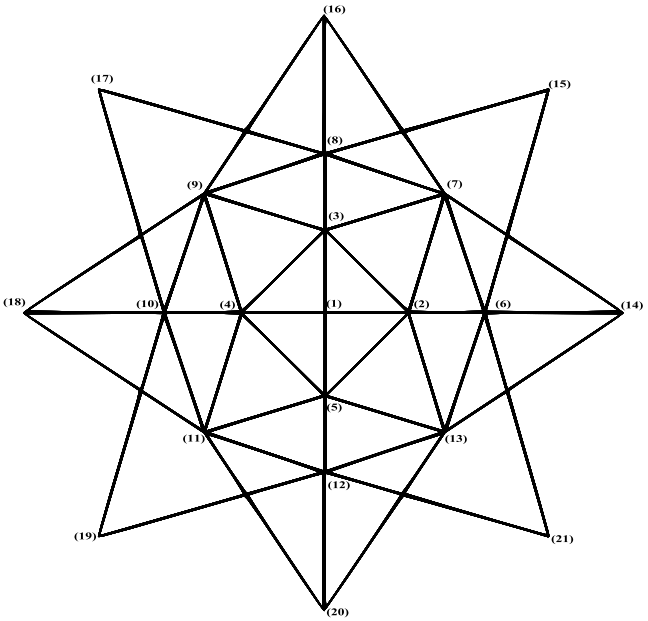


Figura 6.34 - Estrutura de cúpula. Vista superior

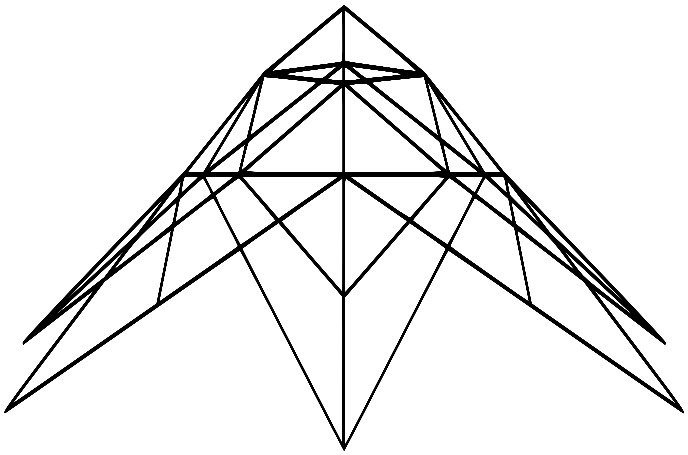


Figura 6.35 - Estrutura de cúpula. Vista em perspectiva

Tabela 6.11 - Coordenadas em metros da cúpula

Nó	Coordenadas (m)		
	x	*Y*	*z*
1	0,0	0,0	10,0
2	5,0	0,0	8,0
6	10,0	0,0	5,0
14	20,0	0,0	0,0

Tabela 6.12 - Casos de carga para a cúpula

Caso de carga	Valores	Nós
1	300 kN	1
2	30 kN	1-13
3	150 kN	1
	100 kN	4,5
4	150 kN	1
	70 kN	2-4

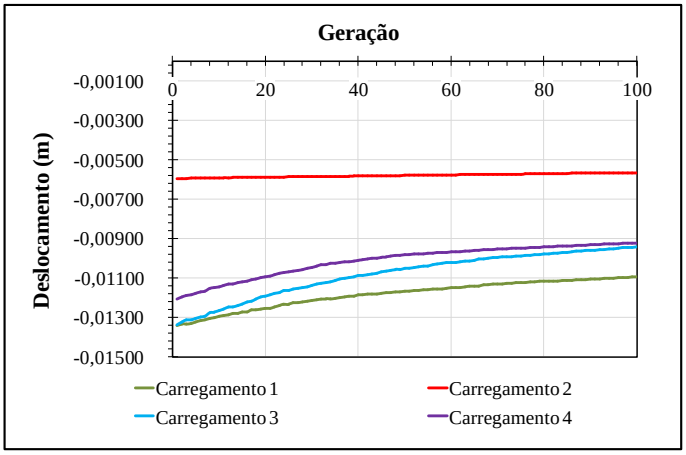


Figura 6.36 - Variação do deslocamento máximo nas 4 condições de carregamento

Os resultados obtidos são apresentados na Tabela 6.13, onde foi obtido um decréscimo de deslocamento, no nó 1, e de peso total na treliça em todos os casos de carga.

Tabela 6.13 - Resultados de deslocamento e peso da cúpula

Caso de carga	Deslocamento inicial (m)	Deslocamento final (m)	Peso inicial (kN)	Peso final (kN)
1	0,0135	0,0112	38,87	37,84
2	0,0059	0,0056	38,87	38,80
3	0,0135	0,0094	38,87	38,86
4	0,0122	0,0092	38,87	38,79

Neste exemplo, obtive-se uma redução no peso de 3% para o carregamento 1. Este resultado é comparado com o resultado obtido por Wang *et al.* (2002) onde se apresentou uma diminuição do peso de 8 %. A redução do peso neste estudo foi menor devido a que os autores Wang *et al.* (2002) utilizaram otimização simultânea, a dimensional e a de forma.

7 CONCLUSÕES E SUGESTÕES PARA TRABALHOS FUTUROS

Os problemas tratados nesta dissertação foram de otimização de peso e deslocamento de estruturas reticuladas tipo treliça. Esta otimização utilizou o método do Algoritmo Genético associado ao Método dos Elementos Finitos. Para atingir a otimização estrutural, foram descritas duas equações de aptidão as quais foram maximizadas com o objetivo de chegar a uma configuração chamada ótima. Para alguns estudos do caso, os valores de tensões axiais, deslocamentos nodais e peso limite foram considerados como restrições à função objetivo.

O Algoritmo Genético em questão foi implementado com técnicas de programação orientado a objetos utilizando-se as linguagens *Python* e *Julia,* com o uso das bibliotecas de Elementos Finitos *Pyfem* e *FEMlab.*

7.1 CONCLUSÕES

Com os resultados encontrados nos exemplos estudados podem-se apresentar as seguintes conclusões:

- *Ao fim das implementações o tempo de gasto na linguagem Julia foi a menor.* Das implementações feitas, *Julia* foi escolhida para continuar no processo de otimização por a redução do tempo de gasto na implementação comparada com a linguagem *Python*.

- *Os resultados apresentam diversos comportamentos na evolução de acordo com os diferentes parâmetros utilizados.* Observou-se que a mudança da taxa de elitismo não teve muita relevância na otimização e que para todos os casos conserva a mesma tendência. Já, no estudo do fator de mutação comprovou-se que os coeficientes com maior variabilidade apresentaram uma convergência mais acelerada. Finalmente, foi observado que taxas de 10% e 30% de elitismo e mutação, respectivamente, favorecem o processo de evolução na otimização de estruturas de treliças, especialmente nas primeiras gerações.

- *Dos estudos realizados com os dois tipos de seleção, a seleção tipo 1acelerou o processo de convergência*. Pode-se concluir que na seleção tipo 1, os indivíduos com elitismo geram filhos com uma aptidão maior e estes estão mais perto da

possível solução. Neste tipo de seleção a solução pode ser um ótimo local devido a que a diversidade da população pode ser prejudicada, mas que na otimização com seleção tipo 2, quando combinar indivíduos de toda a população estes geram filhos com valores de aptidão mais baixos por que se cruzam indivíduos de aptidões maiores com outros que tem aptidão menor.

- *Os resultados obtidos dos testes de otimização de deslocamento estão diretamente associados com o aumento de rigidez da estrutura.* Nos casos de otimização de deslocamento, observou-se a aparição de formatos de arco nas estruturas, o que fornece maior rigidez às mesmas. Este resultado era esperado uma vez que estruturas com maior rigidez fornecem menores deslocamentos. No caso da otimização de peso, observou-se que a configuração da treliça depois da otimização ficou em forma de telhado oriental. Esta configuração proporcionou um menor peso sem superar o deslocamento inicial (limite de deslocamento).

- *Os resultados obtidos, a serem comparados com os resultados de outros autores, demostraram que a metodologia proposta permitiu encontrar soluções satisfatórias nos problemas de estruturas reticuladas.* Da comparação com outros autores dos resultados obtidos no exemplo de referência pode-se concluir que a metodologia utilizada permitiu obter melhores resultados com o mesmo número de gerações. Somente os autores Deb e Gulati (2001) apresentam um peso menor aos obtidos neste estudo, entretanto eles utilizaram uma população muito maior e também um número maior de gerações.

- *A visualização do processo de otimização de forma permitiu acompanhar a evolução da configuração da estrutura*. O uso do *sofware Paraview®* permitiu acompanhar a evolução da otimização das treliças com o passo das gerações.

- *O uso de uma codificação real permitiu ganhar tempo na hora da codificar os indivíduos*. A comparar a codificação real com outras técnicas de codificação encontradas na literatura, como é a codificação binaria, a escolha da codificação real permitiu gerar cromossomos diretamente com os valores das seções transversais e dos nós da estrutura permitindo ganhar tempo na codificação.

- *O uso da aptidão (fitness) apresentado na Eq. (5.14) permitiu uma otimização multiobjectivo*. Isto facilitou a visualização da melhor otimização para um mesmo

caso permitindo encontrar a melhor configuração da estrutura. Por esta razão, a OSPD (otimização simultânea de peso e deslocamento) para o caso 2 na otimização de forma, é escolhida como a melhor otimização, pois cumpre com as restrições iniciais impostas e é aquela que tem o menor aumento no peso.

7.2 SUGESTÕES PARA TRABALHOS FUTUROS

Para pesquisas futuras podem ser considerados os seguintes aspectos:

- *Comtemplar nos diferentes tipos de otimização de treliças outros tipos de restrições*. Para que a implementação da otimização seja mais real, é preciso incluir algumas restrições como: limites referentes a cargas críticas, tensões de flambagem, restrições geométricas como comprimento dos elementos ou seções pré-estabelecidas, carregamentos e restrições gerais previstas em normas.

- *Criação de uma interface gráfica.* Para que o processo de otimização seja mais interativo, a criação de uma interfase gráfica proporciona ao projetista uma ferramenta prática na hora da otimização.

- *Tratamento da otimização submetidas a problemas dinâmicos.* Otimização de estruturas reticuladas submetidas a frequências naturais de vibração.

- *Estudos de convergência para os AGs*. O uso de outros métodos conjuntamente com os Algoritmos Genéticos que permitam acelerar a convergência, em especial a evitar convergência prematura em ótimos locais, para melhora das soluções encontradas.

- *Testes variando as taxas dos operadores genéticos.* Desenvolvimento e teste de novas estratégias de penalização, seleção, mutação, cruzamento e elitismo que permitam obter critérios bem definidos do uso das taxas dos operadores genéticos em busca da melhor solução.

- *Meta-otimização dos operadores genéticos.* Uma otimização dos operadores Genéticos utilizando a técnica dos Algoritmos Genéticos, pode proporcionar uma otimização dos parâmetros a se utilizar no processo de otimização.

- *Otimização simultânea.* Pode-se utilizar a otimização topológica juntamente com a dimensional e a de forma, a fim de se conseguir melhores resultados.

REFERÊNCIAS BIBLIOGRÁFICAS

Afonso, S. M. B., Vaz, L. E. (2000a). Otimização de placas à Flexão Submetidas a Carregamento Dinâmico. I Workshop on Computational Mechanics, 2000, Recife - PE. *I Workshop on Computational Mechanics*, v. 1. pp. 27-37.

Afonso, S. M. B., Vaz, L. E. (2000b). Sensitivity Analysis and Shape Optimisation of Shells Under Dynamic Solicitation. II ASMO UK / ISSSMO *Conference on Engineering Design Optimization*. University of Wales, Swansea, UK, v. 1, pp. 73-78.

Afonso, S. M. B., Sienz, J.; Belbindia, F. (2005). Structural Optimization Strategies for Simple And Integrally Stiffened Plates And Shells. *Engineering Computations*, v. 22, n.4, pp. 429-452.

Afonso, S. M. B., Falco, S. A., Vaz, L. E. (2001). Structural Optimization of Thin Shells Under Dynamic Solicitation. *Third International Conference on Thin-Walled Structures*, 2001, Cracóvia. Thin-Walled Structures - Advances and Developemnts - Oxford, UK, Elsevier Science, v. 1, pp. 373-380.

Castro, L. L. B. (2005). *Algoritmo Genético Para Otimização de Estruturas Reticuladas.* Dissertação de mestrado, Universidade de Brasília, Brasília - DF, 106 p.

Castro, R. E. (2001). *Otimização de Estruturas com Multi-objetivos Via Algoritmos Genéticos de Pareto.* Tese de Doutorado, Universidade Federal do Rio de Janeiro, Rio de Janeiro - RJ, 224 p.

Christensen, P. W., Klarbling, A. (2009). *An Introduction Structural Optimization.* Springer, 211 p.

Coley, D. A. (1999). *An Introduction to Genetic Algorithms for Scientists and Engineers*. World Scientific, Singapore, River Edge, NJ. 223 p.

De Jong, K. A., Sarma, J. (1992). Generation gaps revisited. *Proceedings of the Foundations of Genetic Algorithms Workshop.* Vail, CO: Morgan Kaufmann.

Deb, K., Gulati, S. (2001). Design of Truss-Structures for Minimum Weight Using Genetic Algorithms. *Finite Elements in Analysis and Design*, v. 37(5), pp. 447–465.

Durand, R. D. (2008). *Análise Tridimensional de Estruturas Geotécnicas Submetidas a Reforço e Drenagem*. Tese de doutorado em Geotecnia, Universidade de Brasília, Brasilia - DF, 153 p.

Ebadi, M. M., Rashtchi, V., Behravesh, A. (2011). Sizing and Geometry Optimization of Pin Connected Structures via Real Coded Genetic Algorithm (RCGA). *Global Journal of Computer Science and Technology,* v. 11. 7 p.

Fadel, M.L., Holdorf, L.R., Fadel, M.L. (2013). Multimodal Size, Shape and Topology Optimisation of Truss Structures Using the Firefly Algorithm. *Advances in Engineering Software,* v. 56, pp. 23-37.

Falco, S. A., Afonso, S. M. B., Vaz, L. E. (2004). Analysis and Optimal Design of Shell Structures under dynamic Loads-I: Finite Element and Sensitivity Analysis. *Structural and Multidisciplinary Optimization*, Springer-Verlag, v. 27, pp. 189-196.

Fogel, B.D. (1997). *The Advantages of Evolutionary Computation*. Proceedings of BCEC97: BioComputing and Emergent Computation, World Scientific, Singapore, pp. 1-11.

Fonseca, M. (2007). *Otimização de Estruturas Treliçadas Planas e Espaciais Sob Carregamentos Estáticos e Dinâmicos, Usando Algoritmos Genéticos e Redes Neurais*. Dissertação de Mestrado. Universidade Federal de Ouro Preto, Ouro Preto - MG, 184 p.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Pub. Co., Reading, Mass, 412 p.

Goldberg, D. E., Deb K. (1991). A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. *Foundations of Genetic Algorithms*. Morgan Kaufmann, pp. 69-93.

Goldberg, D. E., Deb, K. (1989). An Investigation of Niche And Species Formation in Genetic Function Optimization. *Proceedings of the third international conference on Genetic algorithms*, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc., pp. 42-50.

Goldberg, D. E., Zakrzewski, K., Sutton, B., Gadient, R., Chang, C., Gallego, P, Miller, B. E Cantú-Paz, E. (1997). *Genetic Algorithms: A Bibliography*, Illigal Report Nº 97011, Illinois Genetic Algorithms Laboratory.

Guerra, C. (2008). *Otimização Paramétrica de Estruturas Treliçadas por Algoritmos Genéticos.* Dissertação de mestrado, Universidade Federal do Rio Grande do Sul, Porto Alegre - RS, 111 p.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*, University of Michigan Press, USA, 33 p.

Ignizio, J. P., Cavalier, T.M. (1994). *Linear Programming*. Englewood Cliffs: Prentice Hall, 666 p.

Kaveh A, Shahrouzi M., 2008. Optimal Structural Design Family by Genetic Search and Ant Colony Approach. *Engineering Computations*, v. 25(3), pp. 268-288.

Koza, J. (1994). *Genetic Programming as a Means for Programming Computers by Natural Selection*. Statistics and Computing. 4(2) pp. 87–112.

Krishnamoorthy, C.S., Prasanna, V.P., Sudarshan, R. (2002). Object-Oriented Framework for Genetic Algorithms with Application to Space Truss Optimization. *Journal of computing in Civil Engineering*, v.75, pp. 66-75.

Kuri, M. Á., Galaviz, C. J. (2002). *Algoritmos Genéticos*. Instituto Politécnico Nacional, Universidad Nacional Autónoma de México, Fondo de Cultura Económica, México, 60 p.

Lemonge, A. C. C. (1999). *Aplicação de Algoritmos Genéticos em Otimização Estrutura*, Tese de Doutorado, Universidade Federal do Rio de Janeiro, Rio de Janeiro - RJ, 218 p.

Liu, G. R., Quek, S. S. (2013). *The Finite Element Method: A Practical Course*. Butterworth-Heinemann, 365 p.

Melchers, R. E., Hough, R. (2007). *Modelling Complex Engineering Structures.* American Society of Civil Engineers. Reston, Virginia, 359 p.

Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs.* Third, Revised and Extended Edition. Springer. New York. 388 p.

Miller, B. L., Goldberg, D. E. (1996). Genetic Algorithms, Selection Schemes, and the Varying Effects of Noise. *Evol. Comput*. v. 4, pp. 113-131.

Nanakorn P, Meesomklin K., 2001. An Adaptive Penalty Function in Genetic Algorithms for Structural Design Optimization. *Computers and structure,* v.79, pp. 2527-2539.

Pizzirani, F. (2003). *Otimização Topológica de Estruturas Utilizando Algoritmos Genéticos*. Dissertação de mestrado, Universidade Estadual de Campinas, São Paulo, 104 p.

Rao, S. S. (2004). *The Finite Element Method in Engineering*, Fourth Edition. Elsevier Science & Technology Books.

Razan, C., Lucian, G., 2014. Steel Truss Optimization Using Genetic Algorithms and FEA. *The 7th International Conference Interdisciplinarity in Engineering*, pp. 339-346.

Sánchez, C. S. (2012). *Optimización Estructural y Topológica de Estructuras Morfologicamente no Definidas Mediante Algoritmos Genéticos*. Tesis doctoral. Universitat politécnica de Valencia, Valencia, Espanha, 361 p.

Silva, F. B. (2011). *Algoritmos Genéticos para Otimização de Estruturas Reticuladas Baseadas em Modelos Adaptativos e Lagrangeano Aumentado*. Dissertação de mestrado, Universidade Federal de Juiz de Fora, Juiz de Fora, 186 p.

Smith, J. (2007). On Replacement Strategies in Steady State Evolutionary Algorithms. *Evol. Comput*, v.15, pp. 29-59.

Soares, G. L. (1997). *Algoritmos Genéticos: Estudo, Novas Técnicas e Aplicações,* Dissertação de Mestrado. Universidade Federal de Minas Gerais, Belo Horizonte, 137 p.

Vanderplaats, G. N. (1993). Thirty Years of Modern Structural Optimization. *Advances in Engineering Software,* v.16, pp. 81-88.

Wang, D., Zhang, W. H., Jiang, J. S. (2002). Truss Shape Optimization whit Multiple Displacement Constraints. *Computer methods in applied mechanics and engineering*, pp. 3597-3612.

Zuo, W., Xu,T., Zhang, H., Xu, T. (2011). Fast Structural Optimization with Frequency Constraints by Genetic Algorithm Using Adaptive Eigenvalue Reanalysis Methods. *Struct Multidisc Optim*, v.43, pp. 799-810.

APÊNDICES

A. - PROGRAMA DE OTIMIZAÇÃO DIMENSIONAL

Neste apêndice é apresentado o código-fonte, escrito em *Julia*, do caso apresentado no item 6.2. Ainda, este algoritmo usa a biblioteca de elementos finitos *FemLab*.

```
# Inclusão da biblioteca de elementos finitos
using FemLab
# Dados de entrada da treliça
coord = [ 0 0; 9 0; 18 0; 0 9; 9 9; 18 9.] #coordenadas dos nós
conn  = [ 1 2; 1 5; 2 3; 2 6; 2 5; 2 4; 3 6; 3 5; 4 5; 5 6] # conectividades

# Comandos de geração de malha
blt  = BlockTruss(coord, conn)
mesh = generate_mesh(blt, verbose=false)

# Criação do domínio
dom  = Domain(mesh)

# Definição dos materiais
set_mat(dom.elems, Truss(E=6.894757e7, A=0.043) )
# Modulo de Young (Kpa) do material e área da seção transversal (m).

# Definição das condições de contorno
set_bc( dom.nodes[:(x==0 && y==0)] , ux=0, uy=0)
set_bc( dom.nodes[:(x==0 && y==9)] , ux=0, uy=0)
set_bc( dom.nodes[:(x==9 && y==0)] , fy=-450.)  # Carregamentos aplicados (kN)
set_bc( dom.nodes[:(x==18&& y==0)] , fy=-450.)  # Carregamentos aplicados (kN)

# Variável truss representa o domínio
truss = dom

# Função para obter um cromossomo a partir de uma treliça (cromossomo das áreas)
function get_chrom(truss::Domain)
    return Float64[ elem.mat.A for elem in truss.elems ]
end

# Função modifica os genes dentro do cromossomo
function mfactor()
    return 0.5 + rand()
End
```

```
# Função gera novos indivíduos por mutação
function mutate(cr, rate=0.5)
    # cr: representa o cromossomo de uma treliça
    n = length(cr)
    # ngens: número de indivíduos a ser mutuados
    ngens = ifloor(n*rate)
    idx = [1:n]
    idx = shuffle(idx)[1:ngens]
    for i in idx
        cr[i] *= mfactor()
    end
    return cr
end

# Função gera um novo cromossomo baseado em cromossomos de dois progenitores
function crossover(cr1, cr2)
    # n: número de genes dentro do cromossomo
    n   = length(cr1)
    pos = rand(1:n-1)
    cr  = [ cr1[1:pos], cr2[pos+1:end] ]
    return cr
end

# Função gera um cromossomo para um novo indivíduo
function gen_ind()
    cr = get_chrom(truss)
    return mutate(cr)
end

# Função gera a população
function gen_pop(num::Int)
    # num: representa o número de indivíduos na população
    pop = [ gen_ind() for i=1:num ]
    return pop
end
```

```
# Função calcula a aptidão para um indivíduo (cromossomo)
function fitness(cr)
    n = length(cr)
    # retribui valores ao cromossomo do individuo
    E = truss.elems[1].mat.E
    for i=1:n
        set_mat( truss.elems[i], Truss(E=E, A=cr[i] ))
    end

    # Cálculo do peso
    weight = 0.0
    gamma  = 25.9  # peso específico (kN/m3)
    for elem in truss.elems
        c1 = elem.nodes[1].X
        c2 = elem.nodes[2].X
        L  = norm(c2-c1)
        A  = elem.mat.A
        weight = weight + A*L*gamma
    end

    # Reinicia os deslocamentos e as tensões
    reset(truss.nodes)
    reset(truss.elems)

    # Análise via elementos finitos
    solve!(dom, verbose=false, reset_bc=false)

    # Verifica os deslocamentos
    disps = [ abs(node.dofdict[:uy].U) for node in truss.nodes ]
    max_disp = maximum(disps)
    if max_disp > 0.0508 # displacement (m)
        return 0.0
    end

    # Verifica as tensões
    stresses = [ elem.ips[1].data.σ for elem in truss.elems]
    tens_stress = abs(maximum(stresses))
    comp_stress = minimum(stresses)

    if tens_stress>130000.0 # stress (KPa)
        return 0.0
    end
    return 1./weight
end
```

```
# Função realiza a evolução
function evolve(n::Int; nger::Int=5, elit::Float64=0.1, mutation::Float64=0.3)
    pop  = gen_pop(n)
    data = DTable( [ :generation, :fitness ] )

    for k=1:nger
        println("\nGeneration: ", k)
        fits = [ fitness(ind) for ind in pop ]
        idxs = sortperm(fits, rev=true)

        # ordena a população de acordo com a aptidão
        pop  = pop[idxs]
        fits = fits[idxs]
        bestw = 1 / fits[1]
        println("Best fitness :", fits[1])
        println("Best Weight  :", bestw)

        push!(data, [k, fits[1] ] )

        if k==nger; break end

        nelit  = ifloor(elit*n)
        ncross = n-nelit
        pop = pop[1:nelit]

        # Cruzamento
        for i=1:ncross
            ind1 = pop[rand(1:nelit)]
            ind2 = pop[rand(1:nelit)]
            ind  = crossover(ind1, ind2)
            push!(pop, ind)
        end

        # Mutação
        nmutate = ifloor(mutation*n)
        for i=1:nmutate
            pos = rand(nelit+1:n)
            ind = pop[pos]
            mutate(ind)
        end
    end
    save(data, "fitness.dat") # Armazena em disco os resultados da análise
end
```

```
# Inicialização do gerador de números aleatórios
srand(0)

# Chamada à função que realiza o processo evolutivo (número de indivíduos, número de gerações, taxa de elitismo e taxa de mutação)
evolve(90, nger=100, elit=0.1, mutation=0.3)
```

B. - PROGRAMA DE OTIMIZAÇÃO DE FORMA – (SEL. TIPO 1)

Neste apêndice é apresentado o código-fonte do caso apresentado no item 6.5.1. Foi utilizado a seleção tipo 1 (seleção aleatória uniforme). A programação foi desenvolvida na linguagem *Julia* juntamente com a biblioteca de elementos finitos *FemLab*.

```
# Inclusão da biblioteca de elementos finitos
using FemLab
#Dados de entrada da treliça
coord = [ 0 0; 9 0; 18 0; 0 9; 9 9; 18 9.] #coordenadas dos nós
conn  = [ 1 2; 1 5; 2 3; 2 6; 2 5; 2 4; 3 6; 3 5; 4 5; 5 6]  # conectividades

# Comandos de geração de malha
blt = BlockTruss(coord, conn)
mesh = generate_mesh(blt, verbose=false)

# Criação do domínio
dom = Domain(mesh)

# Definição dos materiais
set_mat(dom.elems, Truss(E=6.894757e7, A=0.01710) )
# Modulo de Young (Kpa) do material e área da seção transversal (m).

# Definição das condições de contorno
set_bc( dom.nodes[:(x==0 && y==0)] , ux=0, uy=0)
set_bc( dom.nodes[:(x==0 && y==9)] , ux=0, uy=0)
set_bc( dom.nodes[:(x==9 && y==0)] , fy=-450.)  # Carregamentos aplicados (kN)
set_bc( dom.nodes[:(x==18&& y==0)] , fy=-450.)  # Carregamentos aplicados (kN)

# Variável truss representa o domínio
truss = dom
```

```
# Função para obter um cromossomo a partir de uma treliça (cromossomo dos nós)
function get_chrom(truss::Domain)
    movable_nodes = [ 5, 6 ]
    cr =Float64[]
    for idx in movable_nodes
        push!(cr, truss.nodes[idx].X[1])
        push!(cr, truss.nodes[idx].X[2])
    end
       #println("cr:  ",cr)
    return cr
end

# Função modifica os genes dentro do cromossomo
function mfactor()
        return -1.0 + 2.0*rand()
end

# Função gera novos indivíduos por mutação
function mutate(cr, rate=0.5)
    # cr: representa o cromossomo de uma treliça
    n = length(cr)
    # ngens: número de indivíduos a ser mutuados
    ngens = ifloor(n*rate)
    idx = [1:n]
    idx = shuffle(idx)[1:ngens]
    for i in idx
        cr[i] *= 0.02*mfactor()
    end
    return cr
end

# Função gera um novo cromossomo baseado em cromossomos de dois progenitores
function crossover(cr1, cr2)
    # n: número de genes dentro do cromossomo
    n   = length(cr1)
    pos = rand(1:n-1)
    cr  = [ cr1[1:pos], cr2[pos+1:end] ]
    return cr
end
```

```
# Função gera um cromossomo para um novo indivíduo
function gen_ind()
    cr = get_chrom(truss)
    return mutate(cr)
end

# Função gera a população
function gen_pop(num::Int)
    # num: representa o número de indivíduos na população
    pop = [ gen_ind() for i=1:num ]  # num : número de indivíduos na população
    return pop
end

# Função calcula a aptidão para um indivíduo (cromossomo)
function fitness(cr)
    n = length(cr)
    # retribui valores ao cromossomo do individuo
    truss.nodes[5].X[1] = cr[1]
    truss.nodes[5].X[2] = cr[2]
    truss.nodes[6].X[1] = cr[3]
    truss.nodes[6].X[2] = cr[4]

    # cálculo do peso
    weight = 0.0
    gamma  = 25.9  # unit weigth kN/m3
    for elem in truss.elems
        c1 = elem.nodes[1].X
        c2 = elem.nodes[2].X
        L  = norm(c2-c1)
        A  = 0.0171
        weight = weight + A*L*gamma
    end

    # reinicia os deslocamentos e as tensões
    reset(truss.nodes)
    reset(truss.elems)

    # Análise via elementos finitos
    solve!(dom, verbose=false, reset_bc=false)

    # Verifica os deslocamentos
    disps = [ abs(node.dofdict[:uy].U) for node in truss.nodes ]
    max_disp = maximum(disps)
    if max_disp > 0.0508 # displacement (m)
```

```
        return -1. # flag
    end

    # Verifica as tensões
    stresses = [ elem.ips[1].data.σ for elem in truss.elems]
    tens_stress = abs(maximum(stresses))
    comp_stress = minimum(stresses)

    if tens_stress>130000.0 # stress (KPa)
        return -1
    end
    return 1./weight
end

# Atualiza a treliça de acordo com os valores do cromossomo
function set_truss(cr)
    truss.nodes[5].X[1] = cr[1]
    truss.nodes[5].X[2] = cr[2]
    truss.nodes[6].X[1] = cr[3]
    truss.nodes[6].X[2] = cr[4]
end

# Função que realiza a evolução
function evolve(n::Int; nger::Int=5, elit::Float64=0.1, mutation::Float64=0.3)
    pop  = gen_pop(n)
    data = DTable( [ :generation, :fitness ] )

    for k=1:nger
        println("\nGeneration: ", k)

        fits = [ fitness(ind) for ind in pop ]
              #println("Fitness: ", fits)

        idxs = sortperm(fits, rev=true)
        # ordena a população de acordo com a aptidão
        pop  = pop[idxs]
        fits = fits[idxs]
        bestw = 1 / fits[1]
        println("Best fitness :", fits[1])
        println("Best Weight  :", bestw)

        push!(data, [k, fits[1] ] )
        if mod(k,10)==0
        end
```

```
        if k%5==0
            save(truss,"truss_$k.vtk")
        end

        if k==nger; break end
        nelit  = ifloor(elit*n)
        ncross = n-nelit
        pop = pop[1:nelit]

        # Cross over
        for i=1:ncross
            ind1 = pop[rand(1:nelit)]
            ind2 = pop[rand(1:nelit)]
            ind  = crossover(ind1, ind2)
            push!(pop, ind)
        end

        # Mutation
        nmutate = ifloor(mutation*n)
        for i=1:nmutate
            pos = rand(nelit+1:n)
            ind = pop[pos]
            mutate(ind)
        end

    end
       best_cr = pop[1]
    set_truss(best_cr)
    save(truss,"truss2.vtk")
    save(data, "fitness.dat") # Armazena em disco os resultados da análise
end

# Inicialização do gerador de números aleatórios
srand(0)

# Chamada à função que realiza o processo evolutivo (número de indivíduos, número de
gerações, taxa de elitismo e taxa de mutação)
evolve(90, nger=100, elit=0.1, mutation=0.3)
```

C. - PROGRAMA DE OTIMIZAÇÃO DE FORMA – (SEL. TIPO 2)

Neste apêndice é apresentado o código-fonte do caso apresentado no item 6.5.1. Foi utilizado a seleção tipo 1 (seleção por roleta). É utilizada a linguagem *Julia* juntamente com a biblioteca de elementos finitos FemLab.

```
# Inclusão da biblioteca de elementos finitos
using FemLab
#Dados de entrada da treliça
coord = [ 0 0; 9 0; 18 0; 0 9; 9 9; 18 9.] #coordenadas dos nós
conn  = [ 1 2; 1 5; 2 3; 2 6; 2 5; 2 4; 3 6; 3 5; 4 5; 5 6]  # conectividades

# Comandos de geração de malha
blt = BlockTruss(coord, conn)
mesh = generate_mesh(blt, verbose=false)

# Criação do domínio
dom = Domain(mesh)

# Definição dos materiais
set_mat(dom.elems, Truss(E=6.894757e7, A=0.01710) )
# Modulo de Young (Kpa) do material e área da seção transversal (m).

# Definição das condições de contorno
set_bc( dom.nodes[:(x==0 && y==0)] , ux=0, uy=0)
set_bc( dom.nodes[:(x==0 && y==9)] , ux=0, uy=0)
set_bc( dom.nodes[:(x==9 && y==0)] , fy=-450.)  # Carregamentos aplicados (kN)
set_bc( dom.nodes[:(x==18&& y==0)] , fy=-450.)  # Carregamentos aplicados (kN)

# Variável truss representa o domínio
truss = dom
```

```
# Função para selecionar um índice em uma matriz de acordo com os valores
function roulette(norm_fits::Array{Float64,1})
    # n: número de elementos
    n = length(norm_fits)
    # r: # número randômico
    r = rand()
    s = 0.0
    for i=1:n
        s = s + norm_fits[i]
        if r<s
            return i # retorna um indice
        end
    end
    return 1
end

# Função para obter um cromossomo a partir de uma treliça (cromossomo dos nós)
function get_chrom(truss::Domain)
    movable_nodes = [ 5, 6 ]
    cr =Float64[]
    for idx in movable_nodes
        push!(cr, truss.nodes[idx].X[1])
        push!(cr, truss.nodes[idx].X[2])
    end
       #println("cr:  ",cr)
    return cr
end

# Função modifica os genes dentro do cromossomo
function mfactor()
        return -1.0 + 2.0*rand()
end
```

```
# Função gera novos indivíduos por mutação
function mutate(cr, rate=0.5)
    # cr: representa o cromossomo de uma treliça
    n = length(cr)
    # ngens: número de indivíduos a ser mutuados
    ngens = ifloor(n*rate)
    idx = [1:n]
    idx = shuffle(idx)[1:ngens]
    for i in idx
        cr[i] *= 0.02*mfactor()
    end
    return cr
end

# Função gera um novo cromossomo baseado em cromossomos de dois progenitores
function crossover(cr1, cr2)
    # n: número de genes dentro do cromossomo
    n   = length(cr1)
    pos = rand(1:n-1)
    cr  = [ cr1[1:pos], cr2[pos+1:end] ]
    return cr
end

# Função gera um cromossomo para um novo indivíduo
function gen_ind()
    cr = get_chrom(truss)
    return mutate(cr)
end

# Função gera a população
function gen_pop(num::Int)
    # num: representa o número de indivíduos na população
    pop = [ gen_ind() for i=1:num ]  # num : número de indivíduos na população
    return pop
end
```

```
# Função calcula a aptidão para um indivíduo (cromossomo)
function fitness(cr)
    n = length(cr)
    # retribui valores ao cromossomo do individuo
    truss.nodes[5].X[1] = cr[1]
    truss.nodes[5].X[2] = cr[2]
    truss.nodes[6].X[1] = cr[3]
    truss.nodes[6].X[2] = cr[4]

    # cálculo do peso
    weight = 0.0
    gamma  = 25.9  # unit weigth kN/m3
    for elem in truss.elems
        c1 = elem.nodes[1].X
        c2 = elem.nodes[2].X
        L  = norm(c2-c1)
        A  = 0.0171
        weight = weight + A*L*gamma
    end

    # reinicia os deslocamentos e as tensões
    reset(truss.nodes)
    reset(truss.elems)

    # Análise via elementos finitos
    solve!(dom, verbose=false, reset_bc=false)

    # Verifica os deslocamentos
    disps = [ abs(node.dofdict[:uy].U) for node in truss.nodes ]
    max_disp = maximum(disps)
    if max_disp > 0.0508 # displacement (m)
        return -1. # flag
    end

    # Verifica as tensões
    stresses = [ elem.ips[1].data.σ for elem in truss.elems]
    tens_stress = abs(maximum(stresses))
    comp_stress = minimum(stresses)

    if tens_stress>130000.0 # stress (KPa)
        return -1
    end
    return 1./weight
end
```

```
# Retribui valores ao cromossomo do individuo
function set_truss(cr)
    truss.nodes[5].X[1] = cr[1]
    truss.nodes[5].X[2] = cr[2]
    truss.nodes[6].X[1] = cr[3]
    truss.nodes[6].X[2] = cr[4]
end

# Função realiza a evolução
function evolve(n::Int; nger::Int=5, elit::Float64=0.1, mutation::Float64=0.3)
    pop  = gen_pop(n)

    data = DTable( [ :generation, :fitness ] )
    for k=1:nger
        println("\nGeneration: ", k)

        fits = Float64[ fitness(ind) for ind in pop ]
              #println("Fitness: ", fits)

        idxs = sortperm(fits, rev=true)
        # ordena a população de acordo com a aptidão
        pop  = pop[idxs]
        fits = fits[idxs]
        bestd = 1 / fits[1]
        println("Best fitness :", fits[1])
        println("Best disp:", bestd)

        push!(data, [k, fits[1] ] )

        if k%10==0
            save(truss,"truss_$k.vtk")
        end

        if k==nger; break end
        nelit  = ifloor(elit*n)
        ncross = n-nelit
        offspring = Any[]

        # Cruzamento (Roleta)
        nfits = fits/sum(fits)
        for i=1:ncross
            ind1 = pop[ roulette(nfits) ]
            ind2 = pop[ roulette(nfits) ]
            ind  = crossover(ind1, ind2)
```

```
            push!(offspring, ind)
        end

        # Mutação
        nmutate = ifloor(mutation*n)
        for i=1:nmutate
            #pos = rand(nelit+1:n)
            pos = rand(1:ncross)
            ind = offspring[pos]
            mutate(ind)
        end

        pop = [ pop[1:nelit], offspring ]

    end
    save(data, "fitness.dat") # Armazena em disco os resultados da análise
end

# Inicialização do gerador de números aleatórios
srand(0)

# Chamada à função que realiza o processo evolutivo (número de indivíduos, número de
gerações, taxa de elitismo e taxa de mutação)
evolve(90, nger=100, elit=0.1, mutation=0.3)
```