Pontifícia Universidade Católica
do Rio de Janeiro

Alonso Joaquin Juvinao Carbono

Otimização da Disposição de Linhas de Ancoragem Utilizando Algoritmos Genéticos

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Engenharia Civil da PUC-Rio.

Orientadores: Luiz Fernando C. R. Martha
Ivan Fábio Menezes

Rio de Janeiro, setembro de 2005

Pontifícia Universidade Católica
do Rio de Janeiro

Alonso Joaquin Juvinao Carbono

Otimização da Disposição de Linhas de Ancoragem Utilizando Algoritmos Genéticos

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Engenharia Civil da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Luiz Fernando Campos Ramos Martha
Presidente / Orientador
Departamento de Engenharia Civil - PUC-Rio

Dr. Ivan Fábio Mota de Menezes
Co-Orientador
PUC-Rio

Prof. Luiz Eloy Vaz
UFRJ

Prof. Raul Rosas e Silva
Departamento de Engenharia Civil - PUC-Rio

Pedro Colmar Gonçalves da Silva Vellasco
UERJ

Prof. José Eugênio Leal
Coordenador Setorial do Centro Técnico Científico - PUC-Rio

Rio de Janeiro, 30 de Setembro de 2005

Alonso Joaquin Juvinao Carbono

Graduou-se em Engenharia Civil, pela Unicosta CUC – Corporação Universitária da Costa (Colômbia) em 1997. Desenvolveu seu trabalho de pesquisa com ênfase em computação gráfica aplicada.

Ficha Catalográfica

Juvinao Carbono, Alonso Joaquin

Otimização da disposição de linhas de ancoragem utilizando algoritmos genéticos / Alonso Joaquin Juvinao Carbono ; orientadores: Luiz Fernando C. R. Martha, Ivan Fábio Menezes. – Rio de Janeiro : PUC-Rio, Departamento de Engenharia Civil, 2005.

91 f. ; 29,7 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Civil

Inclui bibliografia

1. Engenharia Civil – Teses. 2. Linhas de ancoragem. 3. Otimização estrutural. 4. Algoritmos genéticos. I. Martha, Luiz Fernando C. R. II. Menezes, Ivan Fábio. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Civil. IV. Título.

CDD: 624

Agradecimentos

A Deus, pela sua ajuda diária.

À minha mãe e irmãos, pelo seu amor e apoio.

Aos meus orientadores Luiz Fernando Martha e Ivan Menezes, pela confiança dada durante a realização deste trabalho.

A todos meus amigos da Colômbia e do Brasil, pela força e incentivos dados.

Aos amigos do TecGraf que muito contribuíram, direta ou indiretamente, para o desenvolvimento deste trabalho.

À Rita Cássia e a todos os funcionários e professores do Departamento de Engenharia Civil da PUC.

Ao TecGraf pelo apoio financeiro e tecnológico durante o curso de mestrado.

À CAPES pelo apoio financeiro durante o curso de mestrado.

Resumo

Carbono, Alonso Juvinao; Martha, Luiz Fernando, Menezes, Ivan F. M (Orientadores). **Otimização da Disposição de Linhas de Ancoragem Utilizando Algoritmos Genéticos**. Rio de Janeiro, 2005. 91p. Dissertação de Mestrado – Departamento de Engenharia Civil, Pontifícia Universidade Católica do Rio de Janeiro.

Com o crescimento da demanda de óleo, as empresas de petróleo têm sido forçadas a explorar novas reservas em águas cada vez mais profundas. Em função do alto custo das operações de exploração de petróleo, torna-se necessário o desenvolvimento de tecnologias capazes de aumentar a eficiência e reduzir os custos envolvidos. Neste contexto, a utilização de unidades flutuantes torna-se cada vez mais freqüente em águas profundas. O posicionamento das unidades flutuantes durante as operações de exploração de óleo é garantido pelas linhas de ancoragem, que são estruturas flexíveis compostas, geralmente, por trechos de aço, amarras e/ou cabos sintéticos. O presente trabalho apresenta o desenvolvimento de um Algoritmo Genético (AG) para solucionar o problema da disposição das linhas de ancoragem de unidades flutuantes utilizadas nas operações de exploração de petróleo. A distribuição das linhas de ancoragem é um dos fatores que influencia diretamente nos deslocamentos (*offsets*) sofridos pelas unidades flutuantes quando submetidas às ações ambientais, como ventos, ondas e correntes. Desta forma, o AG busca uma disposição "ótima" das linhas de ancoragem cujo objetivo final é a minimização dos deslocamentos da unidade flutuante. Os operadores básicos utilizados por este algoritmo são mutação, *crossover* e seleção. Neste trabalho, foi adotada a técnica *steady-state*, que só efetua a substituição de um ou dois indivíduos por geração. O cálculo da posição de equilíbrio estático da unidade flutuante é feito aplicando-se a equação da catenária para cada linha de ancoragem com o objetivo de se obterem as forças de restauração na unidade, e empregando-se um processo iterativo para calcular a sua posição final de equilíbrio.

Palavras-chave

Linhas de Ancoragem, Otimização Estrutural, Algoritmos Genéticos.

Abstract

Carbono, Alonso Juvinao; Martha, Luiz Fernando, Menezes, Ivan F. M (Supervisors). **Mooring Pattern Optimization Using Genetic Algorithms**. Rio de Janeiro, 2005. 91p. MSc Dissertation – Civil Engineering Department, Pontifical Catholic University of Rio de Janeiro.

With the increasing demand for oil, oil companies have been forced to exploit new fields in deep waters. Due to the high cost of oil exploitation operations, the development of technologies capable of increasing efficiency and reducing costs is crucial. In this context, the use of floating units in deep waters has become more frequent. The positioning of the floating units during oil exploitation operations is done using mooring lines, which are flexible structures usually made of steel wire, steel chain and/or synthetic cables. This work presents the development of a Genetic Algorithm (GA) procedure to solve the problem of the mooring pattern of floating units used in oil exploitation operations. The distribution of mooring lines is one of the factors that directly influence the displacements (offsets) suffered by floating units when subjected to environmental conditions such as winds, waves and currents. Thus, the GA seeks an optimum distribution of the mooring lines whose final goal is to minimize the units' displacements. The basic operators used in this algorithm are mutation, crossover and selection. In the present work, the steady-state GA has been implemented, which performs the substitution of only one or two individuals per generation. The computation of the floating unit's static equilibrium position is accomplished by applying the catenary equilibrium equation to each mooring line in order to obtain the out-of-balance forces on the unit, and by using an iterative process to compute the final unit equilibrium position.

Keywords

Mooring Pattern, Structural Optimization, Genetic Algorithm.

Sumário

Lista de figuras

Lista de tabelas

1
Introdução

1.1.
Motivação

Com o crescimento da demanda de óleo, as empresas de petróleo têm sido forçadas a explorar novas reservas em águas cada vez mais profundas. No Brasil, a Petrobras extrai atualmente cerca de 17% de sua produção em terra firme, 19% em águas rasas e 64% em águas profundas (acima de mil metros de profundidade). Em função do alto custo das operações de exploração de petróleo, torna-se bastante relevante o desenvolvimento de tecnologias capazes de aumentar a eficiência e reduzir os custos envolvidos. Quando as profundidades envolvidas na exploração eram da ordem de dezenas a uma centena de metros, era possível manter os equipamentos de exploração marinha posicionados por meio de torres treliçadas apoiadas no fundo. Com o aumento da profundidade de exploração, a utilização deste tipo de estrutura se torna inviável. Neste contexto, a utilização de unidades flutuantes ancoradas no fundo do mar é cada vez mais freqüente em águas profundas.

O posicionamento das unidades flutuantes durante as operações de exploração de óleo é garantido pelas linhas de ancoragem, que são estruturas flexíveis compostas geralmente por trechos de aço, amarras e/ou cabos sintéticos. Na extremidade das linhas de ancoragem são utilizadas âncoras e nos trechos intermediários podem ser encontrados alguns acessórios para a conexão de segmentos de materiais diferentes. A distribuição das linhas de ancoragem é um dos fatores que influenciam diretamente os deslocamentos sofridos pelas unidades flutuantes quando submetidas a ações ambientais, tais como ventos, ondas e correntes.

De acordo com a geometria das linhas, diferentes configurações são utilizadas na ancoragem de estruturas flutuantes, como ancoragens em catenária (convencionais), ancoragem tipo *taut-leg* (linhas inclinadas tracionadas) e ancoragem vertical (utilizando tendões). O critério de escolha do tipo de ancoragem depende principalmente do tipo de embarcação, da lâmina d'água, da quantidade de *risers*, do tipo de operação e do custo envolvido.

1.2. Objetivo

A determinação da “melhor” disposição possível das linhas de ancoragem resulta em um problema de otimização cujo objetivo final é minimizar os deslocamentos (*offsets*) das unidades flutuantes quando submetidas a diferentes combinações de forças externas provenientes das ações ambientais.

O objetivo do presente trabalho é o desenvolvimento de um Algoritmo Genético (AG) para solucionar este problema de otimização.

Os algoritmos genéticos, pertencentes ao campo da computação evolucionária, são inspirados no processo de seleção natural e se distinguem das técnicas mais comuns de programação matemática por empregarem uma população de indivíduos ou soluções; trabalharem sobre uma codificação das possíveis soluções ao invés das soluções propriamente ditas; empregarem regras de transição probabilísticas; e não requererem informações adicionais (como derivadas da função objetivo, por exemplo). Os operadores básicos utilizados por estes algoritmos são mutação, *crossover* e seleção. Existe uma grande variedade de processos de seleção dos indivíduos. Neste trabalho, é adotada a técnica *steady-state*, que só efetua a substituição de dois indivíduos por geração. Neste caso, os dois melhores indivíduos da geração corrente são recombinados geneticamente e os indivíduos resultantes são usados para substituir os dois piores.

O cálculo da posição de equilíbrio estático da unidade flutuante é realizado aplicando-se a equação da catenária para cada linha da ancoragem com o objetivo de se obterem as forças de restauração na unidade, e empregando-se um processo iterativo para calcular a sua posição final de equilíbrio.

Recentemente, alguns trabalhos foram desenvolvidos sobre a otimização de sistemas de ancoragem. Albrecht [1] implementou uma ferramenta computacional para a otimização de sistemas de ancoragem, empregados nos processos de exploração de petróleo, em função dos deslocamentos sofridos pelas unidades flutuantes e das tensões nas linhas de ancoragem. Nesta pesquisa, o autor utiliza um algoritmo baseado nos princípios da computação evolucionária. Maffra *et. al.* [2] apresentaram um trabalho com a aplicação de algoritmos genéticos para a otimização dos deslocamentos sofridos por unidades flutuantes submetidas a um conjunto de condições ambientais. Apesar dos resultados obtidos serem interessantes, os autores enfatizam a necessidade de melhoria do algoritmo utilizado. Santos [3] introduziu uma ferramenta

computacional baseada em uma estratégia evolutiva para otimizar os sistemas de ancoragem de unidades de produção do tipo FPSO (*Floating Production, Storage and Offloading Vessel*).

1.3. Organização do trabalho

No capítulo 2 são apresentados os principais tipos de sistemas de ancoragem utilizados em estruturas *offshore*.

O capítulo 3 mostra alguns métodos de otimização, dando-se maior ênfase aos algoritmos genéticos. Uma revisão bibliográfica sobre diversos trabalhos encontrados na literatura técnica também é apresentada neste capítulo.

A formulação matemática do problema proposto, assim como a descrição detalhada da implementação de um algoritmo genético e sua aplicação na solução de diversos exemplos clássicos encontrados na literatura técnica, são mostradas no capítulo 4.

O capítulo 5 apresenta os resultados da aplicação do algoritmo proposto em exemplos reais de sistemas de ancoragem.

As conclusões deste trabalho são apresentadas no capítulo 6, juntamente com algumas recomendações para trabalhos futuros.

2
Sistemas de Produção Offshore

2.1.
Introdução

O conjunto de equipamentos utilizados para a prospecção e exploração marinha de petróleo é conhecido como *Sistema Offshore* e compreende basicamente quatro grupos: o casco, as linhas, os equipamentos submarinos e os poços [1].

Com o avanço da exploração e produção de petróleo em águas profundas, o uso de unidades flutuantes torna-se cada vez mais freqüente e seu posicionamento numa determinada área durante algum tipo de operação passa a ser garantido por meio da utilização de estruturas esbeltas chamadas *linhas de ancoragem*. A ancoragem pode ter uma composição homogênea ou heterogênea (mais usada em águas profundas, visando minimizar o peso suspenso), sendo comumente formada por amarras, cabos de aço e/ou cabos sintéticos (por exemplo, de poliéster).

Neste capítulo, são apresentados, de forma sucinta, os principais componentes dos sistemas de ancoragem e algumas configurações possíveis de linhas para a utilização na ancoragem de estruturas *offshore*.

2.2.
Casco

Com a descoberta de novos poços e reservatórios de petróleo em profundidades cada vez mais elevadas, foi necessária a utilização de novas unidades de produção, denominadas unidades flutuantes, as quais podem ser classificadas em plataformas semi-submersíveis, navios (FPSO), pernas atirantadas (TLP – *Tension Leg Press*) e SPAR [1].

2.2.1.
Plataforma Semi-submersível

As plataformas semi-submersíveis (Figura 2.1), compostas por estruturas flutuantes, são largamente empregadas na produção e perfuração. Consistem de um casco compartilhado e composto por flutuadores e colunas. Os flutuadores, também conhecidos como *pontoons*, apóiam as colunas, também chamadas de *pernas*, e que por sua vez sustentam os conveses.

Uma das principais desvantagens deste tipo de plataforma é o fato de não serem adequadas para o armazenamento do óleo produzido durante o processo de exploração, necessitando assim de meios para a exportação do óleo [1].

Figura 2.1 – Plataforma semi-submersível [1].

2.2.2.
Navios

Diferentemente das plataformas semi-submersíveis, os navios (Figura 2.2) são capazes de armazenar o óleo produzido durante o processo de exploração. Também conhecidos como *unidades de produção* ou pelo termo FPSO (*Floating Production, Storage and Offloading Vessel*), os navios são muitas vezes utilizados como suporte de outras unidades (plataformas) para armazenar e transportar o óleo; neste caso adquire o nome de FSO (*Floating Storage and Offloading Vessel*) [1].

Figura 2.2 – Navio FPSO [1].

2.2.3.
TLP – *Tension Leg Platform*

A TLP (Figura 2.3) consiste em uma estrutura com um casco semelhante ao da plataforma semi-submersível. Quando comparada com outros cascos, a TLP apresenta movimentos menores, o que possibilita que a completação dos poços seja do tipo seca, ou seja, o controle e a intervenção nos poços são feitos na plataforma e não no fundo do mar, o que representa uma diminuição nos custos de instalação e produção dos poços [1].

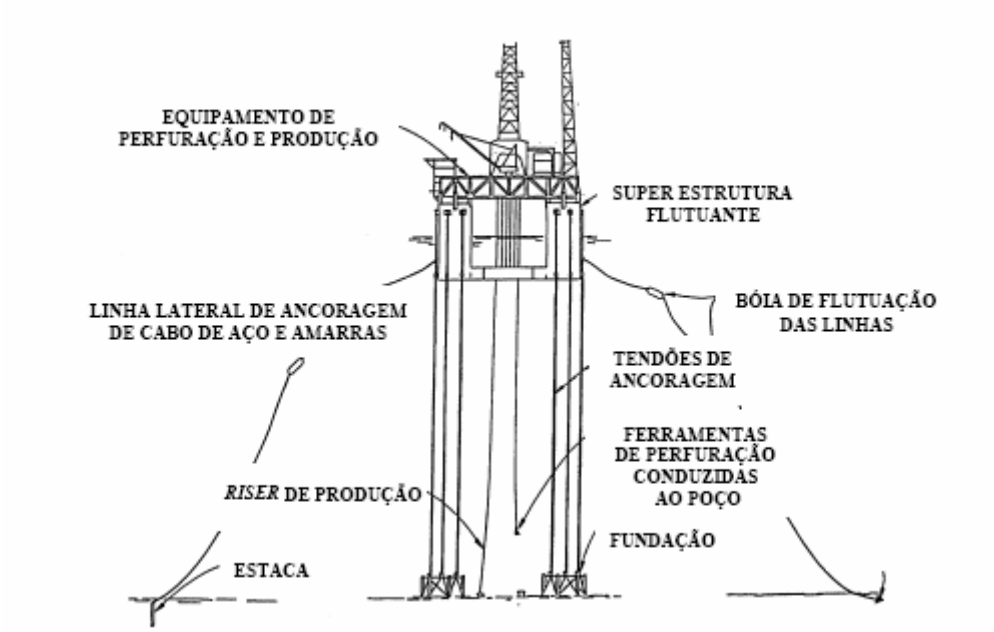


Figura 2.3 – Unidade TLP [1].

2.2.4.
SPAR

O SPAR consiste em um único cilindro vertical de aço, de grande diâmetro, ancorado, que opera com um calado de profundidade constante de cerca de 200 metros, o que gera pequenos movimentos verticais e, portanto, possibilita o uso de *risers* rígidos de produção [1].

2.3.
Linhas de Ancoragem

As linhas de ancoragem são estruturas esbeltas dispostas em catenária, *taut-leg* ou tendões. Sua principal função é fornecer as forças de restauração que mantêm em posição as unidades flutuantes.

Os materiais mais utilizados na construção das linhas de ancoragem são amarras de aço, cabos de aço e, mais recentemente, cabos de poliéster [1]. As amarras são utilizadas geralmente nos trechos iniciais e finais das linhas, já que este material é mais resistente ao atrito com o fundo do mar e com os guinchos das unidades flutuantes.

2.3.1.
Amarras

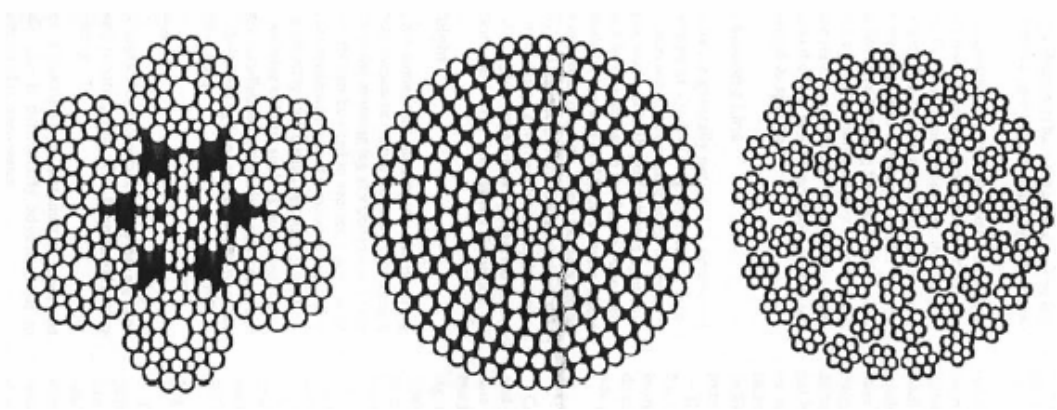
De acordo com Albrecht [1], os tipos de amarras mais utilizados na ancoragem de unidades flutuantes são os que possuem elos com malhete. São diversos os tipos de elo que existem para unir duas partes de corrente, dos quais o mais usado é o *Kenter*. Uma das principais desvantagens destes elementos de união é que, apesar de possuírem uma carga de ruptura igual, e em certas ocasiões superior, à de uma corrente da mesma dimensão, a durabilidade à fadiga é sensivelmente menor. É por isto que o número de elos de ligação utilizado nas linhas de ancoragem deve ser mínimo.

2.3.2.
Cabos de Aço

São vários os tipos de cabos de aço utilizados na ancoragem de unidades flutuantes, sendo os principais os *six strand* e os *spiral strand*. Os primeiros, por apresentarem um fácil manuseio, são empregados com maior freqüência em

unidades de perfuração. No entanto, os segundos possuem alta resistência e durabilidade, sendo mais comumente utilizados em unidades de produção.

Devido à corrosão da trança metálica, o cabos apresentam uma vida útil inferior à das amarras, mas este problema pode ser contornado com a utilização de cabos galvanizados. Com relação à resistência dos arames que formam o cabo, são utilizados normalmente os tipos IPS (*Improved Plow Steel*) e EIPS (*Extra Improved Plow Steel*) [1]. Os cabos com este último tipo de arame são mais resistentes à tração e, portanto, mais recomendados nos sistemas de unidades flutuantes. A Figura 2.4 ilustra os tipos de cabos de aço mais utilizados nos sistemas de ancoragem.



(a) Six Strand Rope (b) Spiral Strand (c) Multi – Strand

Figura 2.4 – Cabos de aço [1]

2.3.3. Cabos de Poliéster

Geralmente, são classificados como cabos todos os cabos sintéticos e, como cordas, todos os cabos e cordas feitos de qualquer tipo de fibra [1]. As fibras mais utilizadas para a fabricação dos cabos são: polietileno, sisal, poliamida (comercialmente conhecida como *nylon*), polipropileno, poliaramida (com grande módulo elasticidade), HMPE (*High Modulus Polyethylene*) e poliéster. Este último é muito utilizado em sistemas de ancoragem, esperando-se que atinja uma vida útil de até 20 anos. A Figura 2.5 mostra um típico cabo de poliéster utilizado em ancoragens de unidades flutuantes.

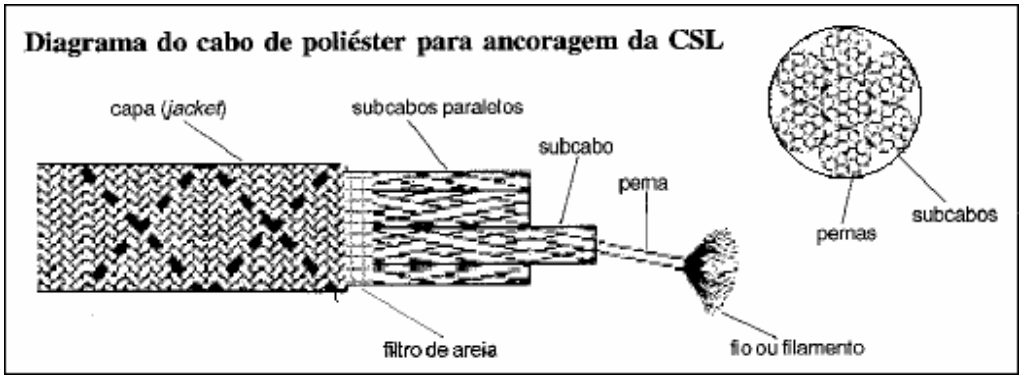


Figura 2.5 – Cabo de poliéster [1].

2.4. Tipos de Ancoragem

Os tipos de ancoragem mais utilizados nas operações de exploração de petróleo por estruturas *offshore* são descritos nas sub-seções seguintes [4].

2.4.1. Ancoragem em Catenária

A ancoragem em catenária é a técnica convencional utilizada em operações de produção ou perfuração, com a vantagem de possibilitar maiores passeios da embarcação sem a necessidade de âncoras com elevado poder de garra. O fato de possuir um raio de ancoragem razoavelmente grande (superior a 1000 metros) e o próprio atrito do trecho de linha encostado no fundo são responsáveis por absorver as solicitações do carregamento ambiental, aliviando os esforços nas âncoras, em condições normais de operação. Sua principal desvantagem é o congestionamento com as linhas de unidades próximas, que interfere diretamente no posicionamento das unidades, além da interferência de linhas com equipamentos submarinos.

2.4.2. Ancoragem em Taut-Leg

Para contornar as desvantagens do sistema em catenária utiliza-se a ancoragem em *taut-leg*. Neste sistema, a linha se encontra mais tensionada, com um ângulo de topo de aproximadamente 45° com a vertical, tendo assim uma projeção horizontal menor, para uma mesma ordem de grandeza da lâmina d'água (Figura 2.6). Este tipo de ancoragem proporciona maior rigidez ao sistema, sendo o passeio da embarcação limitado a *offset*s menores. Neste

caso, as âncoras a serem utilizadas precisam resistir a valores elevados de cargas verticais. A ancoragem *taut-leg* é geralmente utilizada em sistemas localizados em regiões de grandes profundidades.

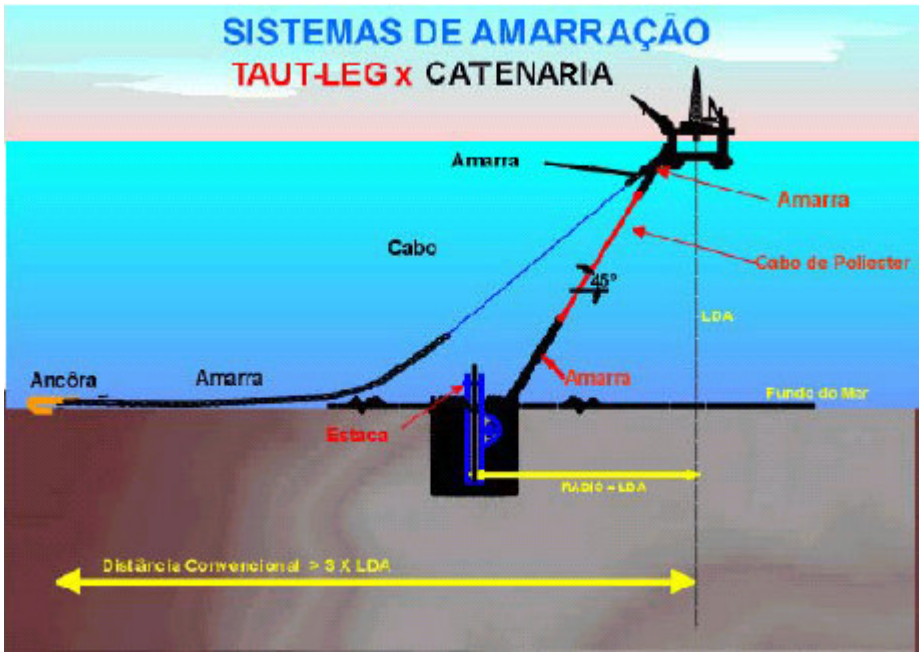


Figura 2.6 – Ancoragem em catenária x Ancoragem *taut-leg* [1].

2.4.3. Ancoragem Vertical

É baseada na utilização de tendões verticais, que precisam estar sempre tracionados devido ao excesso de empuxo proveniente da parte submersa da embarcação. Trata-se da ancoragem usada principalmente em plataformas TLP (*Tension Leg Platform*), mas que também pode ser adotada por bóias e monobóias, dentre outras. Os tendões podem ser de cabo de aço ou de material sintético, proporcionando uma elevada rigidez no plano vertical e baixa rigidez no plano horizontal. A força de restauração no plano horizontal é fornecida pela componente horizontal da força de tração nos tendões. Para tendões de pequenos diâmetros ($d \approx 0.25$ m) os efeitos de flexão podem ser desprezados, enquanto que, para grandes diâmetros ($d \approx 1.00$ m), tais efeitos devem ser considerados. A Figura 2.7 ilustra este tipo de ancoragem.

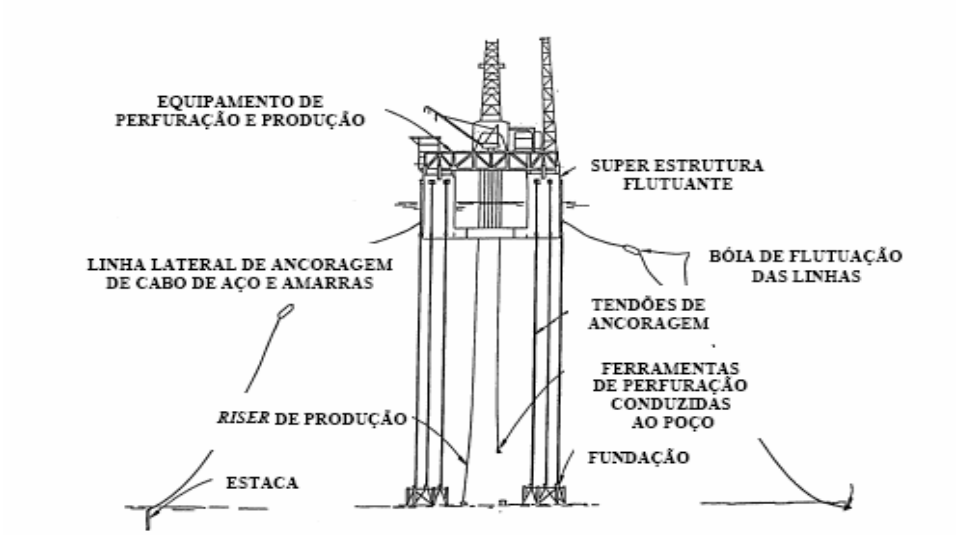
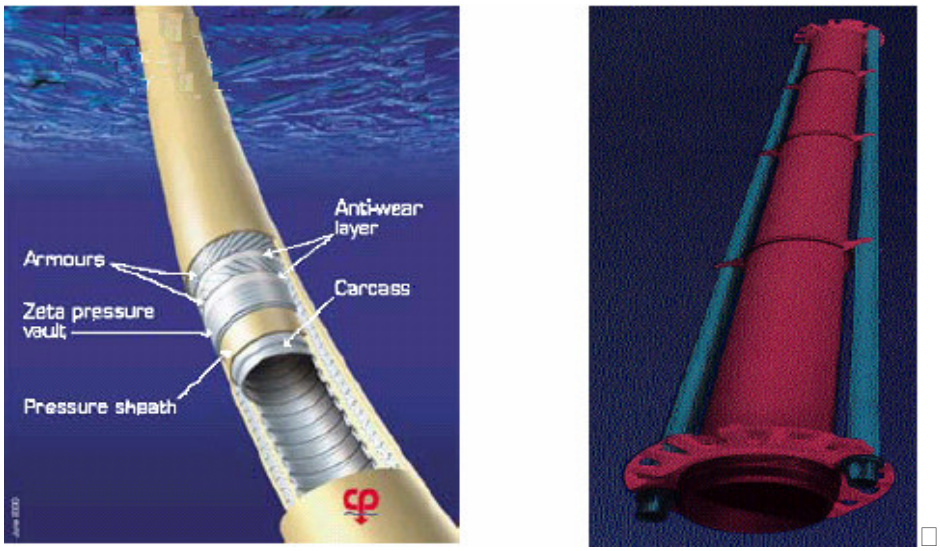


Figura 2.7 – Ancoragem Vertical – TLP [1].

2.5.
Risers

Os *risers* são tubulações verticais, geralmente dispostos em configurações de catenária, utilizados para transportar o óleo ou o gás do fundo do mar até a superfície. Podendo ser flexíveis, compostos por camadas de plástico e aço, ou rígidos, compostos de aço, os *risers* também são usados para injetar a água necessária para estimular o poço durante o processo de exploração [1]

A Figura 2.8 ilustra os dois tipos básicos de *risers* utilizados no processo de exploração de petróleo.



(a) *Riser* Flexível **(b) *Riser* Rígido**

Figura 2.8 – Exemplos de *Risers*: (a) Flexível e (b) Rígido [1].

2.6. Sistemas de Ancoragem

Uma vez que o sistema de ancoragem se encontra sob a ação do carregamento ambiental, é normal que a unidade sofra um passeio ou deslocamento, o qual corresponde à distância horizontal que a unidade percorre desde sua posição inicial de equilíbrio neutro até a posição final de equilíbrio sob a ação das cargas [1]. O passeio é medido como um percentual da lâmina d'água.

Uma medida da eficiência do sistema de ancoragem é a magnitude do deslocamento da unidade flutuante. O passeio é inversamente proporcional à rigidez do sistema de ancoragem; assim, quanto mais alta a rigidez menor será o passeio, porém o tipo de material utilizado para construir as linhas de ancoragem determinará o limite superior da rigidez do sistema. É importante salientar que, ao se aumentar a rigidez do sistema, estão se aumentando as tensões aplicadas às linhas e, portanto, as tensões internas, as quais devem respeitar certos limites de segurança [1].

Para um conjunto de dados ambientais (ventos, ondas e correntes) é necessário estabelecer as correspondências entre eles e suas probabilidades de ocorrência. De acordo com Albrecht [1], são duas as condições ambientais tipicamente aplicadas em projetos de ancoragem: condição máxima de projeto e condição máxima de operação.

A condição máxima de projeto é a condição extrema que o sistema deve suportar sem danos e é definida como uma combinação de ventos, ondas e correntes para a qual o sistema deve ser projetado. Para isto é necessário definir o tipo de ancoragem que será utilizado, dependendo do tempo que o sistema permanecerá em operação. Isto é, podem ser utilizadas ancoragens permanentes ou provisórias – a primeira delas considera um período de 100 anos para a recorrência de um evento, ou seja, o sistema deve ser projetado para um carregamento ambiental que poderá ocorrer uma vez a cada 100 anos. Já a ancoragem provisória é utilizada em sistemas que permanecerão em operação por um período inferior a 5 anos em uma locação.

A condição máxima de operação é a combinação máxima de condições ambientais (ventos, ondas e correntes) sob a qual o sistema deverá operar, seja em atividades de produção ou perfuração.

A seguir, três diferentes sistemas de ancoragem utilizados em estruturas flutuantes serão apresentados: ancoragem com ponto único SPM (*Single Point*

Mooring), amarração com quadro de ancoragem SM (*Spread Mooring*) e ancoragem com posicionamento dinâmico DP (*Dynamic Position*).

2.6.1. Ancoragem com Ponto Único

A ancoragem SPM é mais freqüentemente utilizada por navios petroleiros convertidos em FSO's (*Floating Storage and Offloading Units*) ou FPSO's (*Floating Production, Storage and Offloading Units*). Ela permite que a embarcação se alinhe com o carregamento ambiental, minimizando as forças sobre o casco. Como exemplos, temos: ancoragem com *turret*, CALM (*Catenary Anchor Leg Mooring*) e SALM (*Single Anchor Leg Mooring*).

No sistema de ancoragem com *turret,* todas as linhas de ancoragem e *risers* são presos no *turret* que, essencialmente, faz parte da estrutura a ser ancorada. O *turret* permite que a embarcação gire livremente em torno das linhas e pode ser montado interna ou externamente à embarcação. (Figura 2.9).

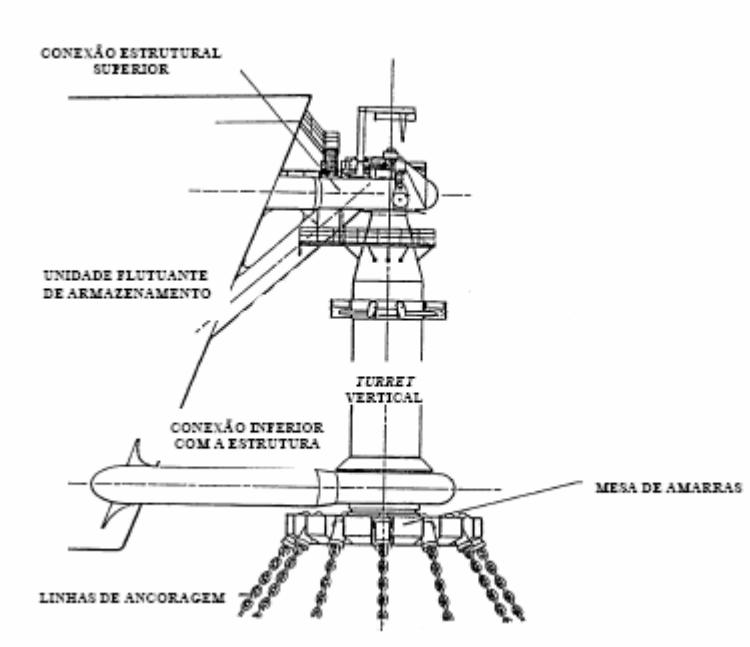


Figura 2.9 – *Turret* externo [1].

O sistema CALM (Figura 2.10) consiste em uma bóia de grandes dimensões que suporta um determinado número de linhas de ancoragem em catenária. Os *risers* são presos na parte inferior da bóia e utilizam um cabo sintético para fazer a amarração entre a bóia e o navio. Uma desvantagem deste sistema é a limitação de sua capacidade de resistir às condições ambientais quando a reação da bóia é razoavelmente diferente da resposta do navio sob

influência das ondas. Assim, quando as condições do mar atingem uma certa magnitude, torna-se necessário desconectar o navio.

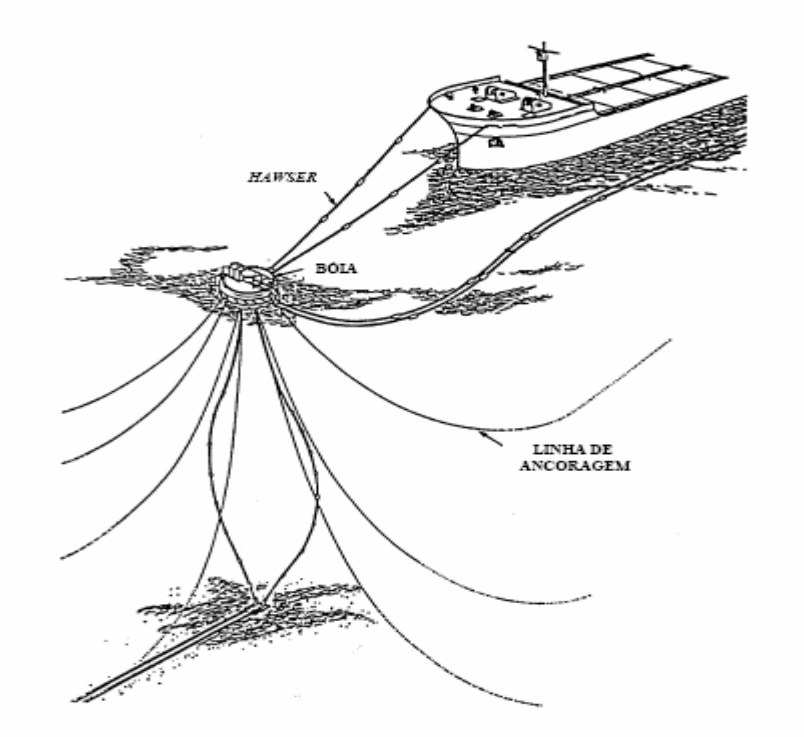


Figura 2.10 – Ancoragem tipo CALM com *hawser* [1].

O sistema SALM utiliza um *riser* vertical que tem uma elevada capacidade de flutuação próximo à superfície e, algumas vezes, na superfície, é mantido por um *riser* pré-tensionado. O sistema basicamente emprega um *riser* articulado com uma forquilha rígida de acoplamento (Figura 2.11).

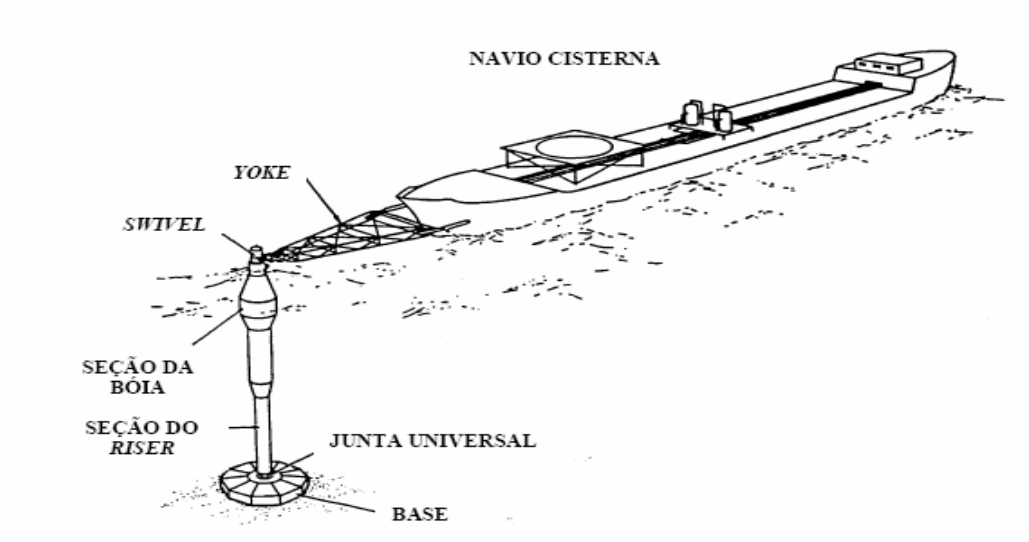


Figura 2.11 – SALM com *riser* e *yoke* [1].

2.6.2.
Amarração com Quadro de Ancoragem (SM)

Este tipo de amarração é mais freqüentemente utilizada por plataformas semi-submersíveis em operações de perfuração e produção. Suas linhas de ancoragem se encontram distribuídas em torno da embarcação (Figura 2.12), tornando-a capaz de resistir a carregamentos ambientais. Assim, a unidade flutuante poderá resistir às cargas ambientais independentemente das direções de atuação.

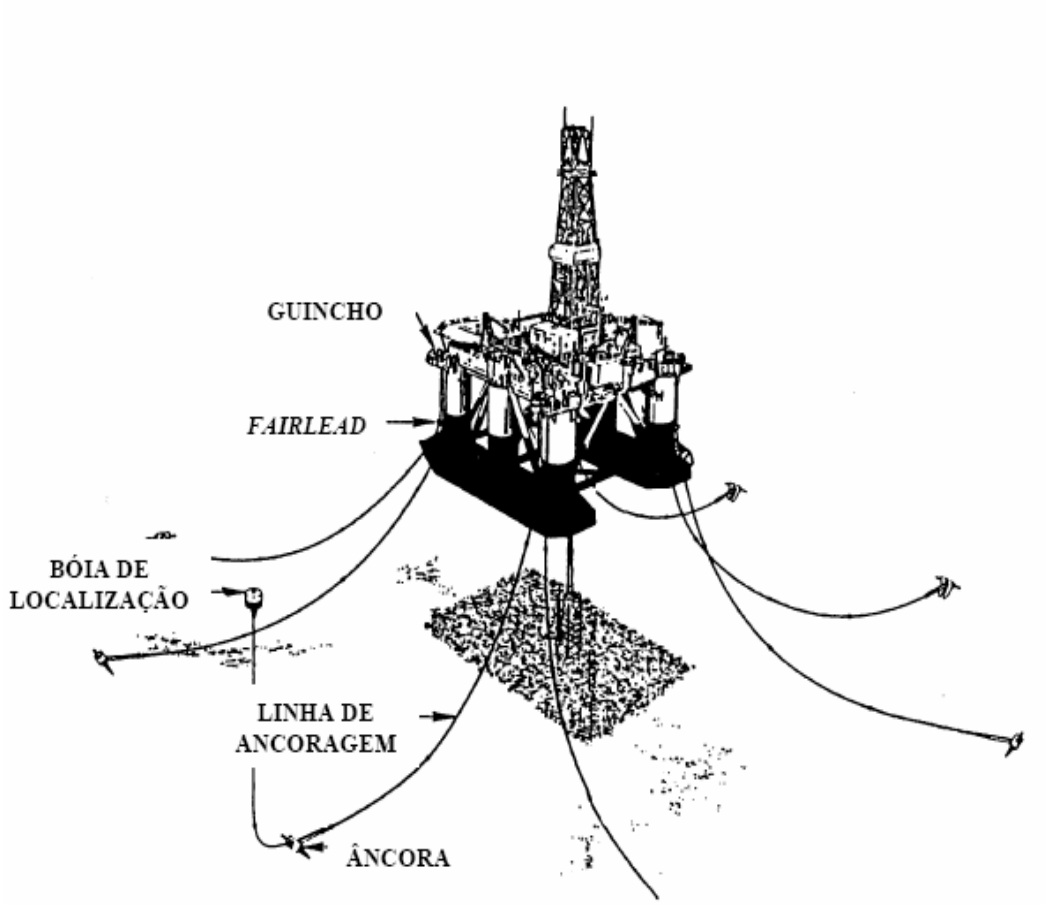


Figura 2.12 – Plataforma semi-submersível com amarração de quadro de ancoragem [1].

Uma concepção recente de ancoragem para navios consiste na adoção de linhas distribuídas, como no caso das plataformas semi-submersíveis, apesar de navios sofrerem maior influência em relação às direções dos carregamentos. Este sistema é conhecido como DICAS (*Differentiated Compliance Anchoring System*) [5] e fornece um alinhamento parcial com a pior direção do carregamento ambiental.

O sistema DICAS, desenvolvido pela Petrobras para produção e armazenamento em área *offshore* brasileira, consiste em um conjunto de linhas

de ancoragem com conexões na proa e na popa do navio [5]. A localização das linhas do sistema de ancoragem DICAS proporciona a existência de diferentes níveis de rigidez na popa e na proa do navio. Tal diferença de rigidez é obtida por meio de uma variação nas pré-tensões das linhas, o que permite ao navio adquirir uma configuração (aproamento) adequada às características ambientais. Diferentes níveis de pré-tensões das linhas irão conduzir a diferentes ângulos críticos de incidência, resultando num melhor posicionamento do navio em relação às mais freqüentes direções de incidência das condições ambientais. A Figura 2.13 mostra uma vista 3D de um sistema DICAS.

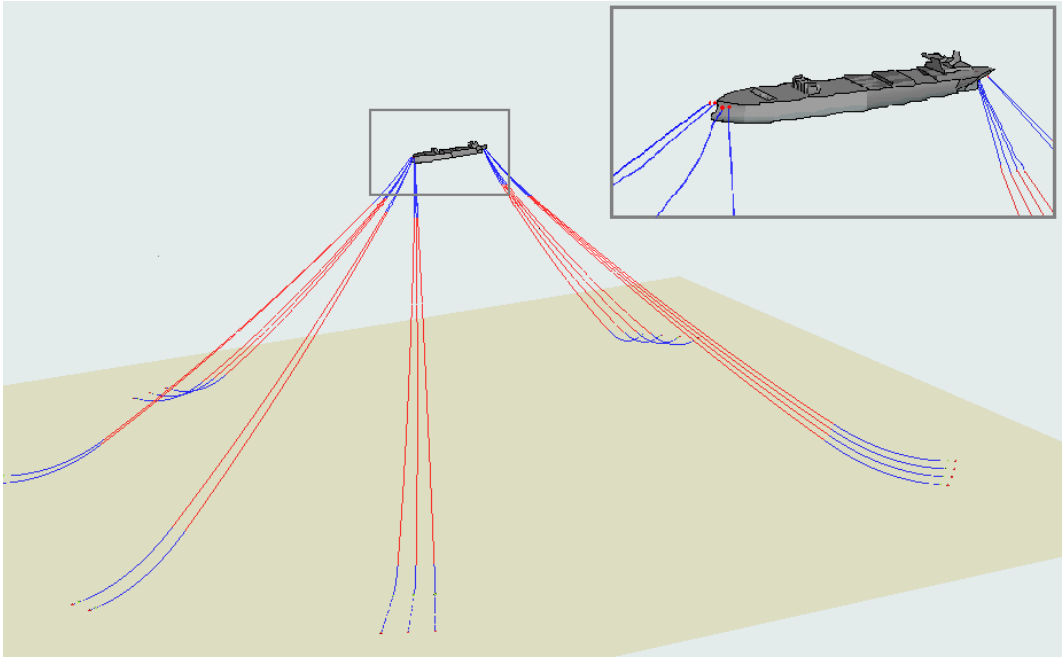


Figura 2.13 – Vista 3D de um sistema DICAS [1].

2.6.3. Ancoragem com Posicionamento Dinâmico

O sistema DP (*Dynamic Position*) pode ser utilizado de forma isolada ou como auxílio para um sistema já ancorado. Trata-se de um tipo de ancoragem utilizado em atividades de perfuração e intervenção em poços de petróleo. As unidades DP (Figura 2.14) podem ser constituídas de navios ou plataformas semi-submersíveis que mantêm sua posição com o auxílio de um conjunto de propulsores. Quando tais unidades operam muito próximas a outras já ancoradas, pode-se fazer necessária a utilização de âncoras de segurança, para o caso de sofrerem alguma falha na geração de energia para os propulsores.

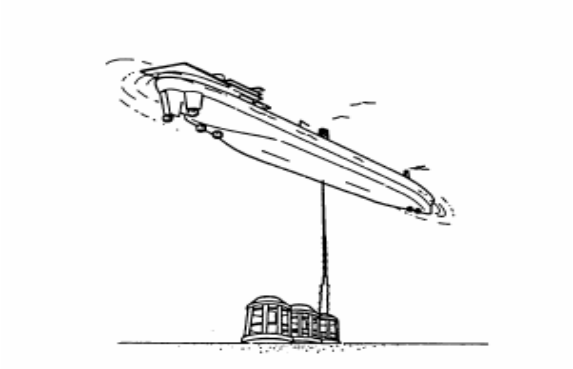


Figura 2.14 – Sistema de ancoragem DP [1].

3
Métodos de Otimização

3.1.
Introdução

Neste capítulo são apresentados alguns conceitos básicos sobre otimização, bem como uma breve descrição dos principais métodos de programação matemática, computação natural e computação evolucionária.

São abordados de forma mais detalhada os algoritmos genéticos, procedimentos, configurações, técnicas básicas, operadores, vantagens e desvantagens, comparando-os com outros métodos convencionais.

3.2.
Otimização

A otimização pode ser definida como o conjunto de procedimentos por meio dos quais se busca minimizar ou maximizar uma determinada função, denominada *função objetivo*, sujeita ou não a restrições de igualdade, desigualdade e restrições laterais, obtendo assim um melhor aproveitamento dos recursos disponíveis [6].

A função objetivo e as funções de restrições podem ser lineares ou não-lineares em relação às variáveis de projeto e, por esta razão, os métodos de otimização podem ser divididos em dois grandes grupos: *programação linear e programação não-linear.*

3.2.1.
Programação Linear

Tem como objetivo encontrar a solução ótima em problemas nos quais a função objetivo e todas as restrições são representadas por funções lineares das variáveis do projeto. Segundo Olivieri [6], qualquer problema linear pode ser representado por uma “formulação padrão”, ou seja:

Minimizar: $$F = c_1 x_1 + c_2 x_2 + ... + c_n x_n$$

Sujeita a: $$a_{11} x_1 + a_{12} x_2 + ... + a_{1n} x_n = b_1$$

$$a_{21} x_1 + a_{22} x_2 + ... + a_{2n} x_n = b_2$$

$$. \qquad . \qquad . \qquad .$$

$$a_{m1} x_1 + a_{m2} x_2 + .. + a_{mn} x_n = b_m$$

e $$x_1 \geq 0, x \geq 0, ..., x_n \geq 0$$

onde F é a função objetivo, x_i são as variáveis de projeto ou incógnitas, e b_i, c_i, e a_i são as constantes do problema.

É importante observar que a maioria dos problemas de otimização, encontrados na Engenharia, não podem ser representados por funções lineares das variáveis de projeto.

3.2.2. Programação Não-Linear

Trata dos problemas em que a função objetivo ou algumas das restrições do problema são funções não-lineares das variáveis envolvidas. Em geral, os métodos de programação não-linear são classificados em dois sub-grupos: métodos determinísticos e não-determinísticos [7].

Nos *métodos determinísticos*, também conhecidos como *métodos clássicos*, a busca do ponto ótimo utiliza as coordenadas da posição corrente ($\boldsymbol{x}_k$) como ponto de partida para a iteração seguinte (k + 1). A solução de problemas sem restrições consiste em se aplicar, de forma iterativa, a equação:

$$\boldsymbol{x}_{k+1} \approx \boldsymbol{x}_k + \lambda_k \boldsymbol{d}_k \qquad (3.1)$$

onde λ_k é o passo de cálculo e $\boldsymbol{d}_k$ é a direção de busca do ponto ótimo. Em geral, a obtenção da direção de busca envolve o cálculo analítico de derivadas[1] (ou aproximações numéricas destas) cuja ordem caracteriza o método utilizado (i.e., métodos de ordem um ou métodos de ordem superior).

[1] Alguns métodos clássicos, conhecidos como métodos de ordem zero, não utilizam o cálculo de derivadas na obtenção da direção de busca ($\boldsymbol{d}_k$).

O passo de cálculo controla a evolução da solução e seu valor pode ser obtido por métodos do tipo g*olden section* (seção áurea) e Fibonacci, dentre outros. A diferença entre os métodos de programação não-linear consiste na estratégia adotada para determinação do vetor $\boldsymbol{d}_k$. Dentre os métodos de otimização sem restrições conhecidos na literatura técnica, destacam-se: o Método do Máximo Declive, o Método de Newton-Raphson, o Método dos Gradientes Conjugados e os quase-Newton [7]. Com relação à minimização com restrições, destacam-se: o Método das Penalidades, o Método das Barreiras, o Método de Programação Quadrática Seqüencial e o Método do Lagrangeano Aumentado [7].

Os *métodos não-determinísticos* são aqueles que procuram imitar fenômenos ou processos encontrados na natureza e, por esse motivo, pertencem a uma área denominada c*omputação natural*. Uma grande variedade de técnicas se destaca neste conjunto de métodos, como por exemplo a inteligência computacional e suas sub-áreas [7].

3.3. Computação Natural

A computação natural procura criar sistemas inteligentes que reproduzam aspectos do comportamento humano, tais como percepção, raciocínio, adaptação e aprendizado [8].

A inteligência artificial é o ramo do conhecimento humano que se propõe a entender e a construir entidades inteligentes que apresentem as mesmas capacidades das entidades inteligentes encontradas no mundo real [9].

A computação natural compreende uma grande variedade de sub-áreas empregadas na otimização de problemas, dentre elas: fractais, recozimento simulado, lógica nebulosa, redes neurais artificiais e a computação evolucionária.

3.3.1. Recozimento Simulado

O recozimento simulado (*simulated* a*nnealing*) é uma técnica de otimização que utiliza o princípio de evolução da solução ao longo do tempo. O termo *recozimento* refere-se à forma como os metais líquidos são resfriados vagarosamente de modo a garantir baixa energia e formatos de estrutura altamente sólidos. Por garantir um alto nível de movimentação por meio do

espaço de busca, o recozimento simulado procura varrer todo esse espaço de forma a permitir uma solução global. Mais adiante no processo, o resfriamento permitirá apenas pequenos movimentos no espaço de soluções, e o processo convergirá para uma solução final. A natureza dos movimentos ao longo do processo indica que, uma vez que o sistema "resfrie", a solução terá sido movida para uma área de menor “energia” [10].

O recozimento simulado necessita dos seguintes elementos para o processamento: uma descrição das possíveis soluções, um gerador de alterações randômicas entre as soluções, uma função objetivo para as soluções, um parâmetro de controle e, finalmente, um "escalonamento de recozimento" que descreva como o parâmetro de controle varia ao longo do tempo.

3.3.2. Fractais

Os fractais são formas geométricas abstratas de rara beleza, com padrões complexos que se repetem infinitamente, mesmo limitados a uma área finita [11]. Mandelbrot [12] constatou ainda que todas essas formas e padrões possuíam algumas características comuns e que havia uma curiosa e interessante relação entre estes objetos e aqueles encontrados na natureza.

Um fractal é gerado a partir de uma fórmula matemática, muitas vezes simples, mas que aplicada de forma iterativa produz resultados fascinantes e impressionantes.

Existem duas categorias de fractais: os geométricos, que repetem continuamente um modelo padrão, e os aleatórios, que são gerados computacionalmente. Além de se apresentarem como formas geométricas, os fractais representam funções reais ou complexas e apresentam como características auto-semelhança, dimensionalidade e complexidade infinita.

3.3.3. Lógica Nebulosa

O conceito de lógica nebulosa *(fuzzy logic)* foi introduzido em 1965 por Lotfi A. Zadeh [13] na Universidade da Califórnia, em Berkeley. A ele é atribuído o reconhecimento como grande colaborador do controle moderno. Em meados da década de 60, Zadeh observou que os recursos tecnológicos disponíveis eram incapazes de automatizar as atividades relacionadas a problemas de natureza industrial, biológica ou química que demandavam a compreensão de

situações ambíguas, não passíveis de processamento através da lógica computacional fundamentada na lógica booleana. Procurando solucionar esses problemas, Zadeh publicou, em 1965, um artigo resumindo os conceitos dos conjuntos *fuzzy*, revolucionando o assunto com a criação de sistemas de lógica nebulosa. Em 1974, Mamdani, da universidade de Queen Mary de Londres, após inúmeras tentativas frustradas de controlar uma máquina a vapor com diferentes tipos de controladores, somente conseguiu fazê-lo através da aplicação do raciocínio *fuzzy* [13].

3.3.4. Redes Neurais Artificiais

Segundo De Castro [8], as redes neurais são inspiradas na estrutura do cérebro humano e têm o objetivo de apresentar características similares a ele, tais como: aprendizado, associação, generalização e abstração. Elas são compostas por diversos elementos, como os processadores (ou neurônios artificiais), altamente interconectados, que efetuam um número pequeno de operações simples e transmitem seus resultados aos processadores vizinhos [14].

Devido à sua estrutura, as redes neurais são bastante efetivas no aprendizado de padrões a partir de dados não-lineares, incompletos, com ruídos e até mesmo compostos de exemplos contraditórios. Algumas de suas aplicações são reconhecimento de padrões (imagem, texto e voz), previsão de séries temporais e modelagens de problemas específicos.

3.3.5. Computação Evolucionária

A computação evolucionária compreende diversos algoritmos inspirados na genética e no princípio Darwiniano da evolução das espécies. São algoritmos probabilísticos que fornecem um mecanismo de busca paralela e adaptativa baseado no princípio da sobrevivência dos mais aptos e na reprodução. O mecanismo é obtido a partir de uma população de indivíduos (soluções), representados por cromossomos (palavras binárias, vetores ou matrizes), cada um associado a uma aptidão (avaliação da solução do problema), que são submetidos a um processo de evolução (seleção, reprodução, cruzamento e mutação) por vários ciclos. A idéia é a evolução de uma população de estruturas computacionais, de tal modo que se possa melhorar a adequação dos indivíduos

que formam esta população em relação ao ambiente a que ela está submetida [15].

Diferentes tipos de problemas podem ser resolvidos pela computação evolucionária. Muitos problemas são de otimização (numérica ou combinatória), outros são de síntese de um objeto (programa de computador, circuito eletrônico) e, em outros contextos, procura-se um modelo que reproduza o comportamento de determinado fenômeno (*machine learning*). Para vários desses problemas, é freqüentemente possível encontrar um algoritmo que ofereça uma solução "ótima" ou uma boa aproximação desta solução. Enquanto alguns algoritmos requerem informações auxiliares, como derivadas, que muitas vezes não estão disponíveis ou são difíceis de se obter, a computação evolucionária dispensa informações auxiliares e oferece algoritmos gerais (i.e., algoritmos genéticos, programação genética, estratégias evolutivas e programação evolutiva) que são aplicados em problemas complexos, com grandes espaços de busca, de difícil modelagem, ou para os quais não há um algoritmo eficiente disponível.

A idéia por trás de cada uma dessas técnicas é a mesma. Tipicamente, os candidatos são representados por números binários nos algoritmos genéticos, vetores de números reais nas estratégias evolutivas, máquinas de estado finito na programação evolutiva e árvores na programação genética.

A Figura 3.1 apresenta os principais algoritmos evolucionários [8] e o esquema geral de um algoritmo evolucionário é mostrado na Figura 3.2 [16].

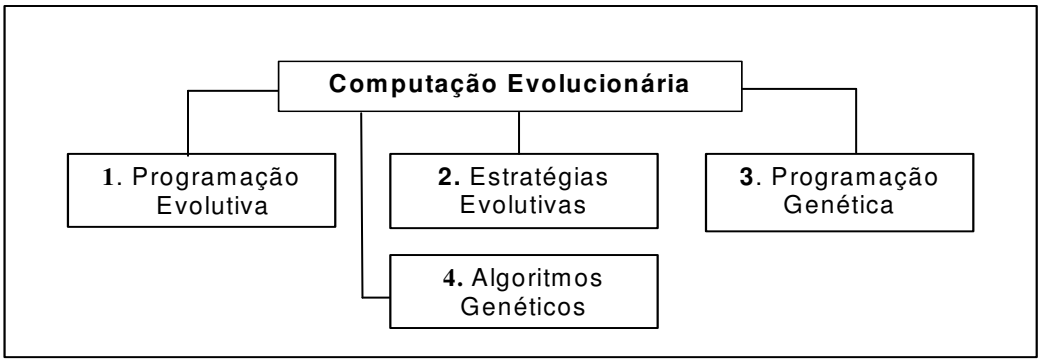


Figura 3.1 – Principais algoritmos evolucionários.

```
Início
 < 1 > Inicializar População (aleatoriamente)
 < 2 > Avaliar População (cada candidato)
 < 3 > Enquanto critério de parada não for satisfeito, Repetir
       < 3.1 > Selecionar pais (melhores indivíduos)
       < 3.2 > Recombinação (pares de pais)
       < 3.3 > Mutação (resultado da recombinação)
       < 3.4 > Avaliar novos candidatos
       < 3.5 > Selecionar indivíduos para a próxima geração
        Fim do Enquanto
Fim
```

Figura 3.2 – Representação geral de um algoritmo evolucionário.

3.3.5.1. Programação Evolutiva

A Programação Evolutiva (PE) foi inicialmente voltada para a evolução de máquinas de estado finito, sendo posteriormente estendida para problemas de otimização de parâmetros [15].

A PE trabalha com populações de indivíduos que sofrem diferentes níveis de mutação ao longo do processo, normalmente reduzindo-se à medida que a solução se aproxima do ponto “ótimo”.

3.3.5.2. Estratégias Evolutivas

As Estratégias Evolutivas (EE) foram concebidas para tratarem de problemas técnicos de otimização, sendo quase exclusivamente empregadas na Engenharia Civil como alternativa aos métodos convencionais. Operam com cromossomos na forma de vetores de números reais na proporção (1+1), isto é, cada progenitor gera um herdeiro por geração. Caso este descendente seja melhor que seu progenitor, ele toma seu lugar. Atualmente as EE foram estendidas para as proporções (m+1) e (m+n), além de terem tido estratégias de recombinação introduzidas no seu processo evolutivo [8].

3.3.5.3. Programação Genética

Usualmente desenvolvida utilizando-se a linguagem Lisp, devido à facilidade requerida na representação [15], a Programação Genética (PG) opera sobre representações de trechos de programas na forma de árvores, de modo que possam ser combinados para gerarem novos trechos de programas mais complexos.

3.3.5.4. Algoritmos Genéticos

Os Algoritmos Genéticos (AG) foram desenvolvidos por John Holland no final da década de 60 buscando inspiração no que se conhece sobre o processo de evolução natural, conhecimento este iniciado solidamente com a teoria da evolução de Darwin no seu famoso trabalho *The Origin of Species* [17].

Os interesses de John Holland, entretanto, não estavam limitados a problemas de otimização: ele estava interessado no estudo de sistemas adaptativos complexos, fossem eles biológicos (como o nosso sistema imunológico) ou não (como a economia global ou setorial).

Os AG constituem uma classe de ferramentas versátil e robusta e que pode ser utilizada na solução de problemas de otimização, embora não devam ser considerados estritamente minimizadores de funções. Quando usado como algoritmo de minimização, um AG se distingue das técnicas mais comuns de programação matemática por [17]:

- empregar uma população de indivíduos (ou soluções);
- trabalhar sobre uma codificação das possíveis soluções (genótipos) e não sobre as soluções (fenótipos) propriamente ditas;
- empregar regras de transição probabilísticas;
- não requerer informações adicionais (derivadas, por exemplo) sobre a função a otimizar e as restrições.

Assim, a busca de soluções pode se dar em conjuntos não-convexos e/ou disjuntos, com funções objetivo também não-convexas e não-diferenciáveis, podendo-se trabalhar simultaneamente com variáveis reais, lógicas e/ou inteiras. Vale ressaltar que os algoritmos genéticos não são facilmente presos a mínimos locais, como os algoritmos clássicos de programação matemática. Em função dessas características os AG, quando utilizados em projetos, podem levar à descoberta de soluções não convencionais e inovadoras, dificilmente vislumbradas por projetistas mais conservadores [8].

Os princípios da natureza nos quais os AG se inspiram são simples. De acordo com a teoria de Charles Darwin, o princípio de seleção privilegia os indivíduos mais aptos garantindo-lhes maior longevidade e maior probabilidade de reprodução. Indivíduos com mais descendentes têm mais chances de perpetuarem seus códigos genéticos nas gerações seguintes. Tais códigos constituem a identidade de cada indivíduo e estão representados nos cromossomos.

Esses princípios são incorporados na construção de algoritmos computacionais que buscam a melhor solução para um determinado problema por meio da evolução de populações de soluções, codificadas por cromossomos. Nos AG, um cromossomo é uma estrutura de dados que representa uma das possíveis soluções no espaço de busca do problema. Os cromossomos são submetidos a um processo evolucionário que envolve avaliação, seleção, recombinação (*crossover*) e mutação. Após vários ciclos de evolução a população deverá então conter indivíduos mais aptos.

A Tabela 3.1 apresenta alguns termos básicos relacionados com os AG [8]:

Tabela 3.1 – Termos básicos relacionados com os AG.

Termo	Definição
Cromossomo	Cadeia de caracteres representando alguma informação relativa às variáveis do problema.
Gene	Unidade básica do cromossomo. Cada cromossomo tem um certo número de genes, descrevendo uma certa variável do problema.
População	Conjunto de cromossomos.
Geração	Número da iteração que o AG executa.
Operações Genéticas	Operações às quais os cromossomos são submetidos.
Região Viável	Espaço que compreende as soluções possíveis ou viáveis do problema. É caracterizada pelas funções de restrição que definem as soluções.
Função Objetivo	Função a ser minimizada. Contém a informação numérica do desempenho de cada cromossomo na população.
Genótipo	Representa a informação contida nos cromossomos.

Comparação dos AG com os Métodos Clássicos

Em geral, a solução de problemas de otimização utilizando-se métodos clássicos é iniciada com um único candidato, chamado de *solução básica*. A direção para a qual se deve caminhar na busca do próximo candidato é feita, em geral, por meio do cálculo de derivadas. Os algoritmos clássicos que trabalham com o cálculo de derivadas são denominados *algoritmos de ordem n*, sendo *n* o grau da maior derivada utilizada. Por exemplo, o Método dos Gradientes Conjugados e o de Newton pertencem aos grupos de algoritmos de primeira e segunda ordem, respectivamente.

Os AG, por não utilizarem o cálculo de derivadas, são considerados algoritmos diretos ou de ordem zero, necessitando apenas da avaliação da função objetivo e introduzindo no processo de otimização dados e parâmetros randômicos. A solução do problema se dá de forma probabilística e orientada [9].

A Tabela 3.2 mostra uma comparação entre os AG e os métodos clássicos de programação matemática [1].

Tabela 3.2 – Comparação dos AG com os métodos clássicos.

Métodos Clássicos	**Algoritmos Genéticos**
Têm dificuldade em identificar soluções ótimas globais, uma vez que dependem do ponto de partida.	Não apresentam nenhuma restrição quanto ao ponto de partida.
Têm dificuldade em tratar problemas com variáveis discretas (problemas comuns em Engenharia).	Trabalham tanto com codificação contínua como discreta das variáveis, ou ainda com uma combinação de ambas.
Requerem funções diferenciáveis, o que pode ser oneroso, complexo e nem sempre possível.	Não necessitam que a função objetivo seja contínua nem diferenciável.
Cada um dos métodos clássicos, de uma forma geral, tem domínio de aplicação restrito.	São razoavelmente eficientes para a maioria dos problemas existentes.
Em geral, não são eficazes quando o problema tem múltiplos objetivos.	São flexíveis para trabalhar com restrições e otimizar múltiplas funções com objetivos conflitantes.

Trabalham com uma única solução em cada etapa do processo iterativo.	Realizam buscas simultâneas em várias regiões do espaço de busca por meio de uma população de indivíduos.
Não são tão fáceis de serem implementados, quando comparados com os AG.	São relativamente fáceis de serem implementados e proporcionam grande flexibilidade na modificação da função objetivo.

Estrutura dos Algoritmos Genéticos

Os AG podem ser caracterizados por meio dos seguintes componentes [16]:

- Problema a ser otimizado
- Representação das soluções do problema
- Decodificação do cromossomo
- Seleção
- Operadores genéticos
- Inicialização da população

São três as estruturas de AG mais encontradas na literatura técnica. A primeira é o AG genérico, ilustrado na Figura 3.3 [15].

```
Início
 < 1 > Inicialização da população P de cromossomos (geração i = 1)
 < 2 > Avaliação de indivíduos na população (função objetivo)
 < 3 > Repita (evolução)
       < 3.1 > Seleção de indivíduos para reprodução
       < 3.2 > Aplicação de operadores genéticos (crossover e/ou mutação)
       < 3.3 > Avaliação dos novos indivíduos criados na população
       < 3.4 > Seleção de indivíduos para sobrevivência (geração i+1)
       Até Satisfazer o critério de parada
Fim
```

Figura 3.3 – Representação em pseudo-código de um AG genérico.

Em função da maneira como são inseridos os novos indivíduos na população são conhecidos dois tipos extremos de algoritmos genéticos: os algoritmos do tipo *geracional* e os do tipo *regime permanente*.

No AG do tipo geracional toda a população é substituída por novos indivíduos gerados pelo processo de seleção e aplicação dos operadores genéticos [17], conforme ilustrado na Figura 3.4.

```
Início
< 1 > Inicialização da população P de cromossomos
< 2 > Avaliação de indivíduos na população P
< 3 > Repita
   < 3.1 > Repita
           < 3.1.1 > Seleção de 2 indivíduos em P para reprodução
           < 3.1.2 > Aplicação do operador crossover com probabilidade Pc
           < 3.1.3 > Aplicação do operador mutação com probabilidade Pm
           < 3.1.4 > Inserção do novo indivíduo em P'
           Até Completar população P'
   < 3.2 > Avaliação de indivíduos de P'
   < 3.3 > P ←—— P'
   Até Satisfazer o critério de parada
Fim
```

Figura 3.4 – Representação em pseudo-código de um AG geracional.

Neste tipo de algoritmo é comum a perda de bons indivíduos, uma vez que a geração de "pais" é totalmente substituída por outra mais nova de "filhos". Por esta razão, especialmente em problemas de otimização, adota-se o procedimento do elitismo, no qual os melhores indivíduos de uma geração são preservados, passando-se uma cópia para a geração seguinte.

No AG em regime permanente (*steady-state*) (Figura 3.5) apenas um (ou dois) indivíduo é criado de cada vez e, depois de sua avaliação, ele é inserido na população em substituição a algum outro elemento, por exemplo, o pior de todos os existentes. Caso o novo indivíduo seja inferior a todos os existentes, então nada é alterado e procede-se uma nova criação de indivíduos [17,18].

Com o intuito de facilitar a comparação do indivíduo gerado com os indivíduos já existentes na população, utiliza-se uma ordenação dentro da população. Desta forma, o indivíduo gerado é comparado apenas com o último indivíduo da ordenação e, caso seja superior a ele, assumirá sua posição correspondente na ordenação, sendo o último eliminado pela seleção natural.

```
Início
< 1 > Inicialização da população P de cromossomos
< 2 > Avaliação de indivíduos na população P
< 3 > Repita
      < 3.1 > Seleção do operador genético
      < 3.2 > Seleção de indivíduo(s) para reprodução
      < 3.3 > Aplicação de operador genético selecionado
      < 3.4 > Avaliação de indivíduo(s) gerado(s)
      < 3.5 > Seleção de indivíduo f para sobreviver
      < 3.6 > Se f é melhor que o pior elemento de P então
              < 3.6.1 > Inserir f em P de acordo com a sua ordem
      Até Satisfazer o critério de parada
Fim
```

Figura 3.5 – Representação em pseudo-código de um AG em regime permanente.

É importante salientar que existem ainda versões de algoritmos genéticos nas quais uma parte da população é substituída, permitindo a coexistência de pais e filhos.

3.3.5.4.1. Operadores Genéticos

O princípio básico dos operadores genéticos consiste em transformar a população por meio de sucessivas gerações, para que se possa obter um resultado satisfatório no final do processo. Deste modo, os operadores genéticos são extremamente necessários para que a população se diversifique e mantenha as características de adaptação adquiridas pelas gerações anteriores [8].

- ***Crossover* (Recombinação Genética)**

Tem por objetivo propagar as características dos indivíduos mais aptos, trocando material genético [7]. Existem vários tipos: *crossover* de ponto único, de dois pontos e uniforme, dentre outros.

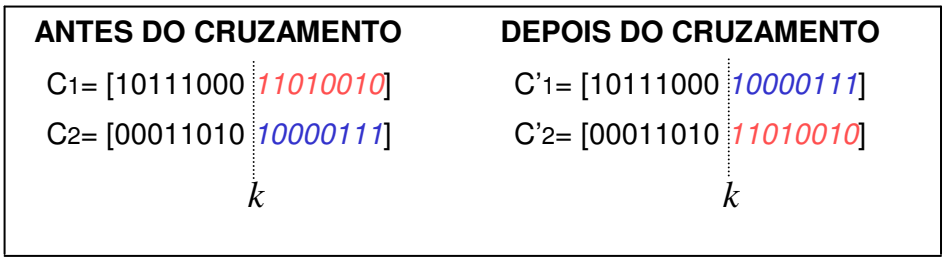


Figura 3.6 – Representação do operador *crossover* [2].

Seja "k" (Figura 3.6) o ponto escolhido aleatoriamente como ponto de cruzamento na cadeia de bits de cada cromossomo, a quantidade de cromossomos ou elementos a serem cruzados é definida pela probabilidade de cruzamento, especificada previamente pelo usuário.

O processo de escolha dos indivíduos que serão cruzados é feito em pares (C_1 e C_2), sorteando-se números randômicos. Desta forma, cada cadeia é partida no ponto "k" e todas as informações do cromossomo C_1 são copiadas para o cromossomo C_2 a partir do ponto escolhido, e vice-versa.

O ponto de cruzamento "k " é calculado a partir da seguinte equação [7]:

$$k = 1 + [(L-1) - 1] \,.\, r \qquad (3.2)$$

onde r é um número randômico, gerado entre 0 e 1, e L é o número total de elementos que formam a cadeia de caracteres (comprimento).

Depois de concluída a aplicação deste operador, a população é submetida a um novo operador genético, a mutação.

- **Mutação**

Apresentando baixa probabilidade, a mutação é responsável pela diversidade do material genético. Trata-se de uma modificação aleatória no valor de um alelo da cadeia. Caso o alelo escolhido seja igual a zero (0) ele passa a valer um (1) e vice-versa.

Na literatura, uma técnica muito usada para se fazer a mutação é criar pares (a,b) randômicos, em que "a" representa a linha e "b" a coluna da matriz de cromossomos, de forma que o bit a ser mudado será o correspondente ao elemento (a,b). Neste processo, exclui-se o melhor indivíduo para garantir que não haverá perda de material genético valioso.

Outro mecanismo consiste em selecionar randomicamente uma posição em um cromossomo obedecendo a uma probabilidade de mutação especificada pelo usuário, mudando-se assim o valor do alelo. Como probabilidade de mutação recomenda-se um valor (em percentagem) no intervalo [0.1 ; 1] de acordo com Saramago [7].

A Figura 3.7 mostra como acontece o processo de mutação.

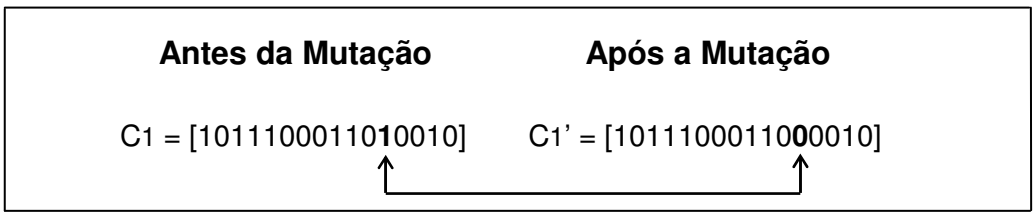


Figura 3.7 – Exemplo da aplicação do operador mutação

3.3.5.4.2. Parâmetros de Controle dos AG

A configuração dos parâmetros de controle dos AG afeta diretamente seu desempenho. A eficiência de um AG é altamente dependente dos seguintes parâmetros [8]:

- **Tamanho da População (N)**

Indica o número de indivíduos ou cromossomos presentes na população; este número é constante durante todo o processo de otimização.

Quanto maior a população maior será a diversidade de soluções, mas o número de avaliações das funções de aptidão para cada geração também será maior. Portanto, a influência deste parâmetro está relacionada com o desempenho global e a eficiência computacional dos AG.

Muitos pesquisadores sugerem tamanhos de população entre 10 e 100 cromossomos [8].

- **Probabilidade de Cruzamento**

Dependendo do valor de probabilidade de cruzamento (Pc), será realizada a inclusão de novos indivíduos na população. Portanto, se este valor for muito alto, indivíduos com boas aptidões poderão ser substituídos.

Geralmente a taxa de cruzamento varia entre 0.5 e 0.95, mas muitas vezes esse valor é limitado dependendo do tipo de algoritmo.

- **Probabilidade de Mutação**

A probabilidade de mutação (Pm) deve ser cuidadosamente definida. Se ela é baixa, geralmente os algoritmos ficam presos em mínimos locais. Por outro lado, para valores elevados de Pm, a propagação de um bom esquema de herança será feita impropriamente e os algoritmos serão corrompidos, caindo em um método de busca aleatório [18].

De Jong [19] sugere que a taxa de mutação seja inversamente proporcional ao tamanho da população.

Hesser e Manner [20] sugerem que a taxa ótima de mutação seja calculada pela expressão:

$$Pm = \frac{1}{N\sqrt{L}} \tag{3.3}$$

onde N é o tamanho da população e L o comprimento dos cromossomos.

Recomendações de Pesquisadores

De Jong [20] sugere para um bom desempenho dos AG a seguinte configuração de parâmetros:

$$\begin{cases} N = 50 \\ Pc = 0.6 \\ Pm = 0.001 \end{cases}$$

Restrições e Funções de Penalização

A maioria dos problemas de otimização apresenta restrições [20]. A solução obtida como resultado final da procura evolucionária de um algoritmo genético deve ser necessariamente viável, isto é, deve satisfazer a todas as restrições.

Alguns aspectos devem ser considerados na idealização das restrições. O primeiro diz respeito ao número de restrições. Problemas com múltiplos objetivos podem ser reformulados de tal maneira que alguns dos objetivos atuem como restrições. A dificuldade de se satisfazer as restrições aumenta com uma taxa de crescimento muito maior do que a do crescimento do próprio número de restrições.

O segundo está associado à sua dimensão, i.e, se contínuo ou discreto, de forma que a violação da restrição possa ser avaliada para se determinar se satisfaz ou não a essa dimensão.

O terceiro está relacionado com a análise crítica que se deve fazer quando ocorrem pequenas violações das restrições. Isto permite que pequenas violações, tecnicamente viáveis, sejam consideradas na solução final.

Finalmente, deve-se levar em consideração a dificuldade de se satisfazer às restrições, a qual pode ser caracterizada pelo tamanho do espaço das soluções comparado ao tamanho total da amostra. Inicialmente, essa dificuldade não é conhecida, mas pode ser definida de duas formas: a primeira está ligada à facilidade de se mudar uma solução que violou a restrição e a segunda diz respeito à probabilidade de se violar a restrição durante a pesquisa.

Os métodos evolucionários produzem soluções novas pela recombinação (*crossover*) e/ou perturbação (mutação) das soluções existentes. Para se preservar indivíduos que satisfaçam às restrições podem ser implementados os operadores genéticos ou operadores de reparo. Infelizmente, para a maioria dos problemas de engenharia não é fácil fazer uma solução não-viável se tornar viável. Na maioria dos algoritmos genéticos, soluções que violam as restrições são descartadas imediatamente e não consideradas nas soluções seguintes.

As restrições podem ser consideradas nos algoritmos genéticos por meio das funções de penalização. Um método simples para se penalizar soluções não-viáveis é aplicar uma penalidade constante. Assim, a função objetivo penalizada assume o valor da função objetivo não penalizada adicionada da penalidade. A função de penalização para o problema de minimização, com *m* restrições, pode ser escrita da seguinte forma [21].

$$F_p = F + \sum C_i \delta_i \tag{3.4}$$

onde:

F_p é a função objetivo penalizada,

F é a função objetivo original,

δ_i = 1, se a restrição é violada,

δ_i = 0, se a restrição é satisfeita,

C_i é o coeficiente de penalização imposto para a violação de i-ésima restrição.

3.3.5.4.3. Revisão Bibliográfica

Nesta seção são apresentados alguns trabalhos que utilizam algoritmos genéticos na otimização do dimensionamento de estruturas em geral, tendo como objetivo principal a minimização de seus pesos. São incluídos também trabalhos sobre a melhoria da convergência dos algoritmos genéticos aplicados na minimização de funções contínuas. São discutidos aspectos como a natureza do problema, os tipos de variáveis e as técnicas empregadas para se evitar possíveis problemas de estagnação na busca da solução “ótima”.

Erbatur e Hasançebi [22] desenvolveram uma ferramenta computacional, denominada GAOS (*Genetic Algorithm Based Optimum Optimization*) para a otimização discreta de estruturas planas e espaciais compostas por elementos unidimensionais, selecionando seções de elementos estruturais a partir de uma lista de perfis disponíveis. A característica principal da metodologia empregada é o uso de um algoritmo genético como otimizador. De acordo com os autores, um AG freqüentemente encontra a região do espaço de busca contendo a solução “ótima” global e não o verdadeiro ponto de mínimo global. Diante desta dificuldade os autores propõem um método de otimização de múltiplos níveis, o qual, segundo eles, elimina o problema de estagnação em mínimos locais. O procedimento consiste em se reduzir o espaço de busca em cada nível sucessivo de otimização de tal forma que, nos níveis subseqüentes, as variáveis sejam movidas para dentro de um espaço de busca mais restrito.

Dentre as principais características da ferramenta computacional desenvolvida, destacam-se [22]:

1. A violação de restrições, considerada por meio de funções de penalização.
2. A incorporação de um programa de análise estrutural, compatível com o AG, para se obter um menor tempo computacional, já que, segundo os autores, o uso de um programa externo é ineficiente.
3. A preparação de um arquivo com dados de entrada: dados da análise estrutural e os dados requeridos pelo AG (inicialização de parâmetros).
4. Para estruturas de aço o programa seleciona seções prontas de uma lista de perfis especificada nos dados de entrada.

Davis [23] propõe a adaptação do AG a cada aplicação particular ao invés de se desenvolver um programa robusto de propósito geral que funcione bem para diversos problemas. A proposta de Davis consiste em adaptar o AG ao problema em estudo, incorporando o máximo de conhecimentos aos AG e fazendo a hibridização dos AG com outros métodos de otimização. São propostos três princípios de hibridização:

(1) *Usar a atual codificação*: representar as soluções candidatas da mesma maneira como são representadas no método atual, usando uma codificação real ou uma representação binária.

(2) *Fazer a hibridização quando possível*: combinar, quando possível, as características úteis do algoritmo atual com o AG.

(3) *Adaptar os operadores genéticos ao problema*: inventar novas formas de mutação e *crossover* que sejam apropriadas para o problema e a codificação.

A expectativa, segundo Davis, é que com estes princípios o algoritmo híbrido opere melhor do que o algoritmo atual ou o AG isolado.

Shyue-Jian Wu e Chow [18] também realizaram pesquisas sobre a otimização discreta de estruturas. Em seu trabalho, os autores discutem a preocupação com as limitações apresentadas pelos diferentes métodos de otimização, tais como baixa eficiência, pouca confiabilidade e a tendência a ficarem presos a mínimos locais. Eles apresentaram o método *Steady-State Genetic Algorithm* (*Steady-State GA*), com um baixo percentual de população a ser substituído durante cada geração. A representação adotada é de vetores de caracteres binários de comprimento fixo, que são construídos sobre o alfabeto binário {0,1}.

Cada variável de projeto é representada por um vetor de tamanho λ. O valor de λ depende do número total de variáveis discretas [18], i.e.

$$2^{\lambda} = \text{número de variáveis discretas} \tag{3.5}$$

Por exemplo, se existem 16 variáveis discretas, então o valor de λ é 4. O comprimento total L do cromossomo é dado por:

$$L \approx \sum_{i=1}^{n} \lambda_i \tag{3.6}$$

onde n é o numero total de variáveis.

Para se fazer a decodificação, são adotados os seguintes passos: (1) o vetor de caracteres binário é decodificado em um número inteiro decimal; (2) o valor físico da variável de projeto é determinado por correspondência de 0 até o número de variáveis discretas.

André *et al.* [24] pesquisaram sobre como melhorar os algoritmos genéticos para evitar a convergência prematura em problemas de otimização contínua. Eles descrevem os conflitos usuais que se apresentam no processo de otimização entre a confiabilidade e o tempo de execução do problema, resultando muitas vezes em soluções insatisfatórias, caracterizadas por uma convergência lenta quando requerida uma solução exata.

Os autores apresentam um algoritmo genético com codificação binária destinado à otimização de problemas reais, complexos e contínuos. A eficiência do algoritmo proposto é testada utilizando-se 20 funções analíticas das quais são conhecidos tanto os mínimos globais quanto os locais. Tais funções serão utilizadas para testar o algoritmo desenvolvido e apresentado no presente trabalho.

As principais melhorias apresentadas no trabalho de André *et al*. são as inclusões do fator de escala (SF, *Scale Factor*) e do intervalo adaptável de estudo (*adaptive study interval*), os quais influenciaram dois dos mais importantes critérios de avaliação: velocidade e convergência. Na inclusão do fator de escala eles explicam que a convergência prematura se dá devido ao fato de que após vários ciclos de execução do programa a população tende a ser homogênea e a ação do operador *crossover* não se faz mais sensível. Portanto, o valor encontrado não é o mínimo global. Para evitar esse fenômeno, o fator de escala é introduzido no cálculo da probabilidade do *crossover:* nas primeiras iterações os melhores indivíduos obtêm uma menor probabilidade de *crossover* da que eles deveriam ter, e os piores uma maior. “A população resultante é mantida mais heterogênea no início do algoritmo, o que diminui o risco de se ficar preso em mínimos locais” [24]. Logo, nas últimas iterações, o fator de

escala é aumentado para garantir a convergência eficiente do algoritmo (SF = 1 na última iteração).

Del Sávio *et. al*. [25] desenvolveram um estudo sobre uma metodologia de otimização para estruturas de aço 2D no qual as principais variáveis estão relacionadas com a rigidez estrutural dos nós. O processo de otimização é feito por meio de um algoritmo evolucionário desenvolvido e implementado em um programa de análise estrutural FTOOL [25]. Nessa pesquisa, os autores estudam a capacidade a flexão de estruturas de aço, variando a rigidez e a resistência de suas juntas semi-rígidas para obter uma distribuição "ótima" de momentos fletores que lhes permita determinar um perfil de aço eficiente para o projeto.

Albrecht [1] desenvolveu uma ferramenta computacional para otimizar os sistemas de ancoragem em termos dos deslocamentos sofridos pelas unidades flutuantes utilizadas para a exploração de petróleo e das tensões nas linhas de ancoragem. No seu trabalho, o autor apresenta uma metodologia baseada nos princípios da computação evolucionária, com ênfase em três métodos: algoritmos genéticos tradicionais, Micro Algoritmo Genético (Micro AG) e Enxame de Partículas. O Micro AG é uma variação do AG tradicional conhecido e, segundo o autor, muito eficiente para evitar problemas de convergência precoce e de estagnação em mínimos locais. Este algoritmo utiliza uma população inicial reduzida, por exemplo, N = 4. Mas, com uma população tão pequena, como é de esperar, o método tenderá a convergir rapidamente para um mínimo local. Quando isso acontece o melhor dos indivíduos é guardado e mantido na população e os demais são substituídos por outros gerados aleatoriamente. O processo é repetido até que seja atingido o critério (ou critérios) de parada.

A otimização pelo método de Enxame de Partículas, também conhecida pela sua sigla em inglês PSO (*Particle Swarm Optimization*), foi desenvolvida a partir da modelagem matemática do comportamento social de bandos de pássaros [1]. Segundo o autor, ao levantar vôo os pássaros se comportam de forma aleatória. Mas, passando algum tempo já estarão voando organizadamente e, caso seja encontrado um bom lugar para se alimentar, todos os pássaros do bando se dirigirão ao local e pousarão. Baseados nesses princípios, Kennedy e Eberhardt [1], autores do método, propuseram um algoritmo muito simples e robusto, no qual cada pássaro é representado por um ponto ou partícula e o local de pouso representa um ponto de mínimo. Albrecht adaptou esse método de otimização com os algoritmos genéticos, resultando em

um algoritmo evolutivo baseado em uma população de indivíduos, porém a evolução da população não é feita com os operadores utilizados no AG tradicional (mutação e *crossover*) mas com o vôo de cada ponto no espaço de busca, e o conceito de gerações é substituído pelo conceito de intervalos de tempo. Neste contexto, o principal elemento que produz a evolução de cada indivíduo da população é a sua velocidade de vôo, definida como um vetor dentro do espaço de busca. Assim, a posição do ponto em cada intervalo de tempo dependerá da sua posição e velocidade anteriores. A posição no intervalo de tempo "i + 1" é calculada por meio da soma da posição com sua velocidade no instante "i".

Neste trabalho, Albrecht otimiza os modelos de ancoragem em relação aos raios de ancoragem, azimutes, comprimentos das linhas e pré-tensões de trabalho, e o comportamento do sistema de ancoragem é estudado por meio de análises estáticas.

Da Silva *et al*. [26] apresentaram uma ferramenta para a otimização de ligações estruturais em aço através de algoritmos genéticos. Na sua pesquisa, os autores desenvolveram um programa para o dimensionamento e detalhamento automático de ligações em estruturas de aço. Assim, o algoritmo genético tem como principal função a generalização do processo de dimensionamento do algoritmo padrão.

Alguns outros trabalhos utilizando algoritmos de computação evolucionária têm sido desenvolvidos no Departamento de Engenharia Civil da PUC-Rio. Dentre eles podem-se citar: Borges *et al*. [27] realizaram um estudo sobre a avaliação da rigidez pós-limite de ligações semi-rígidas viga-coluna; Ramires *et al*. [28] desenvolveram um método para a otimização estrutural de ligações viga-coluna; Fonseca *et al*. [29] apresentaram uma metodologia para o estudo do fenômeno de cargas concentradas utilizando os algoritmos genéticos.

3.3.5.4.4. Exemplo de Otimização de uma Função Utilizando AG

Com a finalidade de ilustrar a aplicação dos AG e seus operadores e apresentar a nomenclatura utilizada por eles, o seguinte problema de minimização de uma função é apresentado [7]:

$$\text{Min } g(x,y) = x\sin(4x) + 1.1y\sin(2y) \tag{3.7}$$

no intervalo $8 < x < 10$, $8 < y < 10$, que define a região viável do problema.

Para este exemplo, será adotada a precisão de duas casas decimais (m = 2). Como o espaço de busca tem amplitude I = 10 – 8 = 2 e precisão de duas casas decimais, este intervalo deve ser dividido em I x 10^m subintervalos iguais, neste caso, 2 x 10^2 = 200 pontos. Portanto, a seqüência binária (cada gene) deverá ter pelo menos 8 bits, ou seja:

$$\overbrace{}^{x}\overbrace{}^{y}$$

C1 = 10000101 00100111
C2 = 00001110 00001001
C3 = 10010001 00000001
C4 = 11000101 00101001
C5 = 01111100 10101100
C6 = 11100010 01001010

Desta forma os cromossomos são formados concatenando-se todos os genes para formar uma única seqüência binária.

Tem-se assim a população inicial de cromossomos definida, em que cada gene é um vetor binário de λ bits, sendo λ função da precisão exigida (10^{-2}) e da amplitude do intervalo definido pelas restrições laterais (I = 2). Portanto, o comprimento total do cromossomo L é calculado aplicando-se a Equação (3.6).

A seguir, todos os indivíduos são modificados, sendo submetidos aos operadores genéticos.

De acordo com a Equação (3.8), se o indivíduo for de baixa adequabilidade, a sua probabilidade de desaparecer da população é alta; caso contrário, terá grandes chances de permanecer na geração seguinte:

$$p_i = \frac{f_i(x)}{F(x)} \tag{3.8}$$

onde $F(x)$ é o somatório de todos os $f_i(x)$, e p_i é a probabilidade proporcional ou probabilidade de ocorrência.

Para se calcular o valor da função de adaptação f_i, deve-se converter a seqüência binária (base 2) para base 10, ou seja, deve-se decodificar o cromossomo conforme as Equações (3.9) e (3.10). Considerando-se um cromossomo $C = [b_7 b_6 ..... b_2 b_1 b_0 a_7 a_6 ..... a_1 a_0]$, tem-se:

$$x_{decimal} = \sum_{i=0}^{\lambda-1} b_i \cdot 2^i \tag{3.9}$$

$$y_{decimal} = \sum_{i=0}^{\lambda-1} a_i \cdot 2^i \tag{3.10}$$

Em seguida, deve-se calcular o valor real dentro da região viável.

$$x = x_{\min} + x_{\max} \cdot \frac{x_{\max} - x_{\min}}{2^{\lambda} - 1} \tag{3.11}$$

onde $x_{\min}$ *e* $x_{\max}$ são as restrições laterais inferior e superior, respectivamente, da variável x.

No problema em questão a função de adaptação $g(x,y)$ permite fazer a avaliação final que decide sobre a vida ou a morte de cada cromossomo. A avaliação da população inicial (Tabela 3.3) é feita aplicando-se as Equações (3.9) –(3.11). O mecanismo para a seleção das melhores cadeias, ou seja, as mais adaptadas, é definido pelo uso das probabilidades proporcionais (p_i) utilizando-se a Equação (3.8), obtendo-se os seguintes valores:

$$p_1 = \frac{-16.26}{-36.92} = 0.44 \qquad p_2 = \frac{+3.21}{-36.92} = -0.09 \qquad p_3 = \frac{-11.01}{-36.92} = 0.30$$

$$p_4 = \frac{-2.76}{-36.92} = 0.07 \qquad p_5 = \frac{-10.32}{-36.92} = 0.28 \qquad p_6 = \frac{+0.22}{-36.92} = -0.00$$

É importante observar que $p_1 + p_2 + p_3 + p_4 + p_5 + p_6 = 1.00$

Tabela 3.3 – Avaliação dos indivíduos da população inicial.

CROMOSSOMO	***x***	***y***	***g(x,y)***
1000010100100111	9.04	8.31	-16.26
0000111000001001	8.11	8.07	3.21
1001000100000001	9.14	8.01	-11.01
1100010100101001	9.55	8.32	-2.76
0111110010101100	8.98	9.35	-10.32
1110001001001010	9.77	8.58	0.22
∑g(x,y)	**- 36.92**		

Considerando-se as possibilidades cumulativas q_i de cada cromossomo, ou seja:

$$q_i = \sum p_j \tag{3.12}$$

são obtidos os seguintes valores cumulativos:

$q_1 = p_1 = 0.44$
$q_2 = p_1 + p_2 = 0.35$
$q_3 = 0.44 - 0.09 + 0.30 = 0.65$
$q_4 = 0.44 - 0.09 + 0.30 + 0.07 = 0.72$
$q_5 = 0.44 - 0.09 + 0.30 + 0.07 + 0.28 = 1.00$
$q_6 = 0.44 - 0.09 + 0.30 + 0.07 + 0.28 - 0.00 = 1.00$

A seguir devem ser selecionadas as cadeias que irão contribuir para a próxima geração. Esta seleção considera um conjunto de números *r*, escolhidos aleatoriamente entre (0,1) em quantidade igual ao número de cadeias. A análise é feita considerando-se as seguintes opções:

- Se $r < q_1$, então seleciona-se o cromossomo C1.
- Se $r > q_1$, então passa-se para o subseqüente e faz-se a análise novamente.

Vale ressaltar que alguns cromossomos poderão ser selecionados mais de uma vez, ou seja, os melhores serão copiados mais vezes, enquanto os demais morrerão.

Para o exemplo em estudo, considere-se que foram gerados os seguintes números randômicos:

r_1 = 0.64; r_2 = 0.08; r_3 = 0.47; r_4 = 0.88; r_5 = 0.93 e r_6 = 0.70

A seleção de cromossomos é dada por:

$r_1 > q_1$; $r_1 > q_2$; $r_1 < q_3$, selecionando-se $q_3 \rightarrow$ C3

$r_2 < q_1$, selecionando-se $q_1 \rightarrow$ C1

$r_3 > q_1$; $r_3 > q_2$; $r_3 < q_3$, selecionando-se $q_3 \rightarrow$ C3

De maneira análoga, selecionam-se os cromossomos C5, C5 e C4, e a nova população passa a ser representada por:

C'1= 1001000100000001 → gerado de C3; $g(x,y)$ = -11.01
C'2= 1000010100100111 → gerado de C1; $g(x,y)$ = -16.26
C'3= 1001000100000001 → gerado de C3; $g(x,y)$ = -11.01
C'4= 0111110010101100 → gerado de C5; $g(x,y)$ = -10.32
C'5= 0111110010101100 → gerado de C5; $g(x,y)$ = -10.32
C'6= 1100010100101001 → gerado de C4; $g(x,y)$ = - 2.76

Existem várias formas de se obter o cruzamento. Neste exemplo será utilizada a seguinte técnica: seja k o ponto que define a posição de cruzamento na cadeia de bits de cada cromossomo escolhido aleatoriamente (C_1 e C_2); a quantidade de cromossomos a ser submetida ao processo de cruzamento é definida através da probabilidade de cruzamento Pc, especificada pelo usuário. A probabilidade de cruzamento adotada neste exemplo foi de Pc = 25%. Cada cadeia é partida nesse ponto k e todas as informações do cromossomo C_1, a partir do ponto escolhido, são copiadas para o cromossomo C_2 e vice-versa, conforme ilustrado na Figura 3.6.

O processo de escolha do cromossomo que será cruzado deve ser feito em pares, sorteando-se números randômicos (r_i). Quando não for possível formar os pares, um novo sorteio será feito até que se obtenham os pares necessários para o cruzamento. Por exemplo, se r_1 for menor que a probabilidade Pc, então o cromossomo C'_1 será selecionado.

O próximo passo consiste em gerar um novo número randômico para determinar a posição k, com a formação de duas novas cadeias devido à troca de todos os caracteres compreendidos entre as posições k +1 e L (comprimento do cromossomo). A posição k é determinada pela Equação (3.2).

Considerando-se que, para este exemplo, foram gerados os seguintes números randômicos:

r_1 = 0.64 > Pc; r_2 = 0.08 < Pc; r_3 = 0.47 > Pc; r_4 = 0.88 < Pc; r_5 = 0.93 < Pc e r_6 = 0.70 > Pc,

então, devem ser selecionados os cromossomos C'_2, C'_4, C'_5 e C'_6, e a nova população passa a ser representada por:

C''$_1$-1001000100000001; $g(x,y)$ = -11.01
C''$_2$-1000010100101100; $g(x,y)$ = -16.72
C''$_3$-1001000100000001; $g(x,y)$ = -11.01
C''$_4$-0111110010100111; $g(x,y)$ = -11.02
C''$_5$-0111110010101001; $g(x,y)$ = -10.67
C''$_6$-1100010100101100; $g(x,y)$ = - 3.10

Em seguida, aplica-se o operador mutação aos novos indivíduos. Uma técnica para se fazer a mutação consiste em gerar pares (a,b) randômicos

onde a representa a linha e b a coluna da mudança do bit. Nesta forma de se aplicar o operador mutação exclui-se da seleção o melhor cromossomo. No exemplo em estudo, considerando-se os pares gerados como sendo (1,10) e (5,3), o cromossomo C"2 não será objeto da mutação por ser o melhor, ou seja, por apresentar o menor valor para a função objetivo. A probabilidade de mutação Pm adotada para este exemplo foi de 1%. Aplicando-se o operador mutação, conforme ilustrado na Figura 3.7, os cromossomos C"1 e C"5 passam a ser:

C"1 posição 10	1001000100000001	1001000101000001
C"5 posição 3	0111110010101001	0101110010101001

Portanto, após a aplicação da mutação, a população se transforma em:

C"'1-1001000101000001; $g(x,y)$ = -11.52
C"2 -1000010100101100; $g(x,y)$ = -16.72
C"3 -1001000100000001; $g(x,y)$ = -10.68
C"4 -0111110010100111; $g(x,y)$ = -11.06
C"'5-0101110010101001; $g(x,y)$ = -12.28
C"6 -1100010100101100; $g(x,y)$ = - 2.09
$\sum g(x,y)$ = -58.52

Concluída a aplicação dos três operadores, considera-se encerrado o primeiro ciclo da primeira geração. Observando-se os últimos dados, conclui-se que a população melhorou no sentido de caminhar na direção da minimização da função objetivo (note-se que o valor de $\sum g(x,y)$ passou de -36.92 a -58,52). Nesta primeira iteração, o menor valor obtido foi $g(x,y)$ = -16.72, correspondente aos pontos x = 9.04 y = 8.31.

4
Implementação Computacional

4.1.
Introdução

Neste capítulo é apresentada a formulação matemática do problema de otimização da disposição das linhas de ancoragem para minimizar os deslocamentos (*offsets)* das unidades flutuantes quando submetidas às condições ambientais. São apresentados os principais aspectos do algoritmo s*teady-state,* implementado neste trabalho, assim como a sua aplicação a diversos problemas de otimização de treliças e outras funções contínuas encontradas na literatura técnica. As configurações usadas em cada problema são detalhadas e comparações com as soluções conhecidas são discutidas.

4.2.
Formulação Matemática

O problema proposto neste trabalho consiste na minimização dos deslocamentos sofridos pelas unidades flutuantes quando submetidas a diferentes combinações de carregamentos provenientes das condições ambientais.

Uma das principais vantagens dos AG, quando comparados com os métodos clássicos, é permitir a manipulação de variáveis de projeto de forma contínua, discreta ou através de uma combinação de ambas. Neste trabalho as variáveis envolvidas no problema foram tratadas de forma contínua e, portanto, a distribuição "ótima" das linhas de ancoragem pode ser expressa como um problema contínuo de otimização sem restrições da seguinte maneira:

$$\text{Min } \sum_{i=1}^{m} \Delta_i^2(\alpha) = \sum_{i=1}^{m} [\Delta x_i^2(\alpha) + \Delta y_i^2(\alpha)] \tag{4.1}$$

sujeito a:

$$\alpha_{j\,\min} \le \alpha_j \le \alpha_{j\,\max}, \quad j = 1...n \tag{4.2}$$

onde Δ é o deslocamento resultante da unidade flutuante, que pode ser decomposto nas suas componentes Δx e Δy, para um dado conjunto de

condições ambientais; $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ é um vetor que contém todas as variáveis de projeto, i. e., os azimutes das linhas de ancoragem; n é o número de variáveis independentes de projeto; m é o número de combinações das condições ambientais; e, finalmente, as desigualdades mostradas na Equação (4.2) são as restrições laterais.

A Figura 4.1 ilustra uma unidade flutuante ancorada por 8 linhas e suas respectivas disposições (α_i) a serem consideradas no problema de otimização.

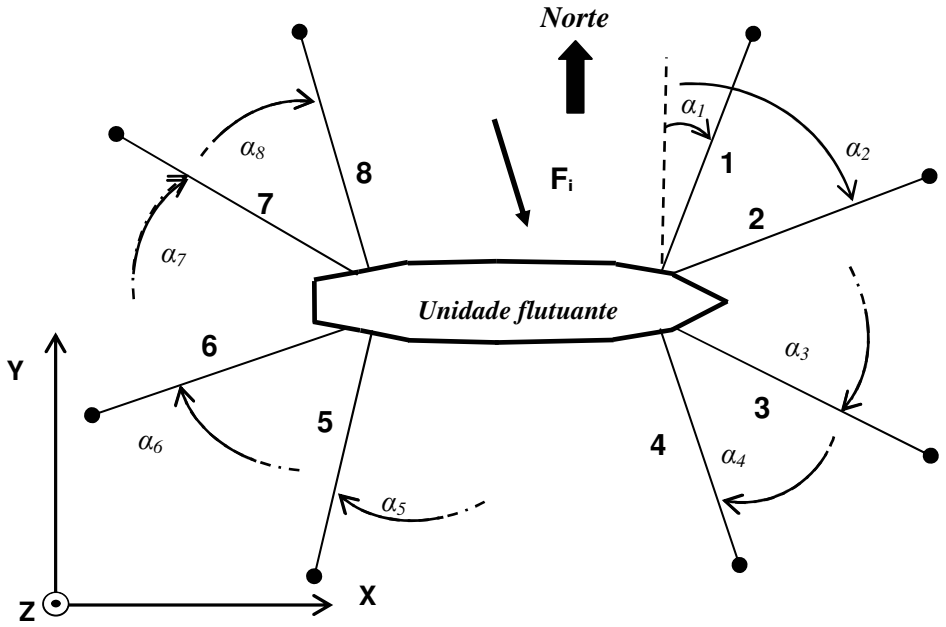


Figura 4.1 – Representação de um sistema de ancoragem com 8 linhas.

A minimização dos deslocamentos (também conhecidos como *offsets* estáticos) implica em se determinar uma disposição "ótima" das linhas de ancoragem (Figura 4.2).

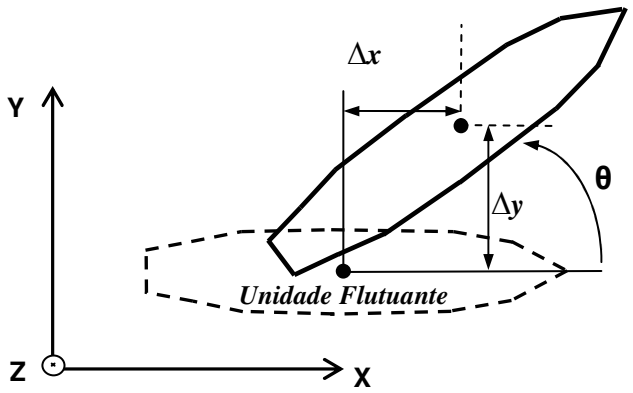


Figura 4.2 – Deslocamentos sofridos por uma unidade flutuante sob a ação de cargas externas.

4.3. Codificação das Variáveis de Projeto

Existem várias formas de se representar as variáveis de projeto, tais como vetores binários, vetores de números reais e listas de permutação, dentre outras. Neste trabalho as variáveis de projeto foram codificadas utilizando-se vetores de números binários de comprimento fixo, construídos com o alfabeto binário {0,1} e concatenados de ponta a ponta para formar um único vetor de comprimento maior. Essa estrutura concatenada representa o cromossomo. Assim, cada cromossomo contém todas as variáveis de projeto.

O comprimento total de cada cromossomo é calculado aplicando-se a Equação (3.6).

4.4. Cálculo dos Deslocamentos

Um conjunto de condições ambientais compreende as ações de ventos, ondas e correntes marinhas. Em geral, para análises estáticas, como é o caso do presente trabalho, as condições ambientais são aproximadas por valores médios, constantes ao longo do tempo, que atuam sobre a unidade flutuante. Esta força externa equivalente (Fi) caracteriza o tipo de análise conhecida como "quase-estática". No cálculo dos deslocamentos das unidades flutuantes, as linhas de ancoragem são tratadas como molas não-lineares (Figura 4.3) que impõem forças de restauração na unidade. Tais forças, expressas como uma função da distância horizontal entre a âncora e o ponto de conexão no extremo superior da linha, são descritas por meio de curvas de restauração, as quais são geradas utilizando-se a equação da catenária.

Obtidas as forças desequilibradas, uma nova posição de equilíbrio estático da unidade flutuante é calculada resolvendo-se o seguinte sistema de equações não-lineares:

$$\boldsymbol{K}\,\boldsymbol{d} = \boldsymbol{F}_i - \boldsymbol{R}_{int} \tag{4.3}$$

onde $\boldsymbol{K}$ é a matriz de rigidez; $\boldsymbol{d}$ é o vetor de deslocamentos que se deseja obter; $\boldsymbol{F}_i$ corresponde às forças externas devido a cada conjunto de condições ambientais; e $\boldsymbol{R}_{int}$ é a resultante das forças internas (de restauração) considerando-se todas as linhas de ancoragem.

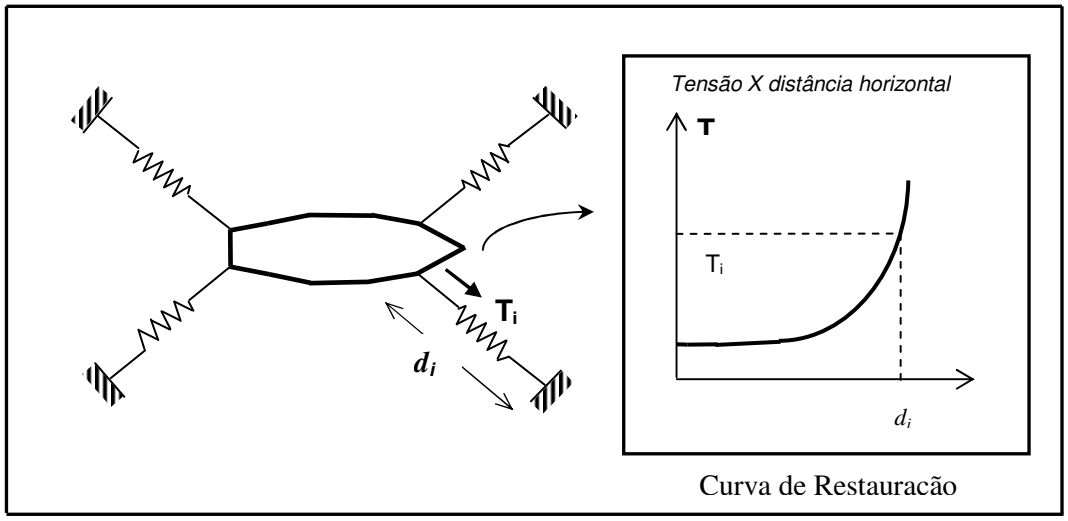


Figura 4.3 – Representação de linhas de ancoragem por meio de molas não–lineares.

A solução do sistema de Equações (4.3) conduz a uma nova posição da unidade flutuante. Esse cálculo é repetido até que a resultante dos deslocamentos obtidos seja menor do que uma dada tolerância (critério de convergência), isto é, até que a unidade alcance a sua posição final de equilíbrio estático.

4.5. Função *Fitness*

A função *fitness* ou de aptidão é um valor de qualidade que mede a eficiência da reprodução dos indivíduos em uma população, de acordo com o princípio de sobrevivência dos mais aptos [18]. Isto significa que um cromossomo com um valor mais elevado de *fitness* terá maiores probabilidades de ser selecionado como pai no processo de reprodução. Conseqüentemente, o problema de minimização da função objetivo pode ser transformado em um problema de maximização da função *fitness*, a qual é definida pelas seguintes expressões:

$$F_i = 1 - \frac{\phi_i}{\phi_{\max}} \tag{4.4}$$

$$\phi_i = \frac{\Delta_i^2}{\Delta_{avg}^2} \tag{4.5}$$

onde Δ_i^2 é a função objetivo (Equação (4.1)) e Δ_{avg}^2 é a média dos valores da função objetivo. De acordo com a Equação (4.4), a função *fitness* é normalizada para que os valores negativos possam ser excluídos.

4.6. Operador *Crossover*

Neste trabalho foi adotado o operador *crossover* simples ou *crossover* de um ponto (1X). Neste tipo de operador, apenas um ponto de cruzamento é escolhido aleatoriamente; assim os dois cromossomos selecionados como pais trocam seu material genético a partir desse ponto para dar origem a novos filhos, tal como ilustrado na Figura 3.6.

4.7. Operador Mutação

Neste trabalho, o operador mutação troca o valor de um determinado alelo de 0 para 1 e vice-versa, de acordo com um valor pré-definido de probabilidade (Pm), conforme ilustrado na Figura 3.7.

4.8. Fator *Generation* GAP

O fator GAP é um parâmetro que controla o número de indivíduos que será substituído a cada geração. Neste trabalho foi implementado o algoritmo *steady-state*, cuja principal característica é a substituição de um ou, no máximo, dois indivíduos por geração. Portanto, o valor de GAP utilizado no presente trabalho é obtido pela seguinte equação:

$$G = \frac{2}{n} \tag{4.6}$$

sendo *n* o tamanho total da população.

4.9. Algoritmo SSGA

Os principais passos computacionais do SSGA, implementados neste trabalho, são:

```
Início
< 1 > Inicialização de parâmetros: tamanho da  população,
      probabilidades de crossover e mutação, dentre outros;

< 2 > Semeação
     < 2.1 > População inicial é gerada aleatoriamente (utilizando
             uma representação binária);
     < 2.2 > População inicial é decodificada (Equações (3.9),
             (3.10) e (3.11));
     < 2.3 > Cálculo do offset;
     < 2.3 > Cálculo do valor fitness de cada indivíduo -
             Equação (4.4);

< 3 > Reprodução
     < 3.1 > Dois indivíduos são selecionados como pais (seleção por
             ranking);
     < 3.2 > Aplicação do operador crossover;
     < 3.3 > Aplicação do operador mutação;

< 4 > Atualização
     < 4.1 > Os dois novos indivíduos resultantes do processo de
             reprodução substituem os dois piores da população atual;

< 5 > Avaliação
     < 5.1 > Os dois novos cromossomos são decodificados;
     < 5.2 > Cálculo do offset;
     < 5.3 > O fitness dos dois novos cromossomos é calculado;

< 6 > Critério de parada
      Se satisfeito, ir para o passo 7; caso contrário, voltar
      para o passo 3(o critério de parada adotado foi o número
      máximo de iterações);

< 7 > Relatório (apresentação dos resultados obtidos);

Fim
```

4.10. Aplicação

Com o objetivo de testar a eficiência do algoritmo proposto, os procedimentos computacionais previamente descritos foram implementados em um programa de computação (escrito em linguagem C) e foram aplicados em vários problemas de otimização discreta de treliças e de otimização de funções contínuas, encontrados na literatura técnica.

Para obter resultados equivalentes e poder fazer as respectivas comparações, todos os problemas foram minimizados utilizando os mesmos

parâmetros no que se refere a tamanho de população, probabilidade de mutação e número de iterações. Os resultados são mostrados nas seções a seguir.

4.10.1. Treliça de 2 Barras

Problema: minimizar o peso da estrutura ilustrada na Figura 4.4, sujeita ao carregamento indicado. Dados: σ_{adm} (tensão admissível) = 25 ksi, u_{adm} (deslocamento admissível) = 2 in, E (módulo de elasticidade) = 10^4 ksi, ρ (peso específico) = 0.10 lb/in^3, P = 100 kips.

Os valores discretos, correspondentes às áreas das seções transversais das barras e que serão atribuídos às variáveis de projeto, podem ser escolhidos a partir da seguinte lista de perfis disponíveis: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.8, 3.0, 3.2 e 3.4 (in^2). Este problema foi estudado por Fonseca e das Neves [30].

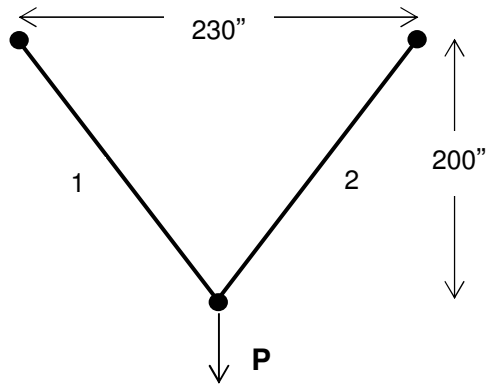


Figura 4.4 – Treliça de 2 barras.

Configurações do AG utilizado:

- População inicial aleatória
- Tamanho da população: 8
- Número máximo de iterações: 100
- Probabilidade de mutação Pm: 0.01
- Coeficiente de penalidade C_i, Equação (3.4): 11

Tempo de execução: 0.01 seg.

Após 100 iterações, o peso mínimo de 110.73 lb foi obtido na iteração 89. A Figura 4.5 mostra o processo de convergência discreta para o problema em

estudo. Os resultados obtidos, bem como as respectivas comparações, são mostrados na Tabela 4.1.

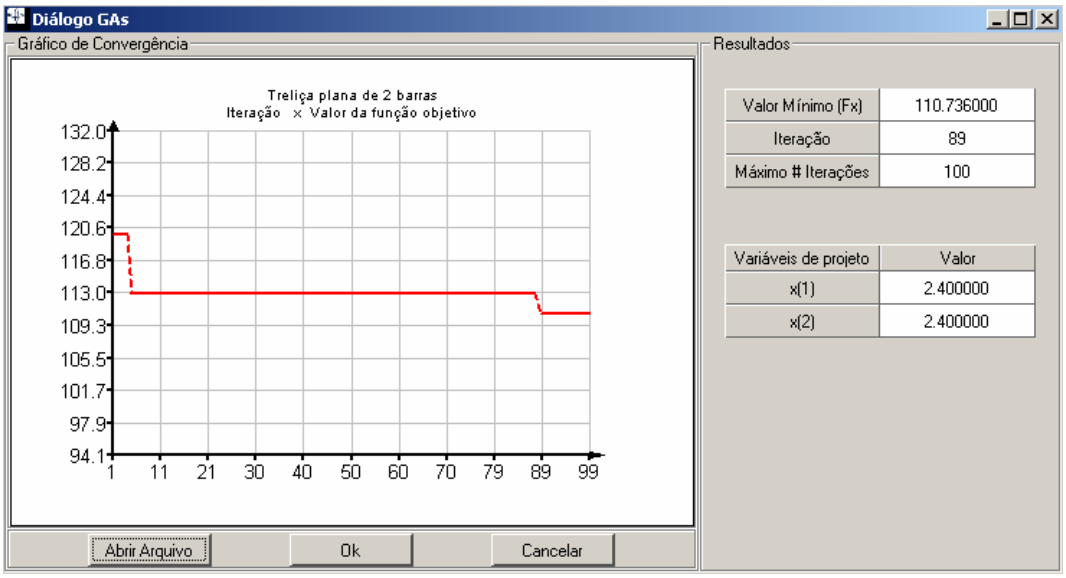


Figura 4.5 – Gráfico de convergência para a otimização discreta da treliça de 2 barras.

Tabela 4.1 – Comparação de resultados obtidos com a treliça de 2 barras.

Variáveis (in^2)	**SSGA**	**Ref. [30]**
A_1	2.40	2.40
A_2	2.40	2.40
Peso (lb)	**110.73**	**110.73**

4.10.2. Treliça de 3 Barras

Problema: minimizar o peso da estrutura ilustrada na Figura 4.6, sujeita ao carregamento indicado. Dados: σ_{adm} = 147.15 Mpa, u_{adm} = 5.08 mm, E = 2.008 x 10^5 Mpa, ρ = 7.85 x 10^3 Kgf/m^3, P = 100 kips. As barras 1 e 2 apresentam a mesma área de seção transversal, isto é, $A_1 = A_2$. Os valores discretos, correspondentes às áreas das seções transversais das barras e que serão atribuídos às variáveis de projeto, podem ser escolhidos a partir da seguinte lista de perfis disponíveis: 1.2, 1.4, 1.6, 1.8, 2.0, 2.2, 2.4, 2.6, 2.8, 3.0, 3.2, 3.4, 3.6, 3.8, 4.0 e 4.4 (cm^2). Este problema foi estudado por Rajeev e Krishnamoorthy [31].

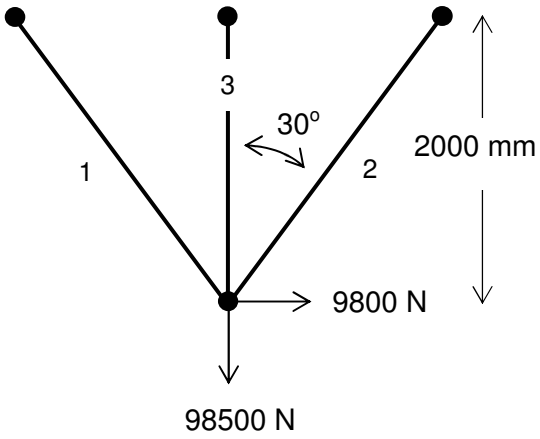


Figura 4.6 – Treliça de 3 barras.

Configurações do AG utilizado:

- População inicial aleatória
- Tamanho da população: 10
- Número máximo de iterações: 200
- Probabilidade de mutação Pm: 0.01
- Coeficiente de penalidade C_i, Equação (3.4): 11

Tempo de execução: 0.016 seg.

Após 200 iterações, o peso mínimo de 14.76 kgf foi obtido na iteração 158. A Figura 4.7 mostra o processo de convergência discreta para este exemplo. Os resultados obtidos, bem como as respectivas comparações, são mostrados na Tabela 4.2.Tabela 4.2

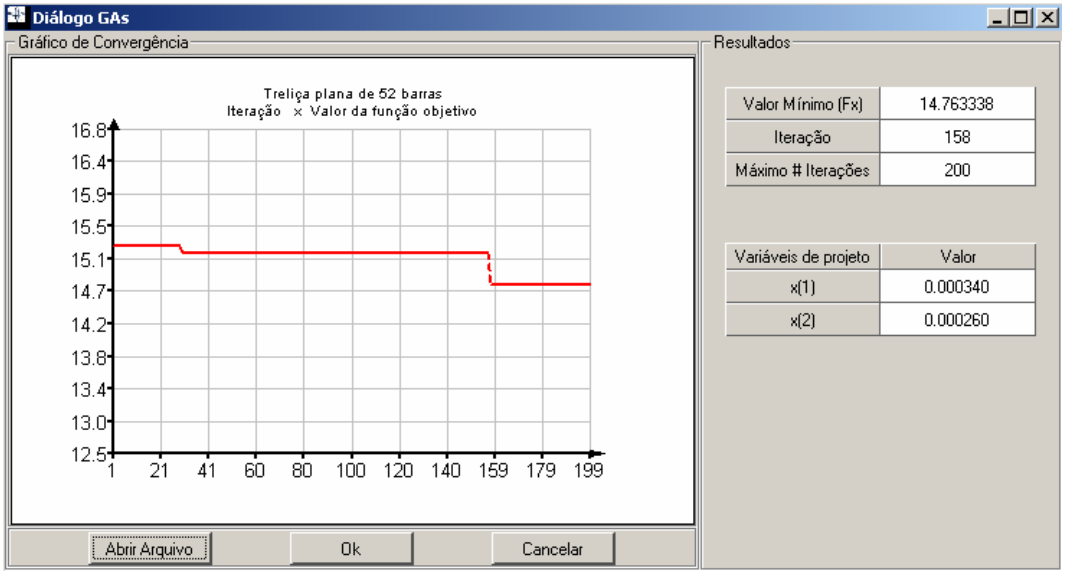


Figura 4.7 – Gráfico de convergência para a otimização discreta da treliça de 3 barras

Tabela 4.2 – Comparação de resultados obtidos com a treliça de 3 barras.

Variáveis (cm^2)	SSGA	Ref. [31]
A_1	2.6	2.4
A_2	2.6	2.4
A_3	3.4	3.8
Peso (Kgf)	**14.76**	14.77

4.10.3. Treliça de 10 Barras

Problema: minimizar o peso da estrutura ilustrada na Figura 4.8 sujeita ao carregamento indicado. Dados: σ_{adm} = 25 ksi, u_{adm} = 2 in, E = 10^4 ksi, ρ = 0.10 lb/in^3, P = 100 kips.

Os valores discretos, correspondentes às áreas das seções transversais das barras, foram obtidos a partir do Manual do Instituto Americano de Construções de Aço (*American Institute of Steel Construction Manual*). Foram considerados os seguintes valores: 1.62, 1.80, 1.99, 2.13, 2.38, 2.62, 2.63, 2.88, 2.93, 3.09, 3.13, 3.38, 3.47, 3.55, 3.63, 3.84, 3.87, 3.88, 4.18, 4.22, 4.49, 4.59, 4.80, 4.97, 5.12, 5.74, 7.22, 7.97, 11.5, 13.5, 13.9, 14.2, 15.5, 16.0, 16.9, 18.8, 19.9, 22.0, 22.9, 26.5, 30.0 e 33.5 (in^2). Este problema foi estudado por Rajeev e Krishnamoorthy [31].

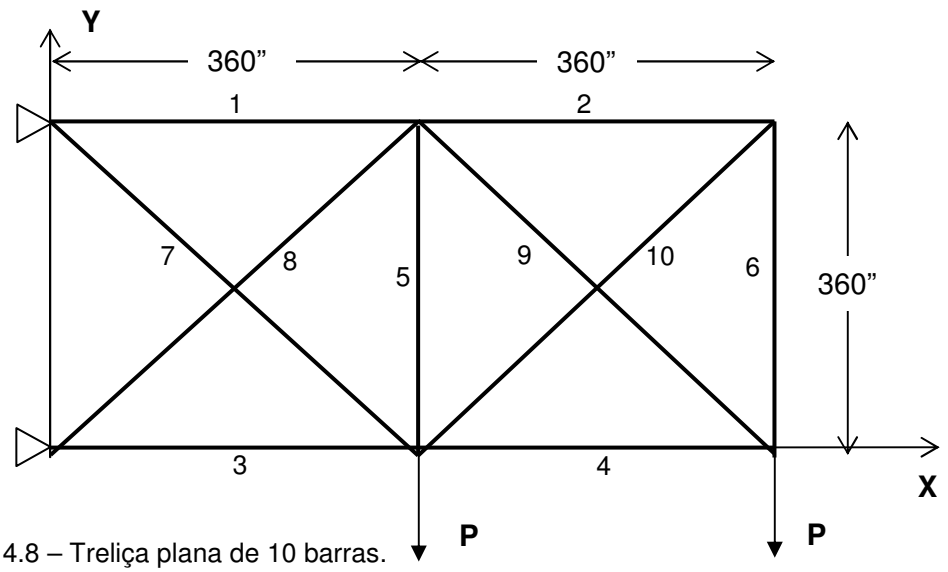


Figura 4.8 – Treliça plana de 10 barras.

Configurações do AG utilizado:

- População inicial aleatória
- Tamanho da população: 40
- Número máximo de iterações: 10000
- Probabilidade de mutação Pm: 0.01
- Coeficiente de penalidade C_i, Equação (3.4): 11

Tempo de execução: 19.84 seg.

A Figura 4.9 mostra o processo de convergência discreta para este exemplo. O resultados obtidos no processo de otimização, juntamente com as respectivas comparações, são apresentados na Tabela 4.3.

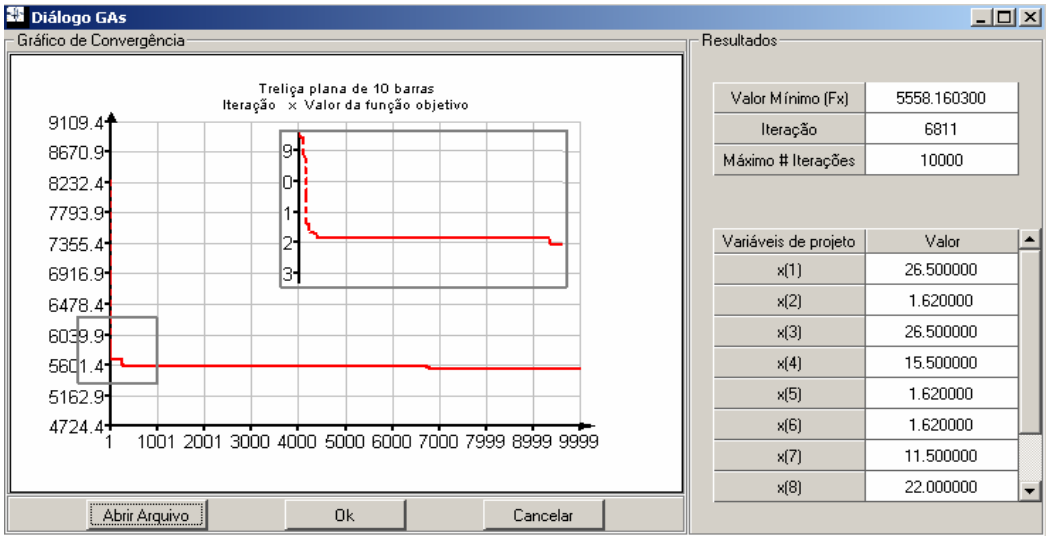


Figura 4.9 – Gráfico de convergência para a otimização discreta da treliça de 10 barras.

Tabela 4.3 – Comparação de resultados obtidos com a treliça de 10 barras.

Variáveis (in^2)	**SSGA**	**Ref [31]**
A_1	26.50	33.50
A_2	1.62	1.62
A_3	26.50	22.00
A_4	15.50	15.50
A_5	1.62	1.62
A_6	1.62	1.62
A_7	11.50	14.20
A_8	22.00	19.90
A_9	22.00	19.90
A_{10}	1.80	2.62
Peso (lb)	**5558.16**	5613.84

4.10.4.
Treliça Espacial de 25 Barras

Problema: minimizar o peso da estrutura ilustrada na Figura 4.10 sujeita ao carregamento indicado. Dados: σ_{adm} = 40000 psi, u_{adm} = 0.35 in, E = 10^7 psi, ρ = 0.10 lb/in^3.

A otimização deve ser realizada mantendo-se a relação de simetria da estrutura em relação aos planos y-z e x-z. Oito variáveis de projeto são usadas para dimensionar os 25 elementos da treliça, conforme ilustrado na Tabela 4.4.

Os valores discretos, correspondentes às áreas das seções transversais das barras e que serão atribuídos às variáveis de projeto, podem ser escolhidos a partir da seguinte lista de perfis disponíveis: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.8, 3.0, 3.2 e 3.4 (in^2). Os carregamentos aplicados são mostrados na Tabela 4.5. Este problema foi estudado por Wu e Chow [18].

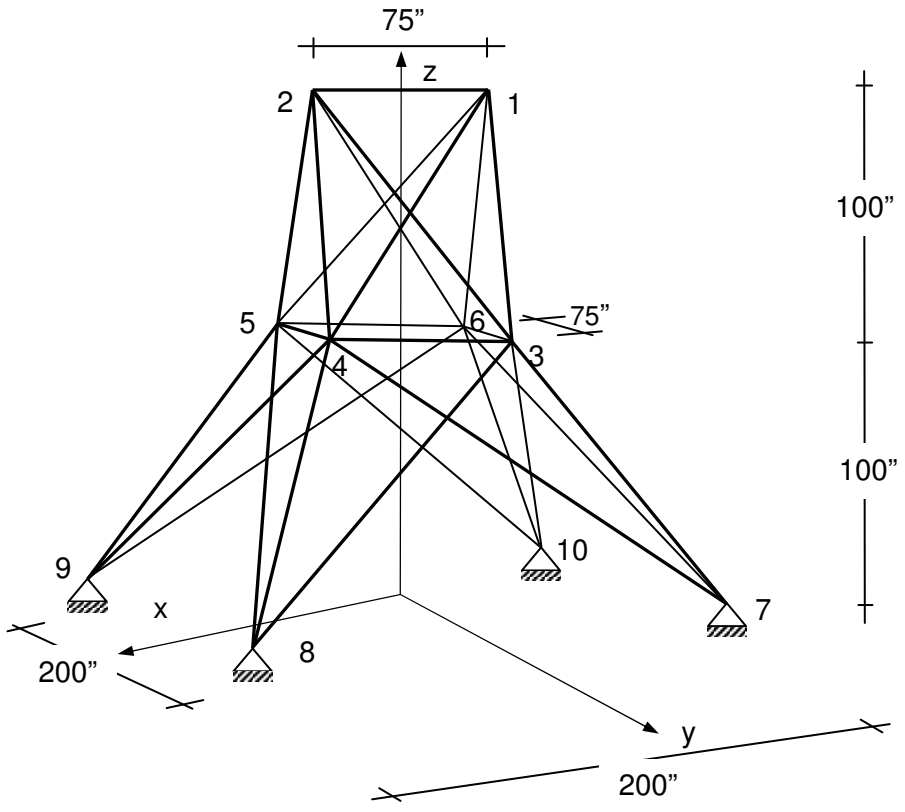


Figura 4.10 – Treliça espacial de 25 barras.

Tabela 4.4 – Grupos de elementos da treliça espacial de 25 barras.

Grupo	Elemento (Nó inicial – Nó final)
A_1	1 (1,2)
A_2	2 (1,4), 3 (2,3), 4(1,5), 5(2,6)
A_3	6(2,5), 7(2,4), 8(1,3), 9(1,6)
A_4	10(3,6), 11(4,5)
A_5	12(3,4), 13(5,6)
A_6	14(3,10), 15(6,7), 16(4,9), 17(5,8)
A_7	18(3,8), 19(4,7), 20(6,9), 21(5,10)
A_8	22(3,7), 23(4,8), 24(5,9), 25(6,10)

Tabela 4.5 – Detalhe de cargas da treliça espacial de 25 barras.

Nó	P_x (lb)	P_y (lb)	P_z (lb)
1	1000	-10000	-10000
2	0	-10000	-10000
3	500	0	0
4	600	0	0

Configurações do AG utilizado:

- População inicial aleatória
- Tamanho da população: 60
- Número máximo de iterações: 30000
- Probabilidade de mutação Pm: 0.01
- Coeficiente de penalidade C_i, Equação (3.4): 60

Tempo de execução: 174.78 seg.

Após 30000 iterações o valor mínimo de 485.04 lb foi obtido na iteração 20465. A Figura 4.9 mostra o processo de convergência do problema de otimização discreta da treliça espacial de 25 barras. Os resultados obtidos são mostrados na Tabela 4.6, bem como as comparações com os resultados da referência [18].

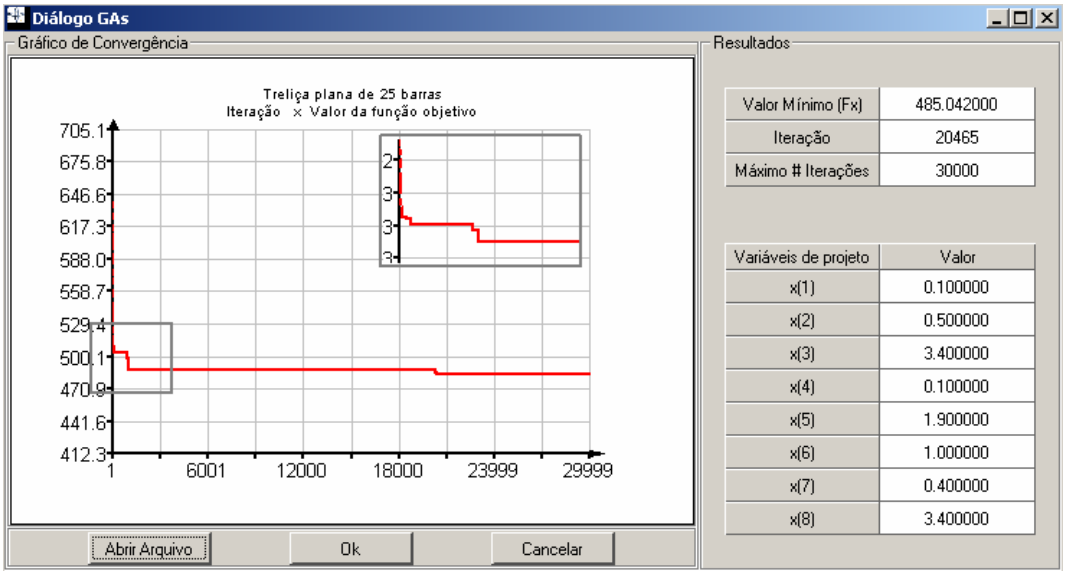


Figura 4.11 – Gráfico de convergência para a otimização discreta da treliça espacial de 25 barras.

Tabela 4.6 – Comparação de resultados obtidos com a treliça espacial de 25 barras.

Variáveis (in^2)	**SSGAs**	**Ref. [18]**
A_1	0.1	0.1
A_2	0.5	0.5
A_3	3.4	3.4
A_4	0.1	0.1
A_5	1.9	1.5
A_6	1.0	0.9
A_7	0.4	0.6
A_8	3.4	3.4
Peso (lb)	**485.04**	486.29

4.10.5. Treliça Plana de 52 Barras

Problema: minimizar o peso da estrutura ilustrada na Figura 4.12 sujeita ao carregamento indicado. Dados: σ_{adm} = 180 Mpa, E = 2.07 x 10^5 Mpa, ρ = 7860.0 kgf/m^3, P_x = 100 kN, P_y = 200 kN.

Os 52 elementos foram distribuídos em 12 grupos. Os valores discretos, correspondentes às áreas das seções transversais das barras e que podem ser atribuídos às variáveis de projeto, são mostrados na Tabela 4.7. Este problema foi estudado por Wu e Chow [18].

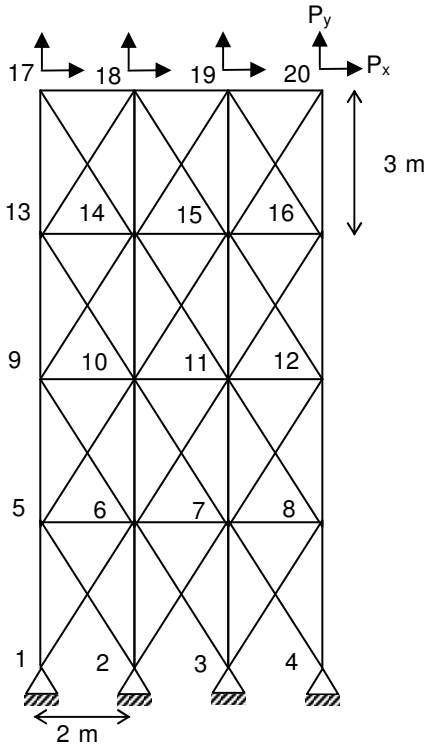


Figura 4.12 – Treliça plana de 52 barras.

Configurações do AG utilizado:

- População inicial aleatória
- Tamanho da população: 60
- Número máximo de iterações: 60000
- Probabilidade de mutação Pm: 0.01
- Coeficiente de penalidade C_i, Equação (3.4): 60

Tempo de execução: 1138.06 seg.

Após 60000 iterações o peso mínimo de 1963.75 kgf foi obtido na iteração 41109. As Figura 4.13 e Figura 4.14 mostram os gráficos de convergência do processo de otimização. Uma comparação dos resultados obtidos com os da referência [18] é apresentada na Tabela 4.8. Observe-se que este problema foi resolvido utilizando-se duas versões do operador *crossover*: de um ponto (1X) e de dois pontos (2X), respectivamente, conforme ilustrado na Tabela 4.8.

Tabela 4.7 – Seções disponíveis para o exemplo da treliça de 52 barras.

No.	in^2	mm^2	No.	in^2	mm^2
1	0.111	71.613	33	3.840	2477.414
2	0.141	90.968	34	3.870	2496.769
3	0.196	126.451	35	3.880	2503.221
4	0.250	161.290	36	4.180	2696.769
5	0.307	198.064	37	4.220	2722.575
6	0.391	252.258	38	4.490	2896.768
7	0.442	285.161	39	4.590	2961.284
8	0.563	363.225	40	4.800	3096.768
9	0.602	388.386	41	4.970	3206.445
10	0.766	494.193	42	5.120	3303.219
11	0.785	506.451	43	5.740	3703.218
12	0.994	641.289	44	7.220	4658.055
13	1.000	645.160	45	7.970	5141.925
14	1.228	792.256	46	8.530	5503.215
15	1.266	816.773	47	9.300	5999.988
16	1.457	940.000	48	10.850	6999.986
17	1.563	1008.385	49	11.500	7419.340
18	1.620	1045.159	50	13.500	8709.660
19	1.800	1161.288	51	13.900	8967.724
20	1.990	1283.868	52	14.200	9161.272
21	2.130	1374.191	53	15.500	9999.980
22	2.380	1535.481	54	16.000	10322.560
23	2.620	1690.319	55	16.900	10903.204
24	2.630	1696.771	56	18.800	12129.008
25	2.880	1858.061	57	19.900	12838.684
26	2.930	1890.319	58	22.000	14193.520
27	3.090	1993.544	59	22.900	14774.164
28	3.130	2019.351	60	24.500	15806.420
29	3.380	2180.641	61	26.500	17096.740
30	3.470	2238.705	62	28.000	18064.480
31	3.550	2290.318	63	30.000	19354.800
32	3.630	2341.931	64	33.500	21612.860

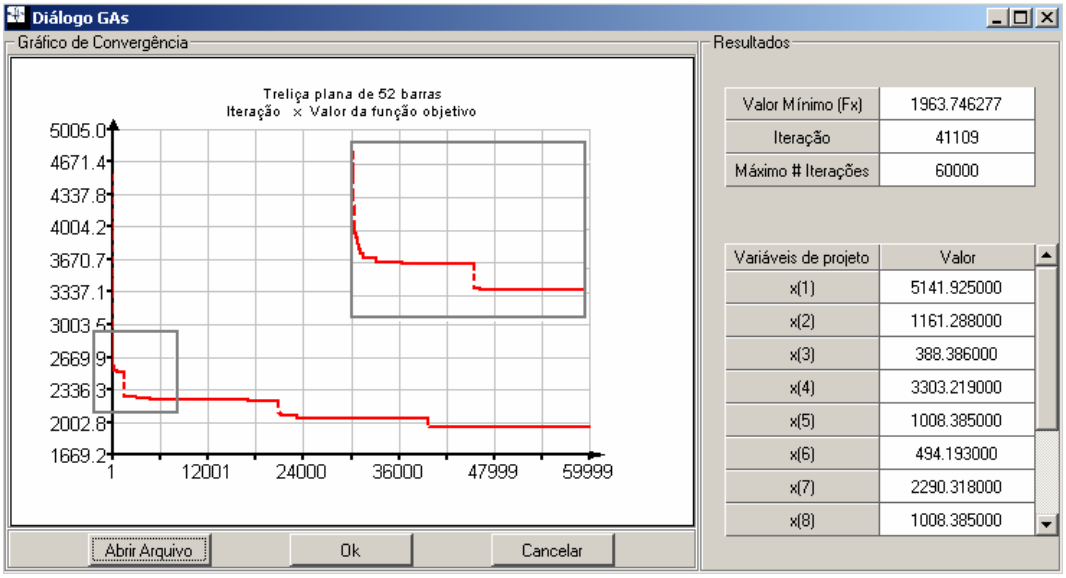


Figura 4.13 – Gráfico de convergência para a otimização discreta da treliça plana de 52 barras – *crossover* de um ponto (1X).

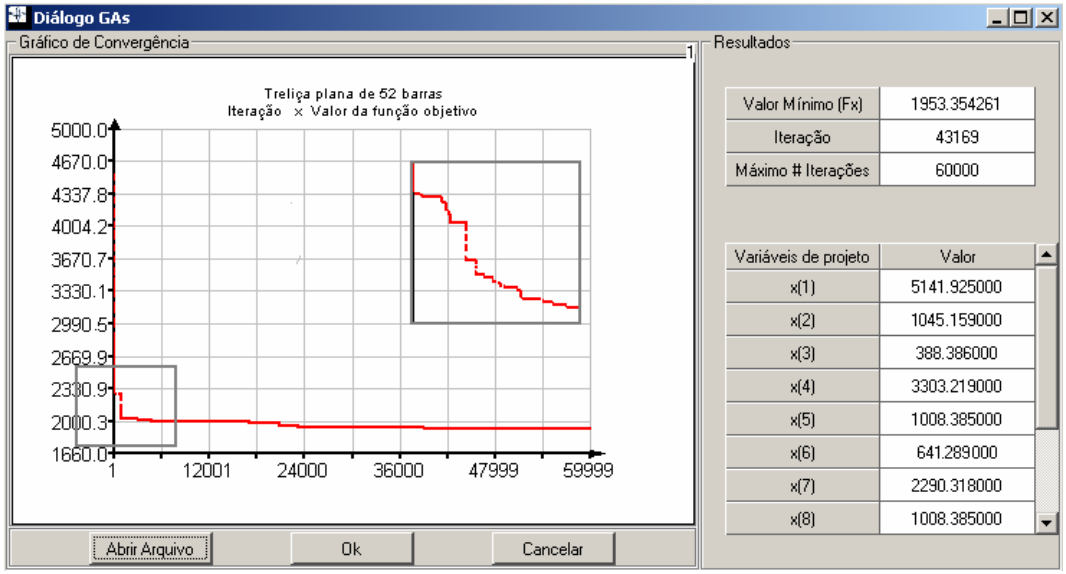


Figura 4.14 – Gráfico de convergência para a otimização discreta da treliça plana de 52 barras – *crossover* de dois pontos (2X).

Tabela 4.8 – Comparações de resultados obtidos com a treliça plana de 52 barras.

Variáveis	SSGA		Ref. [18]	
	IX	2X	IX	2X
A_1(1-4)	5141.925	5141.925	3703.218	4658.055
A_2(5-10)	1045.159	1045.159	2722.575	1161.288
A_3(11-13)	285.161	388.386	1858.061	645.160
A_4(14-17)	3303.219	3303.219	3206.445	3303.219
A_5(18-23)	1161.288	1008.385	1008.385	1045.159
A_6(24-26)	494.193	645.289	1008.385	494.193
A_7(27-30)	2290.318	2290.318	2477.41	2477.414
A_8(31-36)	1008.385	1008.385	1008.385	1045.159
A_9(37-39)	363.225	494.193	388.386	363.225
A_{10}(40-43)	1690.319	1283.86	2477.414	1696.771
A_{11}(44-49)	1008.385	1161.288	1008.385	1045.159
A_{12}(50-52)	388.386	388.386	1008.385	792.256
Peso (Kgf)	**1963.75**	**1953.35**	2294.521	1970.142

4.10.6. Funções Contínuas

Nesta seção, o algoritmo proposto é testado com relação às funções contínuas apresentadas no trabalho desenvolvido por André *et al*. [24].

Os gráficos de convergência são ilustrados nas Figuras 4.15 a 4.19, e as comparações dos resultados obtidos com a referência [24] são mostrados na Tabela 4.9.

- **F1**

$$f(x) = 2(x-0.75)^2 + \sin(5\pi x - 0.4\pi) - 0.125$$

$$onde \;\; 0 \le x \le 1$$

População inicial aleatória

Tamanho da população: 60

Número máximo de iterações: 1000

Probabilidade de mutação Pm: 0.05

Tempo de execução: 0.34 seg.

Após 1000 iterações, o valor mínimo de -1.12323 foi obtido na iteração 98.

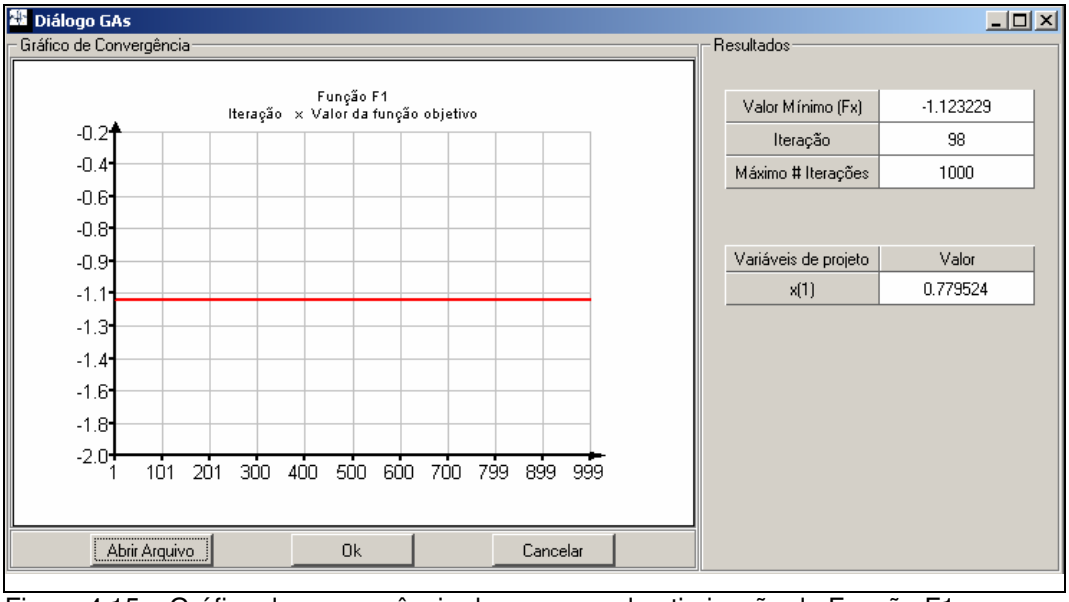


Figura 4.15 – Gráfico de convergência do processo de otimização da Função F1.

- **F3**

$$f(x) = -\sum_{j=1}^{5} \{ j \sin[(j+1)x + j] \}$$

$$onde \quad -10 \le x \le 10$$

População inicial aleatória

Tamanho da população: 60

Número máximo de iterações: 1000

Probabilidade de mutação Pm: 0.05

Tempo de execução: 0.45 seg.

Após 1000 iterações, o valor mínimo de -12.03125 foi obtido na iteração 397.

- **Goldprice**

$$f(x,y) = [1+(x+y+1)^2(19-14x+3x^2-14y+6xy+3y^2)]\cdot$$
$$[30+(2x-3y)^2(18-32x+12x^2+48y-36xy+27y^2)]$$

$$onde \quad -2 \le x \le 2 \quad e \quad -2 \le y \le 2$$

População inicial aleatória

Tamanho da população: 60

Número máximo de iterações: 1000

Probabilidade de mutação Pm: 0.01

Tempo de execução: 0.64 seg.

Após 1000 iterações, o valor mínimo de 3.00000 foi obtido na iteração 149.

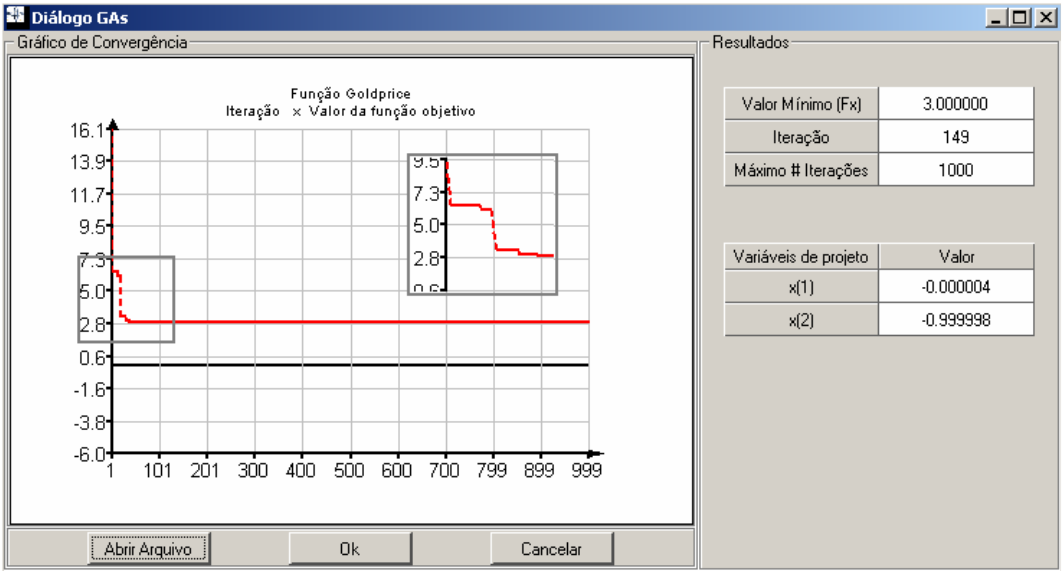


Figura 4.16 – Gráfico de convergência do processo de otimização da Função Goldprice.

- **Quartic**

$$f(x,y) = \frac{x^4}{4} - \frac{x^2}{2} + \frac{x}{10} + \frac{y^2}{2}$$

$$onde \quad -10 \le x \le 10 \quad e \quad -10 \le y \le 10$$

População inicial aleatória

Tamanho da população: 60

Número máximo de iterações: 500

Probabilidade de mutação Pm: 0.01

Tempo de execução: 0.35 seg

Após 500 iterações, o valor mínimo de -0.35239 foi obtido na iteração 411.

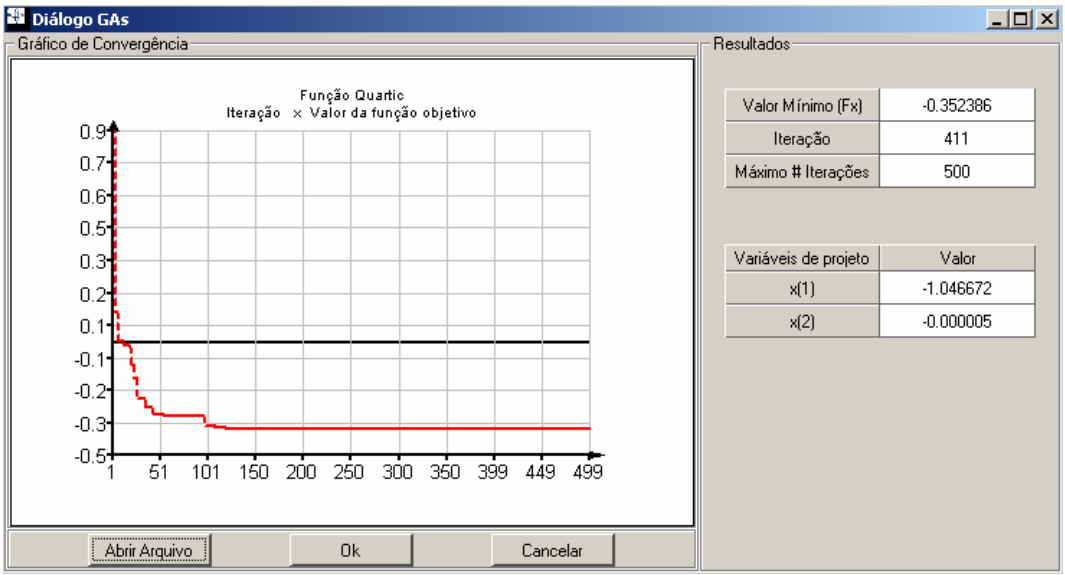


Figura 4.17 – Gráfico de convergência do processo de otimização da Função Quartic.

- **Shubert**

$$f(x, y) = \sum_{i=1}^{5} i \cos[(i+1)x + i] \cdot \sum_{i=1}^{5} i \cos[(i+1)y + i]$$

$$onde \quad -10 \le x \le 10 \quad e \quad -10 \le y \le 10$$

População inicial aleatória

Tamanho da população: 60

Número máximo de iterações: 650

Probabilidade de mutação Pm: 0.01

Tempo de execução: 0.58 seg.

Após 650 iterações, o valor mínimo de -186.73091 foi obtido na iteração 633.

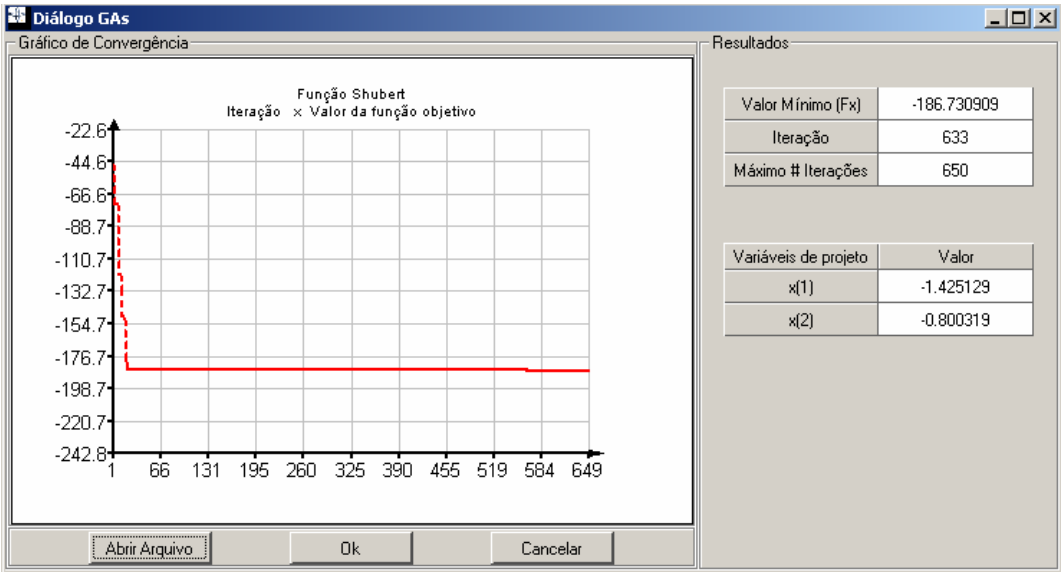


Figura 4.18 – Gráfico de convergência do processo de otimização da Função Shubert.

- **Brown3**

$$f(x) = \sum_{i=1}^{19} [(x_i^2)^{(x_{i+1}^2+1)} + (x_{i+1}^2)^{(x_i^2+1)}]$$

$$onde\ x = (x_1, ..., x_{20})^T \quad e \quad -1 \le x_i \le 4 \ \ para\ 1 \le i \le 20$$

População inicial aleatória

Tamanho da população: 100

Número máximo de iterações: 120000

Probabilidade de mutação Pm: 0.01

Tempo de execução: 1270.8 seg.

Após 120000 iterações, o valor mínimo de 0.010848 foi obtido na iteração 95624.

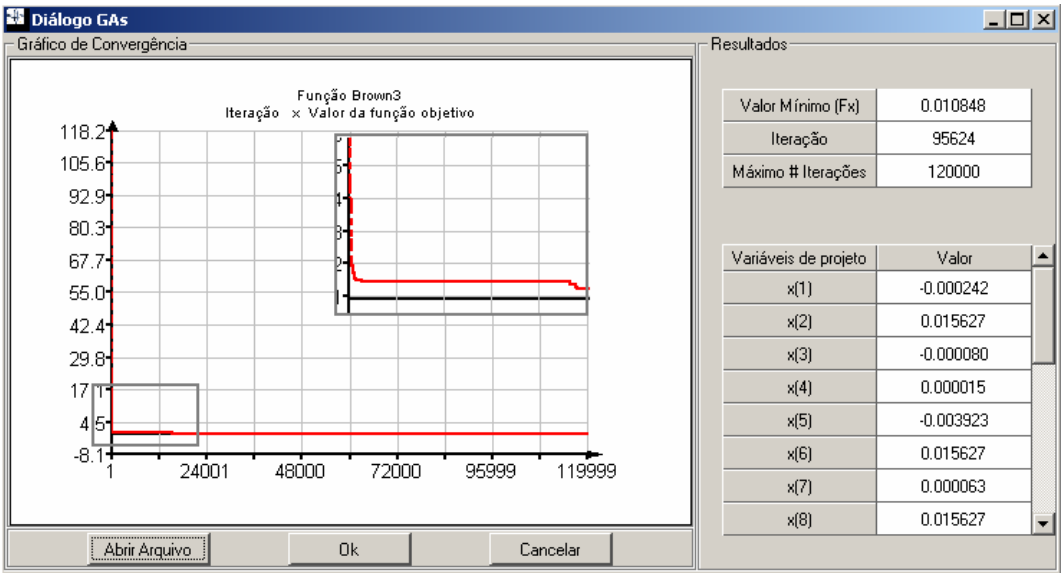


Figura 4.19 – Gráfico de convergência do processo de otimização da Função Brown3.

Tabela 4.9 – Comparações dos resultados obtidos com as funções contínuas.

Função	Mínimo Teórico	SSGA		Ref. [24]	
		Iteração	Mínimo	Iteração	Mínimo
F1	-1.12323	98	**-1.12323**	784	**-1.12323**
F3	-12.03125	397	**-12.03125**	744	-12.03120
Goldprice	3.00000	149	**3.00000**	4632	3.00028
Quartic	-0.35239	411	**-0.35239**	3168	-0.35238
Shubert	-186.73091	633	**-186.73091**	2364	-186.72802
Brown3	0.00000	95624	**0.010848**	106589	0.67464

4.10.7. Comparações com o algoritmo do ICA

Com o objetivo de se ter um outro parâmetro de comparação e verificação da eficiência do algoritmo desenvolvido no presente trabalho, as funções contínuas da seção anterior foram minimizadas utilizando-se um algoritmo genético disponibilizado pelo grupo ICA (Inteligência de Computação Aplicada) do Departamento de Engenharia Elétrica da PUC-Rio [32]. Neste algoritmo os operadores de *crossover* e mutação funcionam com taxas diferentes ao longo do processo de convergência. Essas taxas indicam a probabilidade com que tais operadores serão selecionados e executados.

A Tabela 4.10 apresenta as comparações dos resultados obtidos.

Tabela 4.10 – Comparação de resultados de funções contínuas com referencia [32].

Função	Mínimo Teórico	SSGA			ICA [32]		
		Tempo (seg)	Iter.	Mín.	Iter.	Mín.	Tempo (seg)
F1	-1.12323	0.34	98	**-1.12323**	104	**-1.12323**	2.06
F3	-12.03125	0.45	397	**-12.03125**	166	**-12.03125**	1.91
Goldprice	3.00000	0.64	149	**3.00000**	390	**3.00000**	1.88
Quartic	-0.35239	0.35	411	**-0.35239**	253	**-0.35239**	0.88
Shubert	-186.73091	0.58	633	**-186.73091**	644	**-186.73091**	1.27
Brown3	0.00000	1270.8	95624	0.010848	12000	**0.00000**	1458.5

5
Aplicações

5.1.
Introdução

Com o objetivo de comprovar a eficiência computacional do algoritmo proposto, este capítulo apresenta exemplos de aplicações na otimização do posicionamento (disposição) das linhas em sistemas de ancoragem reais, utilizados pela Petrobras em operações de exploração de petróleo. Os problemas consistem de unidades flutuantes submetidas a diferentes combinações de condições ambientais para as quais se deseja encontrar as disposições das linhas de ancoragem de forma a minimizar os deslocamentos sofridos pelas unidades flutuantes.

5.2.
Exemplo 1 – Plataforma semi-submersível

Este exemplo corresponde a uma unidade flutuante (plataforma semi-submersível) ancorada por 8 linhas, conforme ilustrado na Figura 5.1; cada linha é composta por 3 diferentes tipos de materiais cujas propriedades são apresentadas na Tabela 5.1 (nesta tabela, Φ representa o diâmetro das linhas, EA é a rigidez axial e Wa representa o peso submerso por unidade de comprimento). As restrições laterais, para cada linha, são mostradas na Tabela 5.2.

Dados Gerais:

Tamanho da população: 30

Número máximo de iterações: 10000

Probabilidade de mutação Pm: 0.05

Lâmina d'água: 917 m

Tabela 5.1 – Propriedades dos materiais do Exemplo 1.

Material (unidade-âncora)	Φ (in)	Comprimento (m)	EA (kN)	Wa (kN/m)	Tensão de Ruptura (kN)
1 – Amarra (Corrente)	3	250	549136	1.07866	4884
2 – Cabo (Poliéster)	3.5	1200	254956	0.27555	5520
3 – Amarra (Corrente)	3	990	549136	1.07866	4884

Tabela 5.2 – Restrições laterais das variáveis de projeto (Exemplo 1).

Linha	Restrição inferior (em graus)	Restrição superior (em graus)
1	0	45
2	45	90
3	90	135
4	135	180
5	180	225
6	225	270
7	270	315
8	315	360

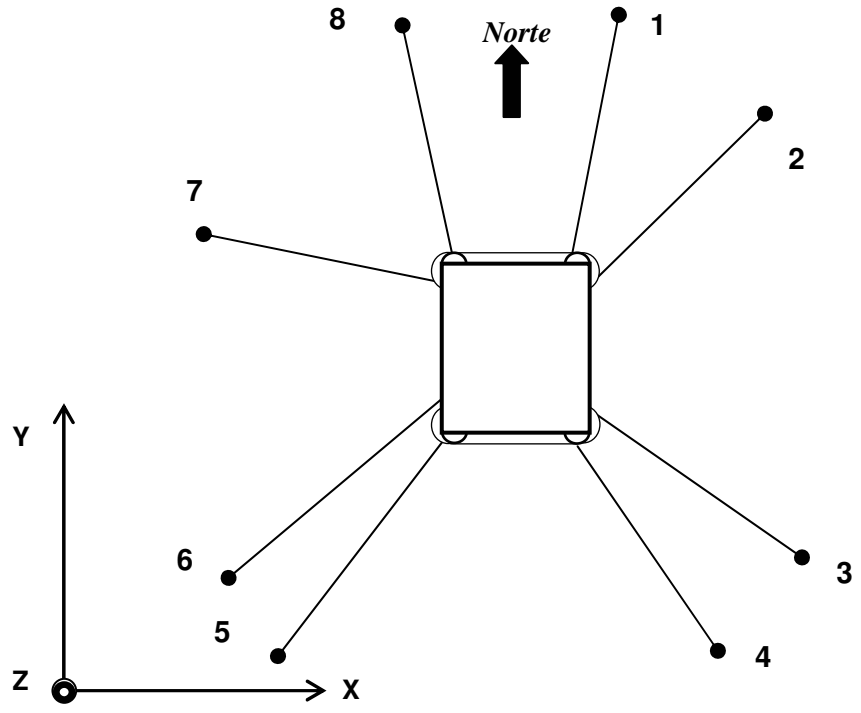


Figura 5.1 – Vista de topo da unidade flutuante (Exemplo 1).

A unidade flutuante foi submetida a um conjunto de condições ambientais combinadas de acordo com um critério de colinearidade, i.e., com ventos, correntes e ondas atuando simultaneamente na mesma direção, em cada caso. Neste exemplo, oito combinações foram consideradas. Para cada combinação, uma força externa resultante de 2000 kN foi aplicada à unidade flutuante, conforme ilustrado na Tabela 5.3.

Tabela 5.3 – Forças externas atuando sobre a unidade flutuante (Exemplo 1).

Caso	**Azimute**	**Direção**
1	0	N
2	45	NE
3	90	E
4	135	SE
5	180	S
6	225	SW
7	270	W
8	315	NW

Tempo de execução: 20356.78 seg.

Após 10000 iterações, um valor mínimo da função objetivo de 62210 m^2 foi obtido na iteração 9077, assim como ilustrado na Figura 5.2. A Tabela 5.4 apresenta os azimutes finais de cada linha de ancoragem, e a disposição final das linhas é ilustrada na Figura 5.3.

Tabela 5.4 – Azimutes finais da unidade flutuante (Exemplo 1).

Linha	**Azimute - α_i (em graus)**
1	22.34
2	67.68
3	112.34
4	157.68
5	202.77
6	247.38
7	292.62
8	337.23

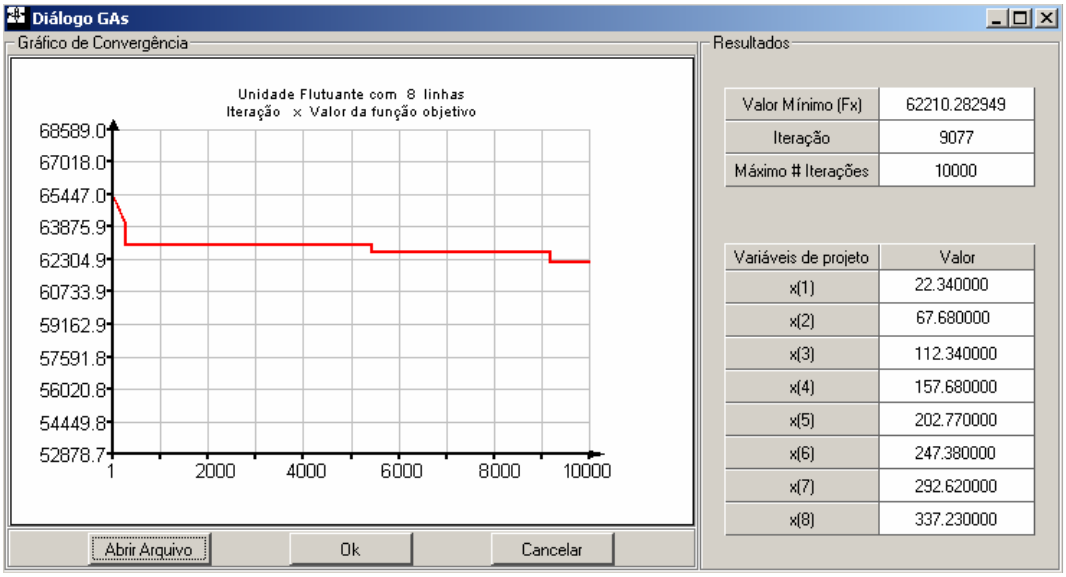


Figura 5.2 – Gráfico de convergência do processo de otimização (Exemplo 1).

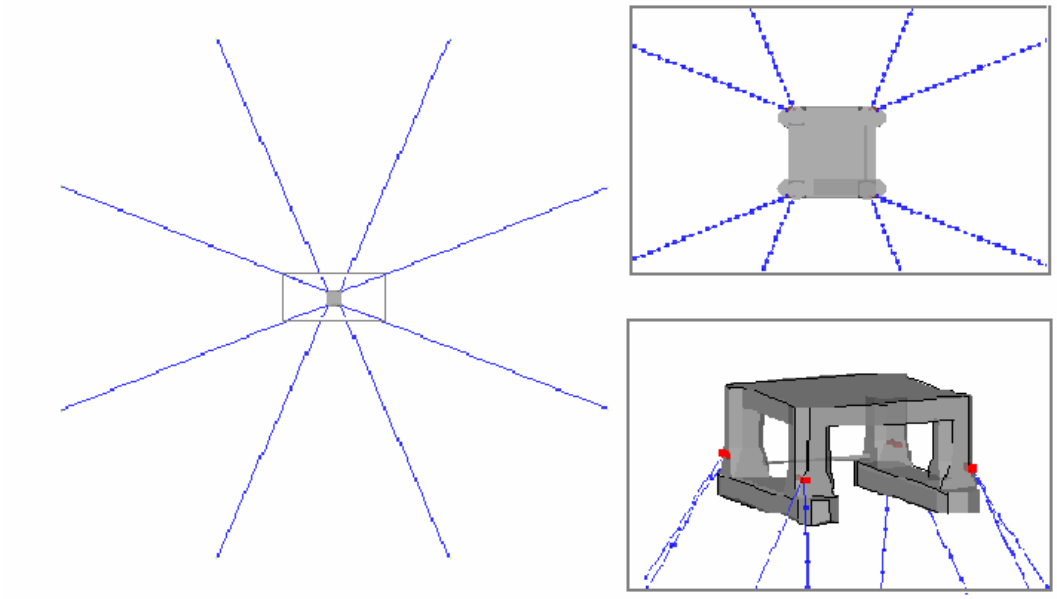


Figura 5.3 – Disposição final das linhas de ancoragem (Exemplo 1).

5.3. Exemplo 2 – FPSO

Este exemplo corresponde a uma unidade flutuante do tipo FPSO [33] ancorada por 18 linhas, conforme ilustrado na Figura 5.4. Cada linha é composta por 3 diferentes tipos de materiais cujas propriedades são apresentadas na Tabela 5.5. As 18 linhas foram divididas em 6 grupos, de acordo com suas restrições laterais, conforme mostrado na Tabela 5.6.

Dados Gerais:

Tamanho da população: 30

Número máximo de iterações: 10000

Probabilidade de mutação Pm: 0.01

Lâmina d'água: 1260 m

Tabela 5.5 – Propriedades dos materiais do Exemplo 2.

Material (unidade-âncora)	Φ (mm)	Comprimento (m)	EA (kN)	Wa (kN/m)	Tensão de Ruptura (kN)
1 – Amarra (Corrente)	120	250	854427	2.4580	13573
2 – Cabo (Poliéster)	225	1500	222113	0.0859	13734
3 – Amarra (Corrente)	120	500	854427	2.4580	13573

Tabela 5.6 – Restrições laterais das variáveis de projeto (Exemplo 2).

Grupo	Linhas	Restrição inferior (em graus)	Restrição superior (em graus)
1	1, 2, 3	0	45
2	4, 5	45	90
3	6, 7	90	135
4	8, 9, 10	135	180
5	11, 12, 13, 14	202.5	247.5
6	15, 16, 17, 18	292.5	337.5

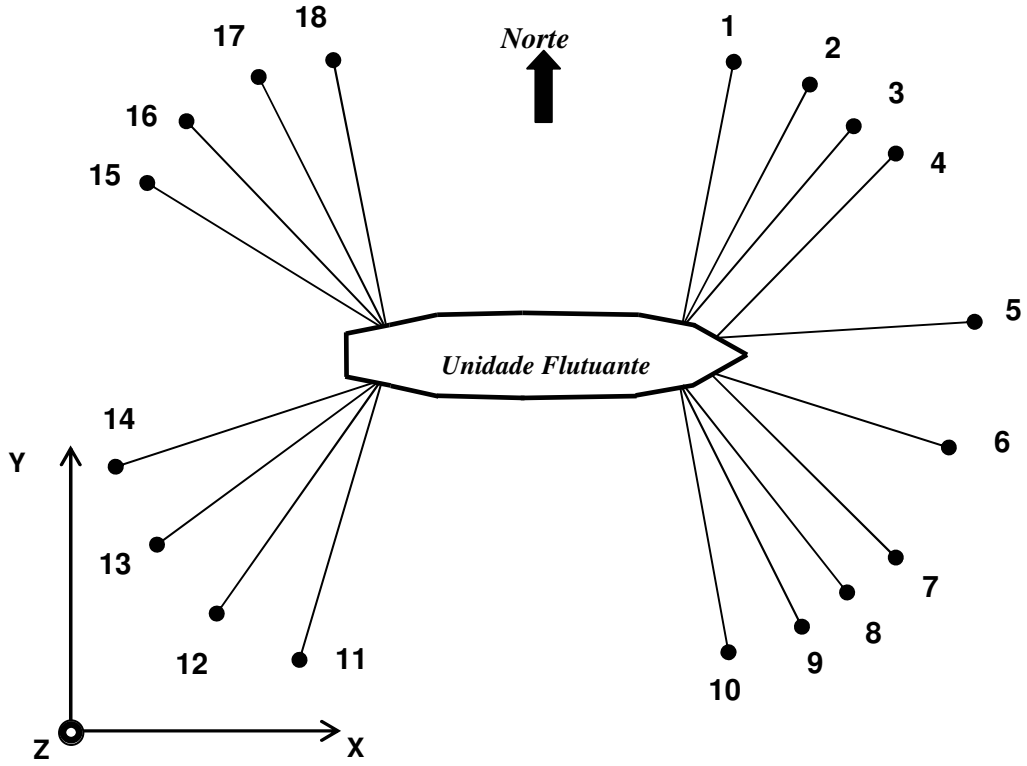


Figura 5.4 – Vista de topo da unidade flutuante (Exemplo 2).

Assim como exemplo anterior, a unidade flutuante foi submetida a um conjunto de condições ambientais combinadas de acordo com um critério de colinearidade, i.e., com ventos, correntes e ondas atuando simultaneamente na mesma direção, para cada caso. Neste exemplo também foram consideradas oito combinações. Para cada combinação, uma força externa resultante de 5000 kN foi aplicada à unidade flutuante, como ilustrado na Tabela 5.7.

Tabela 5.7 – Forças externas atuando sobre a unidade flutuante (Exemplo 2).

Caso	Azimute	Direção
1	0	N
2	45	NE
3	90	E
4	135	SE
5	180	S
6	225	SW
7	270	W
8	315	NW

Tempo de execução: 7221.93 seg.

Após 10000 iterações, um valor mínimo da função objetivo de 9120 m^2 foi obtido na iteração 3192, conforme ilustrado na Figura 5.5. A Tabela 5.8 apresenta os azimutes finais de cada linha de ancoragem, e a disposição final das linhas é ilustrada na Figura 5.6.

Tabela 5.8 – Azimutes finais da unidade flutuante (Exemplo 2).

Linha	**Azimute - α_i (em graus)**	**Linha**	**Azimute - α_i (em graus)**
1	21.5	10	158.5
2	24.5	11	225.5
3	27.5	12	228.5
4	63.0	13	231.5
5	66.0	14	234.5
6	114.0	15	305.5
7	117.0	16	308.5
8	152.5	17	311.5
9	155.5	18	314.5

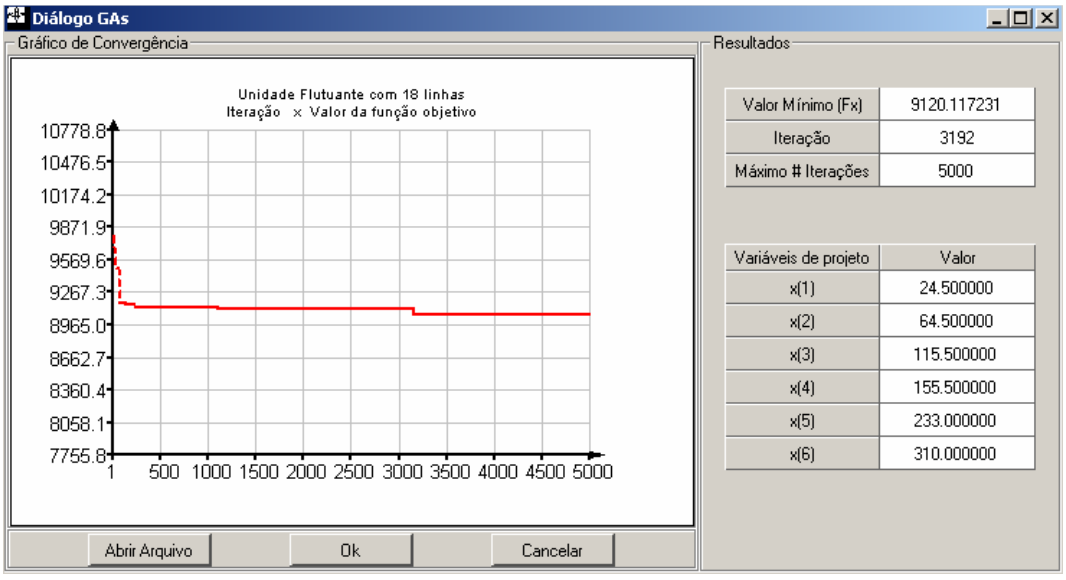


Figura 5.5 – Gráfico de convergência do processo de otimização (Exemplo 2).

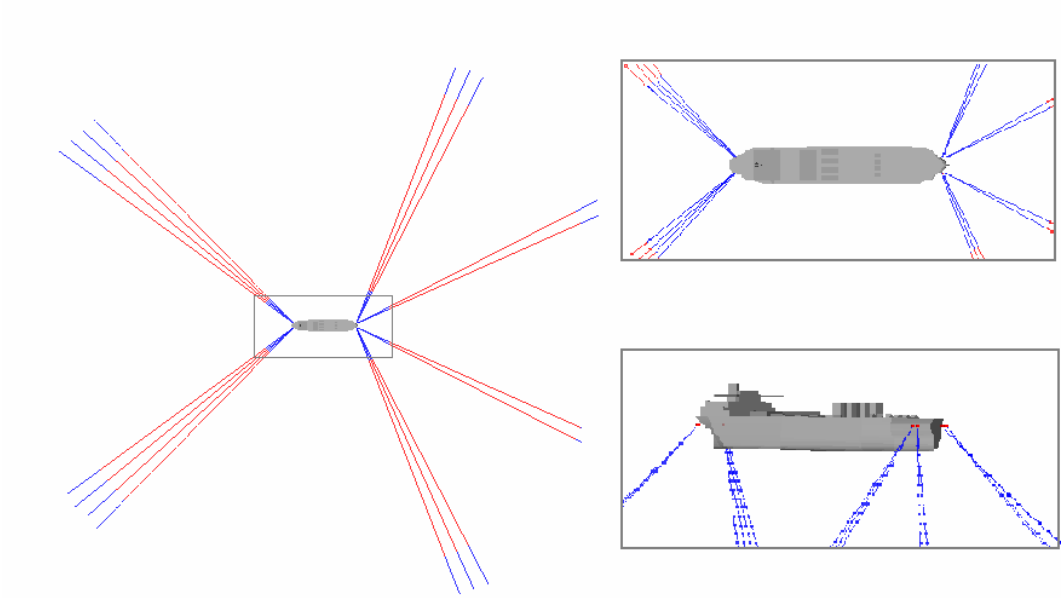


Figura 5.6 – Disposição final das linhas de (Exemplo 2).

6
Conclusões

Este trabalho apresentou uma metodologia para a minimização dos deslocamentos sofridos por unidades flutuantes durante as operações de exploração de petróleo. Para otimizar a disposição das linhas de ancoragem das unidades flutuantes, quando submetidas a diferentes combinações de forças externas resultantes das ações ambientais, foi desenvolvido um algoritmo genético, cujas variáveis de projeto consideradas foram os azimutes das linhas.

O algoritmo genético desenvolvido é uma versão do SSGA (*Steady-State Genetic Algorithm*). Sua principal característica é a substituição de apenas dois indivíduos por geração. Este baixo percentual de substituição diminuiu consideravelmente o número de avaliações da função objetivo durante cada ciclo geracional. Os operadores genéticos implementados foram o *crossover* de um (IX) e dois pontos (2X), a seleção pela técnica *ranking* e a mutação. Para comprovar a eficiência do algoritmo, foram realizados diferentes testes utilizando funções e problemas de otimização de estruturas, encontrados na literatura técnica. O desempenho do algoritmo foi satisfatório, considerando que os resultados obtidos foram equivalentes ou superiores aos apresentados nas diferentes publicações, como foi mostrado no capitulo 4.

Finalmente, o algoritmo proposto foi aplicado a problemas de otimização da disposição de linhas de ancoragem em sistemas reais utilizados pela Petrobras. Tais problemas consistem basicamente na otimização da disposição das linhas de ancoragem em dois tipos de unidades flutuantes: uma plataforma semi-submersível, no primeiro exemplo, e um FPSO, no segundo. Em ambos os casos, as unidades foram submetidas a diferentes combinações de condições ambientais (ventos, ondas e correntes), para as quais se procurou uma disposição das linhas de ancoragem, que garantisse o mínimo deslocamento sofrido pelas unidades. Os resultados obtidos neste trabalho foram compatíveis com os utilizados pela Petrobras.

6.1.
Sugestões para trabalhos futuros

Para o aprimoramento do presente trabalho propõe-se uma reformulação do problema na qual, além dos azimutes, sejam incluídas outras variáveis para a otimização dos sistemas de ancoragem. Por exemplo, os modelos podem ser otimizados levando-se em consideração o raio de ancoragem, a seção transversal, o material utilizado, o comprimento das linhas e a tensão de trabalho.

Uma análise dinâmica seria a mais indicada para realizar o estudo do comportamento dos sistemas de ancoragem. No entanto, isto representaria um elevado custo computacional. Uma boa alternativa seria efetuar uma análise dinâmica apenas nas melhores soluções obtidas ao longo do processo de otimização do problema.

Outras considerações para trabalhos futuros:

- Utilizar um fator que controle a aplicação do operador *crossover* sobre os indivíduos da população. Isto é, o processo se iniciaria com um valor de probabilidade de cruzamento (Pc) baixo que seria ajustado ao longo da evolução do processo;
- Implementar uma roleta de operadores que selecione os operadores genéticos, mutação e *crossover*, com diferentes taxas ao longo do processo de convergência, tal como é feito na referência [32]. A taxa indicará a probabilidade com que esses operadores serão aplicados, podendo-se evitar assim possíveis problemas de estagnação no processo evolutivo;
- Experimentar outras técnicas de otimização como, por exemplo, o Micro AG, apresentado no Capítulo 3 e utilizado pela referência [1], que utiliza uma população relativamente pequena. Isto reduziria consideravelmente o número de avaliações da função objetivo em cada ciclo geracional.

Referências Bibliográficas

1. ALBRECHT, C. H. **Algoritmos Evolutivos Aplicados à Síntese e Otimização de Sistemas de Ancoragem**. Tese de Doutorado, Universidade Federal do Rio de Janeiro/ COPPE, Rio de Janeiro, 2005.
2. MAFFRA, S. A..; PACHECO, M. A.; MENEZES, I. F. **Genetic Algorithm Optimization for Mooring Systems**. 5th IEEE, International Conference on Intelligence Engineering Systems, Helsinki, Stockholm, 2001.
3. SANTOS, C. M. **Otimização de um Sistema de Ancoragem de um FPSO**. Exame de qualificação de Doutorado, Universidade Federal de Rio de Janeiro/ COPPE, Rio de Janeiro, 2000.
4. LOPES, S. R. **Proposta de pesquisa - FAPERJ**. DEC / PUC-Rio, Janeiro de 2005.
5. MASETTI, I. Q.; JACOB, B. P. **Ancoragem com Complacência Diferenciada** – Relatório Técnico RL - MC 35 / CENPES – PETROBRAS, Rio de Janeiro, 2002.
6. OLIVIERI, B. P. **Otimização do Projeto de Pontes Protendidas Pré-moldadas pelo Método dos Algoritmos Genéticos**. Dissertação de Mestrado, Universidade Federal do Rio de Janeiro/ COPPE, Rio de Janeiro, 2004.
7. SARAMAGO, F. P. **Métodos de Otimização Randômica: Algoritmos Genéticos e Simulated Annealling** / Sezimária F. Pereira Saramago, Sociedade Brasileira de Matemática Aplicada e Computacional.
8. DE CASTRO, R. E. **Otimização de estruturas com Multi-objetivos Via Algoritmos Genéticos de Pareto**. Tese de Doutorado, Universidade Federal de Rio de Janeiro/ COPPE, Rio de Janeiro, 2001.
9. Universidade Estadual de Maringá – Departamento de Informática. www.din.uem.br/~nmsantos/OFuturoDaIA
10. ILAB **Sistemas especialistas:** www.ilab.com.br/tecnologia/.htm
11. Mundo dos Fractais: www.educ.fc.ul.pt/icm/icm99/icm14/index/
12. MANDELBROT, B. A. **The fractal geometry of nature**. W. H. Freeman, New York, 1983.
13. **Lógica Fuzzy**: www.rodrigodomingues2.hpg.ig.com.br/fuzzy.htm

14. WASSERMAN, P. D. **Advanced Methods in Neural Computing**. Van Nostrand Reinhold, New York (1993). http://citeseer.ist.psu.edu/ zhou02application. html
15. **The Domain of Genetic Search**. www.kneehighs.com/intro.html
16. PACHECO, M. A. **Algoritmos Genéticos: Princípios e Aplicações**, ICA, Laboratório de Inteligência Computacional/ Departamento de Engenharia Elétrica/Puc-Rio, Rio de Janeiro. http://www.ica.ele.puc-rio.br/cursos/ download/CE-Apostila-Comp-Evol.pdf
17. BARBOSA, H. J. **Algoritmos Genéticos para Otimização em Engenharia: Uma introdução aos Algoritmos Genéticos**. 2ª Escola de verão em Computação Científica, LNCC, Rio de Janeiro, 1977.
18. SHYUE-JIAN W.; PEI-TSE C. **Steady-State Genetic Algorithms for Discrete Optimization of Trusses**. Computers and Structures, Vol. 56 No.6, p. 979 – 991, China, 1995.
19. DE JONG, K. A. **An analysis of the Behavior of a Class of Genetic Adaptive System.** Tese de Doutorado, University of Michigan, 1975.
20. SHAFFER, R. E. **Practical Guide to Genetic Algorithms**. Naval Research Laboratory, Chemistry Division. http://chemistry.nrl.navy.mil/6110/6111/
21. BORGES, F. P. **Otimização Via Algoritmo Genético do Processo Construtivo de Estruturas de Concreto Submetidos à retração restringida tendo em vista a fissuração nas primeiras Idades**. Dissertação de Mestrado, Universidade Federal do Rio de Janeiro/ COPPE, Rio de Janeiro, 2002.
22. EBATUR, F.; HASANÇEBI, O.; HAKAN, I. **Optimal design of Planar and Space Structures with genetic Algorithms**. Computers and Structures. Vol. 56 p. 225 – 233, 2000.
23. MITCHELL, M. **L. D. Davis**, **Handbook Of Genetics Algorithms**. Artificial Intelligence, 100, 325 – 330, 1998.
24. ANDRE, J., SIARRY, P.; DOGNON, T. **An improvement of the Standard Genetic Algorithm fighting premature convergence in continuous optimization**. Advances in Engineering Software, Vol. 2 p. 1-13, 2000.
25. DEL SAVIO, A. et al. **Genetic Algorithm Optimization of Semi–Rigid Steel Structures**. The Eighth International Conference on the Application of Artificial Intelligence to Civil, Structural and Environmental Engineering. Rome, Italy, 2005.
26. DA SILVA, L.F. et al. **Otimização de Ligações Estruturais em Aço Através de Algoritmos Genéticos**. CILAMCE 2000, Congresso Ibero

Latino Americano Sobre Métodos Computacionais para Engenharia, Rio de Janeiro, Dezembro de 2000.

27. BORGES, L. A. et al. **Evaluation of the Post-Limit Stiffness of Beam-to-Column Semi-Rigid Joints Using Genetic Algorithms.** Proceedings of the 9th International Conference on Civil and Structural Engineering Computing. Civil-Comp Press, Edinburgo, 2003, v. 1, p. 1-12.
28. RAMIRES, F. B. et al. **Genetic Algorithms Structural Optimisation of Beam-to-Column Semi-Rigid Joints.** Proceedings of the Seventh International Conference on Computational Structures Technology - CST2004. Stirling: Civil-Comp Press, Lisboa, 2004, v. 1, p. 1-13.
29. FONSECA, E.T. et al. **Estudo do Fenômeno de Cargas Concentradas Utilizando Algoritmos Genéticos**. CILAMCE 2000, Congresso Ibero Latino Americano Sobre Métodos Computacionais para Engenharia, Rio de Janeiro, Dezembro de 2000.
30. DA FONSECA, M.; DAS NEVES, F. A. **Otimização de Torres Metálicas Tubulares Usando Algoritmos Genéticos**. Relatório Técnico – Científico Final, Universidade Federal de Ouro Preto, Ouro Preto, 2004.
31. RAJEEV, S.; KRISHNAMOORTHY, C. S. **Discrete Optimization of Structures Using Genetics Algorithms**, 2002.
32. ICA – **Laboratório de Inteligência Computacional Aplicada**. Departamento de Engenharia Elétrica/ PUC–Rio. http://www.ica.ele.puc-rio.br/cursos/index.asp.
33. CARBONO, A., MARTHA, L. F., MENEZES, I. F. **Mooring Pattern Optimization Using Genetic Algorithms**. WCSMO6 – World Congress on Multidisciplinary and Structural Optimization, Rio de Janeiro, 2005.