OTIMIZAÇÃO DE SISTEMAS DE RISERS PARA EXPLOTAÇÃO DE PETRÓLEO OFFSHORE ATRAVÉS DE ALGORITMOS GENÉTICOS PARALELOS

Luciano Tardelli Vieira

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA CIVIL.

Aprovada por:

_____

Prof. Breno Pinheiro Jacob, D.Sc.

_____

Prof. Beatriz de Souza Leite Pires de Lima, D.Sc.

_____

Prof. Alexandre Gonçalves Evsukoff, Dr.

_____

Prof. Antônio Carlos Fernandes, Ph.D.

_____

Eng. Marcos André Duarte Martins, D.Sc.

_____

Prof. Hélio José Corrêa Barbosa, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
ABRIL DE 2008

VIEIRA, LUCIANO TARDELLI

Otimização de Sistemas de Risers para Explotação de Petróleo Offshore Através de Algoritmos Genéticos Paralelos [Rio de Janeiro] 2008.

VII, 96 p. 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia Civil, 2008)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Projeto de Risers

2. Algoritmo Genético

3. Computação Paralela

I. COPPE/UFRJ II. Título (série)

Dedico à minha esposa Elisangela,
aos meus pais José e Dulce, e ao
meu filho Miguel.

AGRADECIMENTOS

A Deus.

À minha esposa Elisangela pelo carinho, amizade e companheirismo.

Aos meus pais José e Dulce e a todos os outros familiares por todo apoio e fundamental base familiar durante toda minha vida.

Aos Professores Breno Pinheiro Jacob e Beatriz de S. L. P. Lima pelo apoio e incentivo durante minha trajetória acadêmica na Coppe até o doutorado.

Aos colegas de trabalho do LAMCSO (Laboratório de Métodos Computacionais em Sistemas Offshore) do Programa de Engenharia Civil da COPPE/UFRJ e do LAMVI (Laboratório de Métodos Visuais) do Departamento de Expressão Gráfica da EE/UFRJ, em especial ao Coordenador do LAMVI e amigo Carl Horst Albrecht por sua fundamental e prestimosa ajuda.

Aos amigos e colegas do Programa de Engenharia Civil da COPPE – UFRJ e do CENPES - PETROBRAS pelo apoio e incentivo ao longo destes anos de convivência.

A equipe do Bureau Veritas que nos últimos meses me recebeu tão atenciosamente e tem compreendido e apoiado minhas repetidas e tão necessárias ausências para conclusão deste trabalho.

A CAPES pelo apoio financeiro.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

OTIMIZAÇÃO DE SISTEMAS DE RISERS PARA EXPLOTAÇÃO DE PETRÓLEO OFFSHORE ATRAVÉS DE ALGORITMOS GENÉTICOS PARALELOS

Luciano Tardelli Vieira

Abril/2008

Orientadores: Breno Pinheiro Jacob
Beatriz de Souza Leite Pires de Lima

Programa: Engenharia Civil

O presente trabalho apresenta uma metodologia para síntese e otimização de sistemas de risers empregados na explotação de petróleo offshore, baseada em métodos de computação conhecidos atualmente como “soft computing”. A metodologia desenvolvida garante que os sistemas de risers abordados cumprem especificações de desempenho, segurança e economia, sem que se necessite adotar uma extensa metodologia de análises paramétricas e de sensibilidade na fase de estudo de viabilidade.

A metodologia baseia-se no uso de um Algoritmo Genético, inspirado na teoria evolutiva das espécies. Os algoritmos genéticos são conhecidos pela capacidade de sintetizar soluções ou otimizar parâmetros em sistemas que visam objetivos muito específicos, dentro de critérios preestabelecidos de segurança e desempenho muitas vezes conflitantes.

No esquema de otimização proposto, a implementação dos Algoritmos Genéticos é integrada com estratégias de computação paralela, de forma a mitigar o alto custo computacional das análises por Elementos Finitos necessárias à avaliação do comportamento de configurações de risers.

Foram analisados exemplos numéricos que são devidamente apresentados e comentados para melhor compreensão e verificação da metodologia proposta.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

OPTMIZATION OF RISERS SYSTEMS FOR OFFSHORE OIL EXPLOITATION BY PARALLEL GENETIC ALGORITHMS

Luciano Tardelli Vieira

April/2008

Advisor: Breno Pinheiro Jacob
Beatriz de Souza Leite Pires de Lima

Department: Civil Engineering

This work presents the development and implementation of a methodology for the synthesis and optimization of offshore oil exploitation risers system, based on computational paradigms currently referred as “soft computing”. This methodology provides a riser configuration that fits performance, safety and economic requirements without the need of extensive parametric sensitivity analyses.

The methodology employs Genetic Algorithms, inspired on the evolutionary theory of species. Genetic algorithms are known to be able of reaching optimal configurations for systems with very specific objective functions, complying with performance and safety criteria that sometimes are conflicting.

The choice by genetic algorithms was also motivated because the facilities of integration with parallel computation strategies. A parallelization scheme is applied in this work, in order to mitigate the high computational cost of the Finite Element structural analyses required to evaluate the fitness of the riser configurations.

Some numeric examples are presented and commented in order to evaluate the proposed methodology.

SUMÁRIO

1 Introdução

1.1 Contexto

A exploração de petróleo *offshore* tem avançado consideravelmente chegando a limites nunca alcançados. O pioneirismo brasileiro na exploração em águas profundas impulsiona a pesquisa neste sentido, pois, a cada novo desafio são requeridas novas ferramentas computacionais capazes de suprir as necessidades dos profissionais dedicados a esta área.

Em especial, o estudo e a aceitação de um sistema de *risers* requerem um exaustivo número de análises, e uma definição equivocada pode freqüentemente levar um componente do sistema a alcançar ou exceder limites de viabilidade técnica e econômica, mesmo depois de gastos tempo e recursos com centenas de análises anteriores.

Desta forma, para o projeto conceitual de sistemas de *risers* torna-se necessário reconhecer que a seleção de uma configuração com boa performance estrutural e baixo custo deve ser formalmente descrita e tratada como um problema de otimização e síntese. A busca de configurações eficientes através de estudos paramétricos tem se mostrado uma tarefa árdua, como por exemplo, os estudos descritos em [1,2,3], e nem sempre levam aos melhores resultados.

Na maioria das vezes as técnicas desenvolvidas para abordar problemas de otimização permanecem restritas ao meio acadêmico durante muito tempo. Além disso, o fato de não ser possível preestabelecer uma relação custo – benefício sobre o uso da otimização em projetos, tem levado projetistas a optarem por soluções de fácil esboço que são funcionais, mas de performance discutível. Contudo, constantes avanços na área computacional, e a popularização de algumas técnicas de otimização, somados ao crescente interesse por projetos que considerem aspectos de otimização além da funcionalidade justificam cada vez mais pesquisas e investimentos nesta área de conhecimento.

1.2 Motivação

O projeto de sistemas de risers sob o enfoque de otimização constitui-se em um problema com múltiplos objetivos e submetidos a múltiplas restrições [4,5,6]. Para esta classe de problemas os métodos tradicionais de otimização, baseados em programação matemática que requerem a definição de funções objetivo e suas derivadas [7], não têm obtido grande sucesso.

Para o tratamento de problemas com estas características as ferramentas de "*soft computing*" têm se mostrado bastante adequadas [8], especialmente as baseadas em algoritmos evolutivos, incluindo os Algoritmos Genéticos [9,10], que também podem ser implementadas em conjunto com outros paradigmas, tais como a Lógica Nebulosa ("*fuzzy*") [11,12,13], compondo esquemas híbridos.

Contudo, para dar credibilidade ao processo de síntese e otimização de sistemas de *risers*, as análises estruturais realizadas devem ser suficientemente detalhadas e precisas e ainda contemplar diversas condições de carregamento, de forma semelhante a espiral tradicional de projeto. Esta tarefa certamente resultaria em custo operacional proibitivo para uma varredura ampla das possibilidades de projeto, caso seja adotada uma metodologia de computação seqüencial para as análises.

O surgimento de novas arquiteturas computacionais integrando vários processadores, caracterizando máquinas de alto desempenho, com grande velocidade de processamento e aplicações computacionais que tem o objetivo de estabelecer técnicas que possibilitem a distribuição de grandes tarefas de processamento em tarefas menores, realizadas simultaneamente compõe o que se conhece hoje como computação paralela, que surge então como uma alternativa para se contornar o problema de tempo de processamento proibitivo.

1.3 Objetivo

O objetivo do presente estudo é o desenvolvimento de uma metodologia para aplicação de paradigmas de otimização e computação paralela na fase de estudo de viabilidade de sistemas de *risers*. Propõe-se para este fim a utilização de algoritmos genéticos, reconhecidos como uma ferramenta de robustez elevada, capaz de solucionar problemas de síntese e otimização de alto grau de complexidade, e ainda possibilitarem a compensação de um eventual custo computacional extenso, com a utilização computação paralela.

Como a utilização de máquinas vetoriais é bastante restrita no universo das aplicações de engenharia, por caracterizarem uma opção de arquitetura paralela de custo monetário bastante elevado, é adotada como premissa neste trabalho a utilização de clusters de computadores para o desenvolvimento da metodologia.

1.4 Organização dos Capítulos

O segundo capítulo traz definições e conceitos básicos dos algoritmos genéticos, prosseguindo a seguir com uma abordagem mais aprofundada das suas principais características, tais como: codificação, população inicial, operadores genéticos de recombinação e mutação, entre outras.

No capítulo seguinte é apresentada uma breve discussão sobre as principais formas de paralelização de um algoritmo genético típico, sem a pretensão de abordar toda extensão do assunto ou detalhar exaustivamente os passos para a realização desta atividade.

Uma apresentação dos conceitos básicos relacionados à programação em um ambiente paralelo é realizada de forma resumida no capítulo quatro.

A seguir, o quinto capítulo resume os principais critérios de projeto correntes na indústria de projetos de sistemas de *risers*, que serão utilizados a seguir na modelagem numérica do problema físico.

A apresentação formal do problema físico abordado e a representação numérica do mesmo sob o ponto de vista da síntese e otimização com base nos conceitos de implementação de um algoritmo genético são realizadas no capítulo seis.

No sétimo capítulo é apresentada de forma detalhada uma ferramenta computacional criada com base na metodologia de síntese e otimização desenvolvida, e os resultados gerados por esta ferramenta são sumarizados no oitavo capítulo.

As considerações finais e conclusões são apresentadas no capítulo nove, juntamente com as recomendações para trabalhos futuros.

O trabalho é então encerrado com a apresentação das referências bibliográficas utilizadas na confecção do mesmo e durante a pesquisa para seu desenvolvimento.

2 Sistemas de Risers

2.1 Tipos e Configrações

Quanto ao tipo, os risers podem ser ***rígidos*** ou ***flexíveis***. Os risers rígidos, exemplificados pela figura seguinte são formados por sequências de tubos de aço de aproximadamente 12 m, geralmente unidos por solda de topo. Podem ser utilizados em atividades tanto de perfuração quanto em atividades de produção e intervenção, e podem estar envolvidos por flutuadores para diminuir o seu peso, quando em lâminas d'água profundas.

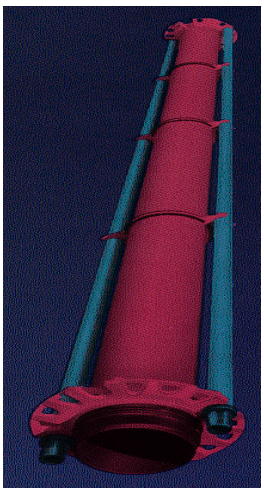


Figura 1: Riser Rígido

Os *risers* flexíveis, mostrados na Figura 1, são mangotes especiais compostos por várias camadas de diferentes materiais, e uma armadura central espiralada.

Figura 2: Riser flexível

Quanto à configuração asssumida pela linha, existem várias possibilidades e serem exploradas, dentre elas:

- A Catenária Livre - Instalação mais simples, maiores esforços na conexão com a plataforma, pode ser utilizada tanto em risers rígidos como em flexíveis. A figura a seguir exemplifica uma configuração em catenária livre.

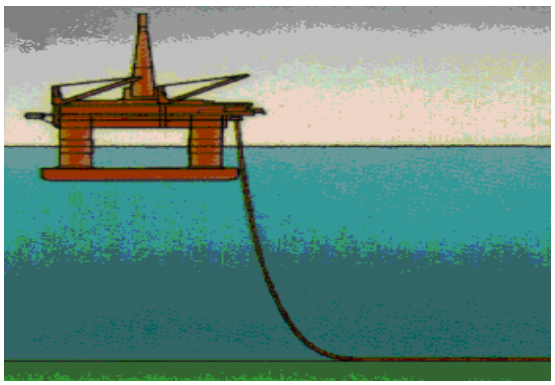


Figura 3 : Catenária Livre

- Lazy S / Steep S - Apresentam uma seção intermediária que passa por um arco com flutuadores, como mostra as figuras seguintes, cujo empuxo, alivia o peso suportado pelo sistema flutuante, e contribui com o momento restaurador quando sob solicitações laterais. Na configuração Lazy, há um tensionador sustentando o arco flutuador. Já na configuração Steep o próprio riser tensiona o arco flutuador.

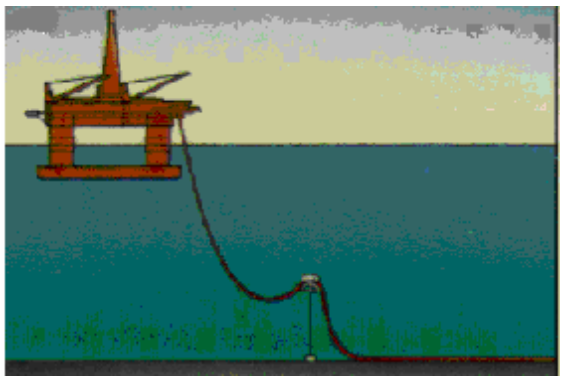


Figura 4 : Configuração Lazy S

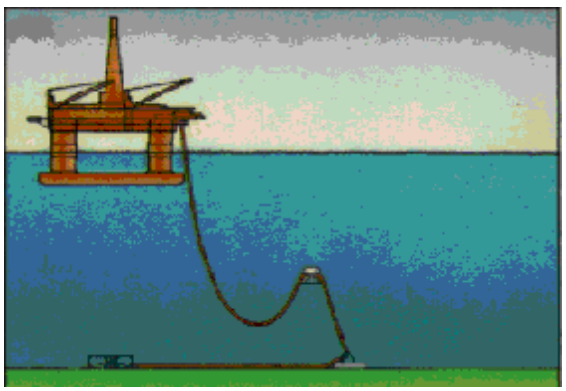


Figura 5 :Configuração Steep S

- Steep Wave / Lazy Wave– Estas configurações, mostradas nas figuras seguintes tem comportamento semelhante às configurações Lazy s / Steep s, mas o arco é substituído por uma seção intermediária com flutuadores distribuídos, simplificando a instalação.

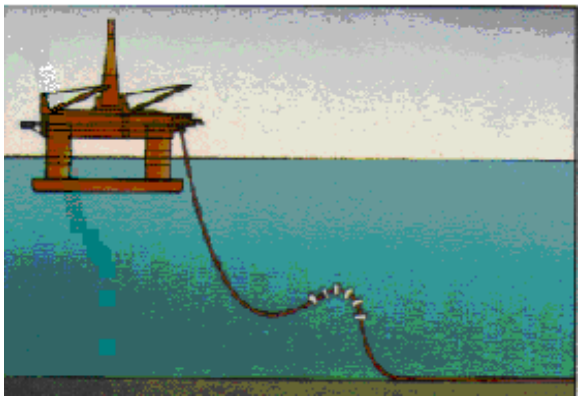


Figura 6 : Configuração Lazy Wave

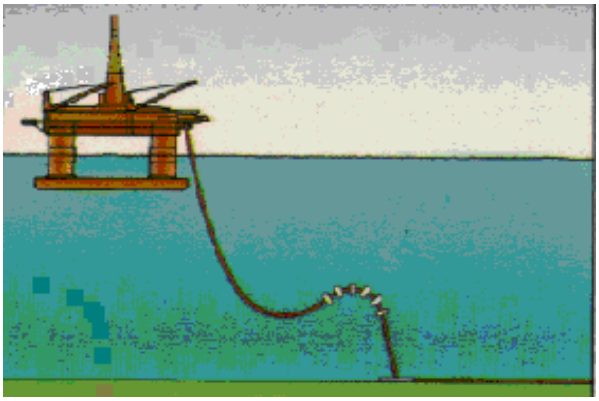


Figura 7 : Configuração Steep Wave

Outras configurações são possíveis, como a utilização de um sistema híbrido ou misto, utilizando combinações de riser rígido e flexível, baseadas em bóias submersas, como mostram as figuras seguintes.

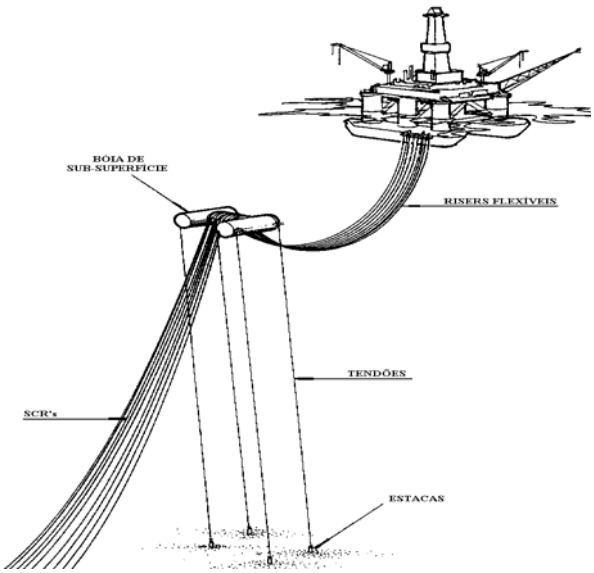


Figura 8 : Sistema híbrido de Risers (TLR)

Alguns estudos sobre a utilização de *risers* flexíveis ligando as unidades flutuantes diretamente ao fundo do mar foram realizados, mas tem se notado que esta solução por muitas vezes pode alcançar ou exceder seus limites de viabilidade técnica e econômica. Desta forma vêm se buscando alternativas ao uso de *risers* flexíveis em situações que exijam linhas de grande diâmetro, particularmente quando associadas às altas temperaturas e pressões internas, com o objetivo de: contornar a limitação do diâmetro imposta pelo processo de fabricação de um *riser* flexível, reduzir o custo dos *risers*, e também aumentar a capacidade de resistência às severas condições de serviço impostas aos *risers*.

Uma alternativa é a utilização de *risers* rígidos em catenária (*SCRs*) ligados à unidade flutuante. Os *risers* rígidos em catenária podem constituir uma alternativa mais econômica à opção convencional de linha flexível.

Nos sistemas baseados em navios, caracterizados por grande *offsets* estáticos e movimentos verticais maiores, foram observados problemas no comportamento estrutural dinâmico quando se empregam configurações em catenária livre. Os principais problemas estão relacionados com a flexão excessiva e compressão dinâmica dos *risers* junto ao *touch-down point*, devidos à raios de curvatura muito pequenos, e as tensões no topo dos *risers* geradas pelo *offset* e pelos movimentos angulares. Uma alternativa promissora é a adoção de sistemas com flutuadores de modo a contornar os problemas observados no comportamento estrutural dinâmico de *risers* em catenária simples que conectam o fundo diretamente à unidade flutuante. Porém um outro problema ainda deve ser bem estudado como, por exemplo, a ocorrência de grandes danos por fadiga causados por Vibrações Induzidas por Vórtice(VIV).

2.2 Critérios de Projetos de Risers

Risers rígidos são aqueles constituídos de material metálico, como aço ou titânio, e são objeto de estudo da API 2RD[14], onde é possível perceber que a definição completa de critérios práticos que levem a um projeto seguro é um assunto bastante complexo que remete a distintas áreas da engenharia estrutural.

Atualmente é bastante difícil conseguir considerar os vários efeitos que devem ser apreciados durante o projeto de um sistema de risers em uma única ferramenta de análise, visto que alguns efeitos necessitam de observação sob uma ótica global e outros devem ser estudados segundo premissas de análise local da região estrutural. Ainda é possível considerar que o atendimento de alguns critérios seja pré-requisito a análise de outro critério. Por exemplo, não seria admitido que se inicie um conjunto de análises de fadiga para uma certa configuração que não atende aos critérios de tensão combinada nem mesmo para as condições operacionais de serviço.

2.2.1 Considerações de Carregamento

O conjunto de carregamentos atuantes possíveis para um riser é dividido em três subgrupos: carregamentos funcionais, carregamentos ambientais e carregamentos acidentais.

Carregamentos funcionais são aqueles que são conseqüência da existência do sistema e não levam em consideração os efeitos ambientais ou acidentais. Carregamentos ambientais são aqueles impostos direta ou indiretamente pelo ambiente oceânico. Carregamentos acidentais são aqueles que resultam de ocorrências não planejadas.

A tabela seguinte, reproduzida da API-2RD [14] mostra as possíveis ocorrências destes carregamentos.

Tabela 1: Carregamentos em Risers

Risers Loads		
Functional	Environmental	Accidental
Weight of riser	Waves	Small dropped object
Weight of tubing contents and annulus fluid	Current	Normal handling impacts
Internal pressure	Vessel motions	Tensioner failure
External hydrostatic pressure	Seismic	Partial loss of station keeping capability
Nominal top tension	Ice	Flow-induced impact between risers
Loads caused by internal fluid flow, surges, slugs or pigs		
Buoyancy		
Soil interaction (catenary risers)		
Inertia		
Internally run tools		
Thermal		
Installation		
Vessel constrains		

O projeto completo de um riser deve considerar um conjunto mínimo de casos de carregamento que possibilite a correta avaliação da resposta estrutural do sistema, para comparação com os limites admissíveis para as seguintes condições:

- Tensão máxima;
- Deflexão e curvatura máximas;
- Fadiga;
- Colapso hidrostático;
- Limites de carregamento máximo para componentes específicos.

Para isso, casos de carregamento de projeto são definidos combinando carregamentos funcionais, ambientais e acidentais conforme a figura a seguir:

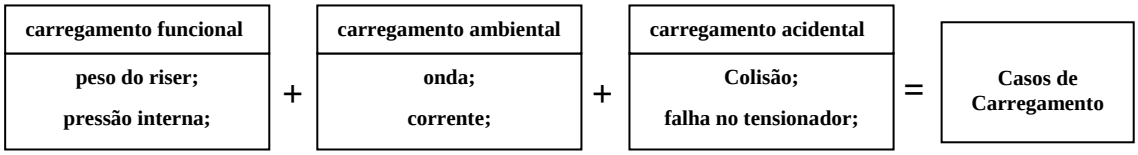


Figura 9: Exemplo de formação da matriz de casos de análise

Condições ambientais características são definidas de forma a cobrir as principais situações de avaliação da resposta de um sistema de risers ao carregamento ambiental, numa determinada locação, devendo incluir informação sobre ocorrência e persistência de condições.

A tabela seguinte, mostra as recomendações para as condições ambientais que devem ser consideradas num projeto de sistema de risers.

Tabela 2: Recomendações da API 2RD para Condições Ambientais

Environmental Conditions				
Environmental conditions	Description	Wave	Wind	Current profile
Extreme	Extreme wave Extreme wind Extreme current	100 years Hs/Tp Varied Associated Associated	Associated 100 years Associated	Associated Associated 100 years
Maximum Operating	Restricted conditions	See 4.3.1 *	See 4.3.1 *	See 4.3.1 *
Temporary	Instalation/Retrievel/Transportation	Seasonal	Associated	Associated
Fatigue	Fatigue conditions Wave VIV	Wave Scatter Diagram	Associated	Associated Annual Distribution
Survival	Survival condition	See 4.3.6 *	Associated	Associated
(*) Atender recomendações do citado item da API 2RD.				

São as combinações das condições ambientais citadas anteriormente com situações de carregamento funcional e acidental que geram os casos de carregamento que devem dar minimamente a noção do comportamento do sistema de risers ao longo de sua vida em utilização. A API 2RD[14] recomenda os nove casos de carregamento mostrados na tabela abaixo, extraída de referida norma.

Tabela 3: Recomendações da API 2RD para Casos de Carregamento em Projeto de Riser Rigido

Design Matrix for Rigid Risers					
Design Case	Load Category	Environmental Condition	Pressure	Reduced Tensioner Capacity or One Mooring Line Broken	Cf[a,b]
1	Operating	Maximum Operating	Design	No	1.0
2	Extreme	Extreme	Design	No	1.2
3	Extreme	Maximum Operating	Extreme	No	1.2
4	Extreme	Maximum Operating	Design	Yes	1.2
5	Temporary	Temporary	Associated	No	1.2
6	Test[d]	Maximum Operating	Test[d]	No	1.35
7	Survival	Survival	Associated	No	1.5
8	Survival	Extreme	Associated	Yes	1.5
9	Fatigue	Fatigue	Operating	No	Note[c]

Notes:
Anisotropic Material may require special considerations
a) Use of Cf is described in section 5.
b) Pipeline codes may require lower Cf for risers that are part of pipeline.
c) Not applicable.
d) Plant testing for rigid risers should be agreed between user and manufacturer.

2.2.2 Métodos de Análise

Um problema de análise global de risers passa pela definição das características físicas e geométricas do sistema estrutural, bem como as cargas a que o mesmo está sujeito, além dos movimentos impostos pelo corpo flutuante ao sistema de risers. O objetivo das análises é o de monitorar os esforços e as deformações impostas ao sistema de risers pelos carregamentos citados.

Para atingir este objetivo, na grande maioria dos casos não é possível a utilização de soluções analíticas com um grau aceitável de precisão. Desta forma o modelo de análise global mais aceito tradicionalmente é aquele que gera a solução através do MEF (Método dos Elementos Finitos).

Devido às características tanto do ambiente, quanto do próprio sistema, as análises de riser devem considerar um comportamento não linear acentuado. Devem ser consideras não linearidades geométricas, provenientes da magnitude dos deslocamentos; não linearidades físicas devido às relações constitutivas não lineares dos materiais; das condições de contorno e cargas não conservativas; entre outras.

Ainda levando em conta as características do sistema e do carregamento ao qual ele se submete, as análises devem contemplar efeitos Estáticos e/ou Dinâmicos, da seguinte forma:

- Ações permanentes: peso próprio, empuxo, correnteza e parcela estática do movimento da unidade flutuante imposto no topo do riser – ANÁLISE ESTÁTICA.
- Ações variáveis: onda e parcelas dinâmicas do movimento da unidade flutuante (freqüência de onda e deriva lenta);
  - Ações que variam "lentamente no tempo" – ANÁLISE ESTÁTICA;
  - Ações variando com período próximo ao período natural da estrutura – ANÁLISE DINÂMICA.

Tais análises podem ser feitas por técnicas de discretização no domínio do tempo ou no domínio da freqüência. As análises no domínio do tempo são bastante dispendiosas computacionalmente, mas podem tratar com rigor as não linearidades típicas do comportamento de sistemas de risers, enquanto as análises no domínio da freqüência são menos dispendiosas, mas tratam as não linearidades envolvidas no problema através de técnicas de linearização.

Além disso, dever ser destinada boa atenção aos problemas de convergência que são inerentes ao tratamento numérico do problema. Percebe-se então que, para que se tenham resultados consistentes e uma metodologia robusta para as análises propostas neste trabalho, torna-se necessária a utilização de técnicas que certamente irão resultar em custo computacional elevado.

2.2.3 Modos de Falha e Análises de Tensão e Deformação

Um estudo completo sobre a segurança de um projeto de risers rígidos deve contemplar vários mecanismos de falha e definir critérios para a prevenção dos mesmos. Comumente definem-se como possíveis critérios de falha para risers rígidos o nível de submissão a ação de tensões combinadas, deflexões excessivas, colapso hidrostático, efeitos de temperatura extrema, fadiga, flambagem e desgaste.

O problema de avaliação das soluções estruturais proposto se resume submeter uma configuração geométrica do tipo Lazy-wave a um conjunto de carregamentos característico. Para o presente trabalho são considerados na busca da configuração ótima apenas os critérios de projeto baseados em análises globais, admitindo que algumas características dos risers já haveriam sido previamente fixadas com base em análise de colapso hidrostático, condições de temperatura extrema e outras condicionantes locais. Considera-se que esta seria uma escolha acertada para criar critérios de avaliação para otimização pois, é a partir de resultados de análises globais que as demais características da configuração são geralmente definidas, além de os resultados destas análises serem condicionantes para as análises de fadiga.

Considerando que serão adotadas análises globais nas avaliações feitas durante a busca de uma solução otimizada, definem-se no estudo dois critérios de projeto a serem atendidos:

- um critério de tensões admissíveis
- um critério de deflexões máximas admissíveis.

Tipicamente as tensões em materiais metálicos são avaliadas em termos de tensões principais, ou melhor, sob a avaliação de um critério de tensões combinadas. O critério comumente adotado e recomendado pela API 2RD é o critério de Tensões de Von Mises, que deve ser verificado em cada seção crítica do riser.

A tensão equivalente de Von Mises (σ_e) pode ser definida da seguinte forma:

$$\sigma_e = \frac{1}{\sqrt{2}}\sqrt{(\sigma_1 - \sigma_2)^2 + (\sigma_2 - \sigma_3)^2 + (\sigma_3 - \sigma_1)^2} \qquad \textit{(eq. 1)}$$

Onde σ_1, σ_2, σ_3 são as tensões principais.

As componentes de tensão principais podem ser classificadas, segundo a citada norma, como um dos seguintes casos.

Primária	Qualquer tensão normal ou cortante que é necessária ao equilíbrio estático das forças impostas e momentos. Uma tensão primária não é autolimitante. Assim, se uma tensão primária excede substancialmente a tensão e escoamento, ocorrerá escoamento ou colapso estruturais totais.	
	Membrana (σ_p)	É o valor médio ao longo da espessura de uma seção sólida, excluindo efeitos de descontinuidades e concentração de tensões.
	Flexão (σ_b)	É a parcela da tensão primária proporcional à distância ao centróide da seção transversal, excluindo efeitos de descontinuidades e concentração de tensões.
Secundária (σ_q)	é qualquer tensão normal ou de cisalhamento que se desenvolve como resultado de restrição material. Este tipo de tensão é autolimitante, ou seja, um escoamento local pode aliviar as condições que causam a tensão, e uma única aplicação de carga não vai causar colapso.	

Assim, dependendo das tensões primárias usadas no cálculo, a tensão equivalente de Von Mises calculada deve então ser menor que um valor de tensão admissível, definido pelo lado direito de uma das seguintes inequações:

$$(\sigma_p)_e < C_f \cdot \sigma_a \qquad \textit{(eq. 2)}$$

$$(\sigma_p + \sigma_b)_e < 1.5 \cdot C_f \cdot \sigma_a \qquad \textit{(eq. 3)}$$

$$(\sigma_p + \sigma_b + \sigma_q)_e < 3.0 \cdot C_f \cdot \sigma_a \qquad \textit{(eq. 4)}$$

onde:

$\sigma_a = Ca \cdot \sigma_y$ - tensão combinada admissível básica;

Ca – fator de tensão admissível, igual a $2/3$;

C_f – fator de caso de projeto, definido na tabela 3;

σ_y – tensão de escoamento do material.

Há também a necessidade de exercer controle sobre as deflexões dos risers, de forma a prevenir em sistemas com múltiplos risers, a interferência entre eles ou com outros elementos do sistema. Além disso, o sistema de risers pode incluir tensionadores, *flex joints*, ou outros equipamentos acoplados às linhas, que serão limitantes das deflexõe. As exigências de torque ou rotação nestes componentes devem ser tomadas, então, como os valores máximos admissíveis para os resultados das análises de riser para os casos de projeto, multiplicados por fatores de segurança relacionados a cada caso. No presente trabalho, optou-se pelo monitoramento das deflexões angulares, conforme será abordado adiante.

2.3 O Problema físico e sua representação numérica

Uma configuração de riser em Lazy-wave pode ser definida como a composição de uma dupla catenária, com um trecho intermediário dotado de flutuadores distribuídos cujo empuxo alivia o peso suportado pelo sistema flutuante, e contribui com o momento restaurador quando sob solicitações laterais, como se nota na figura abaixo.

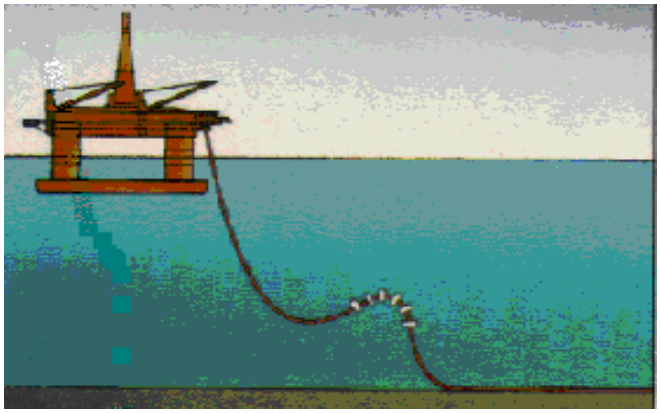


Figura 10: Ilustração de sistema de riser Lazy-wave

Desta forma é possível representar globalmente uma configuração lazy-wave, apenas com sete parâmetros, além é claro dos materiais e propriedades da seção do riser e flutuadores, como se observa na figura seguinte.

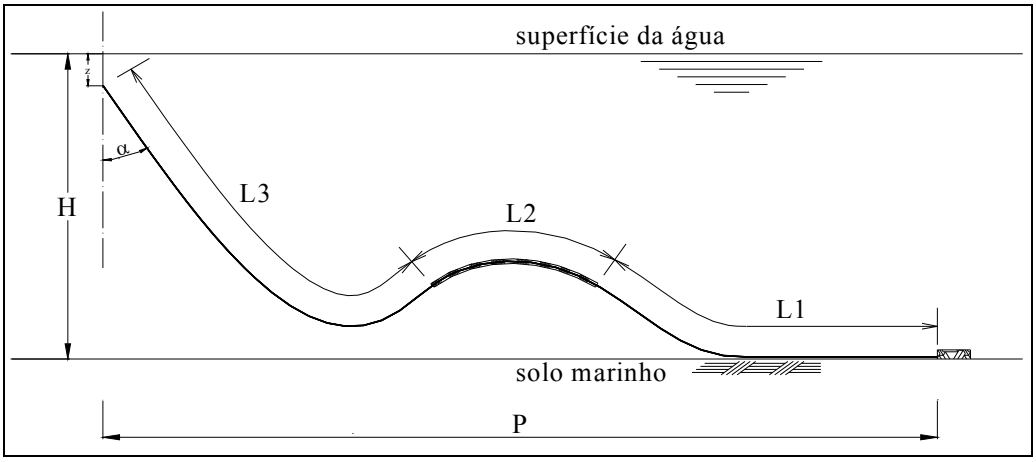


Figura 11: Definição de uma configuração Lazy-wave

Sendo:

- L1: comprimento final, entre o final do trecho com flutuadores e a conexão de fundo;
- L2: comprimento do trecho com flutuadores;
- L3: comprimento entre a conexão de superfície e o início do trecho com flutuadores;
- α: ângulo de topo da configuração;
- H: lâmina d'água;
- z: profundidade da conexão;
- P: projeção horizontal.

A lâmina d'água (H), bem como as condições ambientais caracterizam o cenário para o qual se pretende achar a configuração otimizada. A profundidade da conexão depende do calado da unidade flutuante que estiver conectada à linha. Logo, é evidente que, mudadas as condições ambientais ou a lâmina d'água, bem como a profundidade da conexão (z), um novo processo de otimização deve ser iniciado. Desta forma é conveniente que estes parâmetros permaneçam fixos ao longo do processo de otimização.

A projeção horizontal (P) e o ângulo de topo (α) podem ser facilmente definidos pela equação de catenária, ao se definirem os três comprimentos característicos. Fixada a projeção horizontal pode-se definir o ângulo de topo e vice-versa. Assim, estes parâmetros permanecem fixos ao longo do processo de busca.

Como esclarecido em [3], as características geométricas da seção transversal dos risers, bem como as características do material que o compõe, são definidas apropriadamente obedecendo a critérios de colapso hidrostático e propagante. Desta forma, consideram-se constantes a espessura e os diâmetros interno e externo da seção circular, bem como o módulo de elasticidade e a tensão de escoamento do aço que a compõe. Estes valores serão informados adiante, para cada aplicação realizada.

As propriedades dos flutuadores são geralmente definidas com base nas características definidas para o riser especificado, contudo admitem certa variação de acordo com o resultado que se objetiva. Desta forma, é possível incluir ou não no processo de busca e otimização as variáveis que definem basicamente como mostra a figura a seguir.

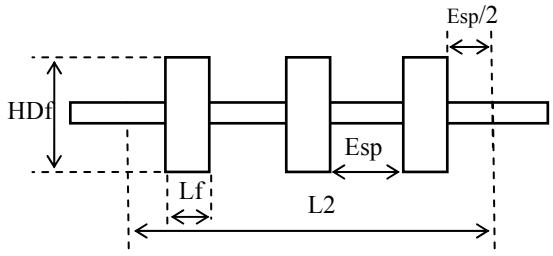


Figura 12: Definição das características dos flutuadores

Onde HDf é o diâmetro externo do flutuador, Lf é o seu comprimento e Esp é o espaçamento entre flutuadores.

É possível observar então, que os parâmetros mais importantes a serem incluídos no processo de síntese e otimização do problema físico abordado aqui são: o comprimento inicial do topo (L3), o comprimento do trecho com flutuadores (L2), o comprimento dos flutuadores até a conexão de fundo (L1), o diâmetro do flutuados (HDf), o comprimento dos flutuadores (Lf) e o espaçamento entre os flutuadores (Esp).

3 Algoritmos Genéticos

3.1 Introdução

Este capítulo faz uma apresentação aprofundada dos algoritmos genéticos, iniciando com definições e conceitos básicos, e prosseguindo com uma abordagem das principais características desta poderosa ferramenta de otimização, que é um algoritmo genético, tais como: codificação, população inicial, operadores genéticos de recombinação e mutação, entre outras.

Os ***Algoritmos Genéticos*** pertencem a uma classe de algoritmos conhecidos como algoritmos evolucionistas ou evolucionários, que são inspirados nos conhecidos princípios de sobrevivência e evolução das espécies.

Na natureza, os seres vivos são continuamente submetidos a transformações, que proporcionam a melhora da capacidade de sobrevivência no meio onde vivem. Contudo estas transformações são de natureza extremamente complexa, pois nem sempre grandes mudanças são eficazes, e pequenas nuanças aleatórias podem acarretar significante progresso. O processo de seleção natural é responsável pela reprodução mais provável de indivíduos melhor adaptados ao meio, logo, há maior difusão do material genético destes.

Foi norteado por estes princípios básicos da sobrevivência dos mais aptos, que no início da década de 70, o pesquisador John Holland desenvolveu a técnica implementada computacionalmente dos Algoritmos Genéticos, com o objetivo de resolver problemas de natureza complexa. São algoritmos probabilísticos, considerados robustos pela capacidade de resolverem problemas complexos em diversas áreas, como finanças, medicina e engenharia.

Para a perfeita compreensão do método serão apresentadas algumas das terminologias encontradas fartamente na literatura, como se segue:

- **Cromossomo**: é uma estrutura de armazenamento de dados que carrega informações sobre as variáveis do problema (geralmente um vetor ou uma cadeia de bits), ou seja, representa apenas uma solução no espaço de busca;

- **Gen ou Gene:** é um parâmetro codificado dentro de um cromossomo, isto é, a parte que representa uma única variável dentro de um vetor de dados;

- **Alelo:** representa uma das alternativas que um gen pode carregar, um valor que pode ser assumido por uma certa variável;

- **Genótipo:** representa toda a informação contida no cromossomo;

- **Fenótipo:** é a estrutura construída a partir do genótipo;

- **Operações Genéticas:** são operações que o algoritmo realiza sobre cada cromossomo;

- **Seleção:** é a operação genética que tem por objetivo escolher os indivíduos mais aptos, que servirão como pais no processo de reprodução;

- **Recombinação (*Crossover*):** é o operador genético responsável pela geração de novos cromossomos a partir da combinação aleatória dos genes de outros cromossomos;

- **Mutação:** é o operador genético, aplicado depois dos processos de seleção e *crossover*, responsável pela diversidade entre os indivíduos, através de modificações na estrutura genética de alguns poucos cromossomos;

- **Função de Avaliação:** representa uma medida da capacidade de sobrevivência de um cromossomo no processo de evolução, e consequentemente a probabilidade dele se reproduzir com mais freqüência. Pode ser uma composição da função objetivo com funções de penalidade estabelecidas a partir das restrições do problema;

- **Indivíduo**: é um simples membro da população, formado por um cromossomo e sua aptidão;

- **População:** é um conjunto de indivíduos;

- **Geração:** é um ciclo de uma evolução do algoritmo, ou seja, é uma iteração que o algoritmo genético executa;

Em um algoritmo genético, geralmente, uma população inicial é gerada de maneira aleatória. Desta forma não se induz a uma preferência de busca em uma região particular do espaço de busca. Através da avaliação do valor da função aptidão de cada cromossomo é realizada o processo de seleção, que indica quais os indivíduos tem condições de sobreviver e a probabilidade de se tornarem genitores, sendo assim submetidos aos operadores de recombinação e mutação para formarem uma nova geração.

Consecutivamente as etapas de seleção e reprodução são repetidas, levando o algoritmo a regiões mais promissoras do espaço de busca, onde se encontra a solução ótima, até que o critério de parada do algoritmo seja atendido.

A forma mais tradicional de codificação de uma solução é através de um código binário (de 0's e1's) armazenado num vetor relacionado com as variáveis do problema. Além desta, também e possível a codificação através de números reais. Um algoritmo genético, genérico e simplificado, é apresentado por Lemonge[**Erro! Indicador não definido.**] da seguinte forma:

Algoritmo Genético Genérico

Inicialize a população

Avalie indivíduos na população

Repita

Selecione indivíduos para reprodução

Aplique operadores de recombinação e mutação

Avalie indivíduos na população

Selecione indivíduos para sobreviver

Até critério de parada satisfeito

Fim

A maior diferença entre os algoritmos genéticos e os métodos clássicos talvez seja o fato de os algoritmos genéticos trabalharem com populações de soluções e não necessitarem de informações relativas a derivadas Além disso, algumas características são atrativos a mais para esta técnica de otimização:

- Geralmente não apresentam dificuldades em identificar soluções globais, mesmo para problemas com muitos ótimos locais;
- Não requerem funções objetivo contínuas ou diferenciáveis;
- Podem otimizar funções com muitas variáveis e se adaptam bem a problemas com variáveis contínuas e discretas, ou uma combinação destas;
- Em geral, não requerem formulações ou reformulações complexas para os problemas;
- Podem ser combinados com métodos clássicos específicos para alguns problemas (hibridação);
- Não requerem estudos preliminares para determinação de soluções iniciais no espaço de busca;
- A função objetivo pode ser facilmente modificada sem a necessidade de recodificações extensas;
- Realizam buscas simultâneas em várias regiões do espaço de busca;
- São flexíveis para trabalhar com restrições arbitrárias e otimizar múltiplas funções com objetivos conflitantes;
- São tolerantes a ruídos e dados incompletos, além de poder trabalhar com dados gerados experimentalmente;
- Fornecem uma lista de parâmetros ótimos e não uma simples solução.

3.2 Codificação de um algoritmo genético

A codificação de uma solução representada no algoritmo genético pode ser feita tradicionalmente de dois modos, usando uma representação binária ou uma representação por números reais.

Em um problema com codificação binária, sendo $x = (x_1,x_2,...,x_n)$ onde $x_i \in R$, uma solução possível para um problema qualquer, a representação de x (cromossomo) é feita com a justaposição da codificação binária de cada elemento x_i. Por exemplo, em um problema com três incógnitas, x_1, x_2 e x_3, onde uma codificação possível para estas variáveis seja respectivamente 01001, 01010 e 11010, um dos cromossomos do problema seria representado por: x = 010010101011010.

Com este tipo de codificação, para que se consiga recuperar o valor original de cada variável, é necessário um procedimento de decodificação. O procedimento de decodificação é diferenciado de acordo com o tipo das variáveis.

Para uma variável discreta, a decodificação fornece um índice que localiza o valor da variável numa lista de referência, que representa o espaço de busca desta variável.

Desta forma, se uma variável discreta fosse representada por x_i = 0011, o seu índice (IND) de referência no espaço de busca seria o seguinte:

$$\mathbf{IND = 0\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 = 3} \qquad \textbf{(eq. 5)}$$

Vale lembrar que para um espaço de busca discreto, o número de valores possíveis para uma variável (nv*)*, ou seja, o número de posições contidas na lista de referência desta variável é dado por $nv = 2^{nb}$, onde ***nb*** é o número de bits usado na representação da variável.

Para uma variável contínua, a decodificação fornece um valor real da seguinte forma:

$$x = x^i + IND \times \varepsilon \qquad \textbf{(eq. 6)}$$

$$\varepsilon = \frac{x^s - x^i}{2^{nb} - 1} \qquad \textbf{(eq. 7)}$$

Onde x^i e x^s são, respectivamente, os limites, inferior e superior do espaço de busca, e ε é a precisão da solução.

Consequentemente, o número de bits necessários para garantir a precisão ε, é:

$$nb = \log_2 \frac{x^s - x^i}{\varepsilon} \qquad \textbf{(eq. 8)}$$

Pode-se então notar que cadeias mais longas proporcionam maior precisão numérica à solução. No entanto, cadeias muito longas fazem o algoritmo convergir vagarosamente, sendo recomendado o uso do mínimo necessário de casas decimais nas soluções.

Um problema da codificação binária é que dois pontos vizinhos na representação de um espaço de busca de uma variável real nem sempre os torna também vizinhos na codificação. Por exemplo, para passar do valor real 7, 0111 em codificação binária, para o valor real 8, 1000 em codificação binária, seria necessária a troca de todos os bits da cadeia. Esta dificuldade pode ser superada através da utilização do código de Gray, que também usa uma representação com zeros e uns, contudo, torna pontos vizinhos no espaço de busca também vizinhos na codificação.

A representação real é mais naturalmente compreendida pelo ser humano, pois não necessita de codificações e decodificações, ou seja, um cromossomo é um conjunto solução possível ao problema. Além disso, a codificação real é mais eficiente caso se esteja trabalhando com variáveis contínuas e necessita-se de uma precisão grande, onde a representação binária resultaria em cromossomos extremamente longos, que requerem muito espaço de memória.

Os operadores de recombinação convencionalmente usados para codificações binárias não têm um bom desempenho para codificações reais, pois basicamente eles apenas trocam informações entre os parâmetros, e assim não criam nenhuma informação nova. É, então, recomendada a utilização de operadores aritméticos ou baseados em direção, quando se usa codificação real. Tais operadores são abordados mais adiante.

3.3 População inicial

Após a escolha do sistema de codificação, deve-se então definir de maneira adequada ao problema, como será explorado inicialmente o espaço de busca. Existem diversas maneiras de se gerar uma população inicial para um algoritmo genético, mas geralmente isto se faz de maneira aleatória, através de funções "pseudo-randômicas" introduzidas nas rotinas computacionais.

Gerando uma população inicial maior que as seguintes é possível garantir que o espaço de busca será mais bem explorado. Uma alternativa para melhorar a exploração do espaço de busca, em se tratando de codificação binária, seria gerar uma metade da população inicial aleatoriamente, e a outra metade a partir da primeira, invertendo-se os bits desta. Assim, garante-se que cada posição na cadeia de bits tem ao menos um representante na população com os valores zero e um.

Caso se tenha conhecimento de alguma solução anterior do problema, esta pode ser introduzida na população inicial, garantindo que a solução encontrada nunca será pior que a existente. Esta técnica é conhecida como "*seeding*".

3.4 Função Avaliação

A função de avaliação de um problema de otimização é responsável por retornar uma medida da capacidade de um cromossomo (solução candidata) ser selecionado tanto para sobreviver para uma geração futura quanto para se tornar genitor. Assim, um processo de seleção é capaz de classificar as soluções geradas de acordo com sua avaliação.

A função avaliação de um problema de otimização pode ser representada através da soma de sua função objetivo com funções de penalização [15]. Cada função de penalização está diretamente ligada a uma restrição do problema, e somente é ativada se uma destas restrições for violada. No caso de um problema sem restrições, a função avaliação é coincidente com a função objetivo do problema.

Desta forma, problemas de otimização com restrições podem ser transformados em problemas sem restrições, através do uso de funções de penalização relacionadas com as restrições, associadas à função objetivo, formando uma única função a ser otimizada.

Este tipo de tratamento do problema atua sobre a violação de uma restrição, e segundo a literatura corrente [8,10], é o mais indicado para problemas onde a região factível é larga. De forma genérica, a função de avaliação de um problema com restrições pode ser representada da seguinte forma:

$$F(x) = f(x) + penal(x) \quad \textbf{(eq. 9)}$$

Geralmente, a parcela "penal(x)" é representada por um somatório de funções de penalização diferentes, e f(x) é a função objetivo do problema.

Existem diversos métodos para se introduzir funções de penalidade num problema de otimização com restrições, dentre os quais se destacam o método das penalidades estáticas, o das penalidades dinâmicas, e o das penalidades adaptativas, que podem ser encontradas em [23, 24].

Talvez a maior dificuldade em problema de otimização com restrições seja a ponderação da função de penalização. Isto porque, para penalizações muito pequenas é possível que o algoritmo convirja para uma solução não factível, e por outro lado se as penalizações forem muito grandes, é possível que haja uma convergência prematura para uma solução factível, porém não ótima.

Em alguns problemas, a função avaliação pode ser tão complexa, que demande um grande esforço computacional em suas avaliações. Para minimizar este problema é ideal evitar que cromossomos idênticos sejam avaliados mais de uma vez, reaproveitando assim a avaliação já feita.

Ainda com o intuito de minimizar o esforço computacional destinado as avaliações de aptidão, é possível usar métodos de avaliação diferentes nas primeiras gerações, onde existem muitas soluções não factíveis, substituído estes posteriormente por uma função de avaliação mais completa, quando o algoritmo já tiver encontrado regiões mais promissoras do espaço de busca. Outras estratégias de avaliação são discutidas nas referências [25 ,26].

3.4.1 Mapeamento da Função de Avaliação

A classificação em função das aptidões das soluções geradas pode ser feita em ordem decrescente ou crescente, respectivamente, caso o problema seja de minimização ou maximização.

Caso o algoritmo seja implementado como referência a um problema de maximização, há a necessidade da transformação de um problema de minimização em um problema de maximização onde a função de avaliação de soluções cresça a medida que o função objetivo decresça.

A idéia mais simples seria inverter a função objetivo e somar ao denominador do resultado as funções de penalidade, todas de mesmo sinal que a função objetivo. Contudo isso geraria um problema quanto a definição das magnitudes das penalidades, pois seria ideal que as parcelas do denominador fossem da mesma ordem de grandeza.

Surge então, a necessidade de mapear a função de avaliação, de forma a mantê-la variando entre valores conhecidos. O primeiro passo para tal seria tornar o valor da função objetivo adimensional, da seguinte forma:

$$f = \frac{f(x_1, x_2 ..., x_n)}{f(x_1^{\lim}, x_2^{\lim} ..., x_n^{\lim})} \qquad \textbf{(eq. 10)}$$

onde: f – função objetivo adimensionalizada;

x^{max} – limites no espaços de busca para cada variável, que levam ao valor máximo de f.

Desta forma uma função de avaliação para um problema de minimização, usando um algoritmo de maximização poderia ser escrita de forma genérica como se segue:

$$fitness = \frac{1}{f + \sum P_j} \qquad \textbf{(eq. 11)}$$

Onde P_j são os valores das funções de penalidade impostos por cada um dos j critérios de restrição impostos ao problema, e tratados na seção seguinte.

3.5 Seleção

Depois de determinado o tipo de codificação que mais se ajusta ao problema, definida a função aptidão com suas respectivas penalidades, e gerada a população inicial, o algoritmo pode então entrar na fase de seleção. A seleção é responsável por escolher os indivíduos que estão mais aptos a se tornarem genitores, ou seja, aqueles com maiores valores de aptidão.

Existem diversas formas de se proceder no processo de seleção [**Erro! Indicador não definido.**], contudo, aqui são explanadas com detalhes apenas as três técnicas mais utilizadas: ***seleção por rank***, ***Seleção proporcional a aptidão***, e ***seleção por torneio.***

Na técnica de seleção por rank, os indivíduos da população são classificados e ordenados em função de uma nota, ou seja, uma situação num rank (lista ordenada de acordo com a aptidão). Ao cromossomo de maior aptidão, é associado um número inteiro e ao de menor aptidão é associado o número zero, o resto da população recebe a sua posição interpolando-se entre estes valores extremos. Por exemplo, se o número inteiro associado ao cromossomo de maior aptidão for dois, a interpolação seria feita da seguinte forma:

$$fi = 2\frac{N-i}{N-1} \qquad \textbf{(eq. 12)}$$

Onde N é o tamanho da população, i é o rank, de acordo com a aptidão, e f_i é a situação do indivíduo no rank.

Os indivíduos mais bem colocados no rank têm maior chance de serem selecionados para uma população intermediária de genitores, conhecida como ***mating pool***. Este tipo de seleção não conduz a nenhuma conotação com a biologia.

Outra técnica bastante usada é a seleção por torneio, onde n indivíduos são selecionados aleatoriamente na população, e na comparação entre estes, o melhor indivíduo é copiado para a população intermediária. O processo é então repetido até que a população intermediária seja completada. Geralmente faz-se n igual a dois ou três.

A seleção por torneio pode também ser feita com probabilidades desiguais de escolha dos indivíduos que serão comparados. Desta maneira, para uma seleção por torneio com n candidatos, o primeiro candidato é vencedor com a probabilidade q, o segundo com probabilidade q(1-q), o terceiro $q(1-q)^2$, e assim por diante.

Uma outra técnica, um pouco mais complexa, e muito encontrada na literatura, é a de seleção proporcional à aptidão. Esta técnica é baseada numa aptidão acumulada para indivíduos seqüenciados aleatoriamente.

O processo de seleção proporcional à aptidão pode ser descrito da seguinte forma: Primeiro calcula-se a aptidão acumulada dos indivíduos, que estão listados aleatoriamente; depois, é escolhido um número aleatório entre zero e a soma de todas as aptidões; o indivíduo escolhido, então, é aquele que tem a aptidão acumulada imediatamente superior aquele número anteriormente escolhido. Estes passos são repetidos até que a população intermediária, de genitores, esteja completa.

Desta maneira, cada cromossomo S_i , com aptidão f_i, tem a probabilidade p_i, de ser selecionado, seguinte forma:

$$p_i = \frac{fi}{\sum_{i=1}^{N} fi} \qquad \textbf{(eq. 13)}$$

Onde N é o tamanho da população

Este procedimento é conhecido como “Roda da Roleta”, e é necessário ressaltar que ele não funciona se puderem ocorrer valores negativos para a aptidão de alguns indivíduos.

Após selecionados, os indivíduos armazenados na geração intermediária são submetidos ao processo de reprodução, para então formarem uma nova geração. Se uma geração é totalmente substituída por outra mais jovem, corre-se o risco de se perder um indivíduo de boa qualidade. Para que isto não ocorra, um percentual da população que tenha um desempenho exemplar pode ser mantida na geração seguinte, diretamente copiada, esta prática é denominada ***elitismo***.

3.6 Reprodução e operadores genéticos

Na fase de reprodução, os indivíduos selecionados são submetidos a um operador de recombinação, conhecido na língua inglesa como ***crossover***, e em seguida a um operador de ***mutação***, formando assim novos indivíduos que substituirão a geração anterior, total ou parcialmente.

Os operadores de *crossover* são, geralmente, aplicados aos pares de cromossomos retirados da população intermediária que são, então, submetidos à troca de material genético (parte da cadeia de bits, no caso de codificação binária), formando dois novos cromossomos chamados de "filhos".

O *crossover* é aplicado a cada par de cromossomos selecionados com uma probabilidade determinada (entre 60% e 90%), chamada de ***taxa de crossover***, possibilitando que alguns indivíduos sejam preservados integralmente. Isso pode ser feito associando a cada par de cromossomos um número pseudo-aleatório entre 0 e 1, e fazendo com que o crossover somente ocorra com aqueles pares que tiverem o número associado menor que a taxa de crossover.

O operador de mutação tem o objetivo de melhorar a diversidade na população, impedindo problemas de convergência prematura, contudo ele destrói parte da informação contida no cromossomo, sendo assim, é recomendado o uso de uma ***taxa de mutação*** entre 0,1% e 5%. A mutação, no caso de codificação binária, inverte o valor de um dado bit do cromossomo filho, com uma dada probabilidade. Operadores de mutação e crossover, tanto para codificação binária quanto para codificação real serão vistos mais adiante.

Os dois procedimentos para substituição de cromossomos mais comuns na literatura são conhecidos como ***geracional*** e ***steady-state.***

A principal característica de um algoritmo geracional é a de substituir toda a população em cada geração. Contudo, a fim de evitar a perda de bons indivíduos, pode se adotar um critério de seleção elitista, onde os k melhores pais nunca são substituídos. Geralmente faz-se k=1 para evitar problemas de convergência prematura.

Uma outra variação da substituição geracional, que visa a manutenção dos melhores indivíduos é, gerar uma população de N filhos a partir N pais, e a os N melhores indivíduos da união de pais e filhos forma a população da geração seguinte.

Um algoritmo genético com substituição de cromossomos do tipo “steady-state” gera apenas um ou dois filhos por vez, e a cada criação uma avaliação é feita, então, se os indivíduos criados forem melhores que os piores da lista de classificação, eles sobrevivem e os piores são eliminados.

Uma variação deste procedimento pode ser criada, caso se considere que só se pode inserir um indivíduo na população se este possuir aptidão maior que a aptidão média da população. Alternativamente, os indivíduos criados podem substituir os indivíduos mais velhos da população, admitindo que um cromossomo muito antigo já transmitiu seus genes para a população.

Os seguintes algoritmos foram citados por Lemonge como exemplos básicos de algoritmos genéticos tipo Geracional e Steady-state:

ALGORITMO GENÉTICO GERACIONAL

Início

Inicialize a população P aleatoriamente

Avalie indivíduos da população P

Repita

Repita

Selecione 2 indivíduos em P

Aplique operador de recombinação com probabilidade p_c

Aplique operador de mutação com taxa p_m

Insira novos indivíduos em P’

Até população P’ completa

Avalie indivíduos na população P’

P←P’

Até critério de parada satisfeito

Fim

ALGORITMO GENÉTICO STEADY-STATE

Início

Inicialize a população P aleatoriamente

Avalie indivíduos da população P

Repita

Selecione indivíduos em P

Aplique operador de recombinação com probabilidade p_c

Aplique operador de mutação com taxa p_m

Avalie indivíduos gerados

Selecione indivíduo f para sobreviver

Se f é melhor que o pior indivíduo de P **Então**

Insira f em P, de acordo com seu Ranking

Até critério de parada satisfeito

Fim

3.6.1 Operadores para codificação binária

O operador mais simples para recombinação de cromossomos, em codificação binária, é o ***crossover de um ponto.*** Este operador corta os dois pais selecionados em um mesmo ponto da cadeia de bits, aleatoriamente, separando-os em duas partes, que podem ser chamadas de caudas e cabeças dos cromossomos. Então, as caudas são trocadas, formando dois novos cromossomos, o "filho 1" e o "filho 2". A operação é mais bem visualizada no exemplo abaixo.

Pai 1	1111| 111	Filho 1	1111000
Pai 2	0000| 000	Filho 2	0000111

Neste caso a posição escolhida aleatoriamente para o corte foi a quarta posição.

Da mesma forma, pode ser implementado um ***crossover de dois pontos***, que analogamente ao anterior, escolhe dois pontos aleatórios de corte, e em seguida troca o material genético entre estes pontos, formado também dois filhos, como é ilustrado a seguir:

Pai 1	1111| 111| 111	filho 1	1111000111
Pai 2	0000| 000| 000	filho 2	0000111000

Desta vez, as duas posições de corte escolhidas foram a quarta posição e a sétima posição.

Interpretando um cromossomo como um anel fechado pela junção de suas extremidades, pode-se entender o crossover de um ponto como um caso particular do crossover de dois pontos, onde um dos pontos está localizado exatamente na junção da cadeia de bits. Este é um motivo para se considerar o crossover de dois pontos melhor que o de um ponto.

A generalização natural destas formas de crossover seria o ***crossover de n-pontos***, contudo, a literatura mostra que o aumento excessivo de pontos não traz bons resultados.

Existe também o ***crossover uniforme***, onde para cada par de cromossomos genitores é escolhida uma máscara de bits aleatórios. A partir da escolha da máscara, podem ser notadas na literatura, duas formas distintas de se proceder.

Na primeira, um dos filhos é formado copiando para a sua cadeia de bits, todas as posições da cadeia de bits do Pai 1 que corresponderem ao valor 1 na máscara, e as posições da cadeia de bits do Pai 2 que corresponderem ao valor 0 na máscara. Fazendo o procedimento contrário, é então formado o segundo filho.

Pai 1	10100110110
Pai 2	11011011100
Máscara	10010110110
Filho 1	11001111110
Filho 2	10110010100

Alguns pesquisadores revelam que, em termos de desempenho, não há grandes diferenças entre a utilização de um crossover de n-pontos ou de um crossover uniforme, e isto é devido ao quão robusto é um algoritmo genético.

O ***operador de mutação*** é sempre utilizado após o crossover, e sua atuação é a de efetuar uma troca em um dos bits do cromossomo, de 0 para 1 ou de 1 para 0, com a probabilidade p_m de ordem 1/L, onde L é o tamanho do cromossomo. A mutação de um cromossomo é exemplificada abaixo:

Antes da mutação	1001| 0| 11010
Depois da mutação	1001| 1| 11010

3.6.2 Operadores para codificação real

Como já citado, no item relativo à codificação, os operadores utilizados em codificação binária não apresentam um bom resultado, se diretamente usados para algoritmos com codificação real. Os operadores de crossover, por exemplo, simplesmente trocariam as informações entre os genes, não gerando assim nenhuma solução nova.

Deste modo, diversos operadores distintos podem ser propostos para lidar com cromossomos codificados com números reais, dentre os quais os mencionados a seguir parecem ser os mais utilizados.

Para o melhor entendimento dos processos de recombinação utilizados em codificação real será aqui assumida a seguinte representação:

Cromossomos genitores mãe = $(m_1, m_2, ..., m_i)$

pai = (p1,p2,..., pi)

Cromossomos filhos $filho_i = (a_1, a_2, ..., a_i)$

O mais simples operador para codificação real é o ***crossover uniforme***, onde é gerado apenas um filho cujas componentes a_i são escolhidas aleatoriamente no intervalo $[m_i, p_i]$.

Os crossovers de n-pontos e uniformes da codificação binária podem ser utilizados com sucesso para codificação real, se for adotada uma modificação proposta em [22], onde cada um dos genes dos filhos é formado da seguinte maneira:

$$a_i = \lambda m_i + (1 - \lambda) p_i \quad \textbf{(eq. 14)}$$

Onde λ é um número aleatório escolhido na distribuição uniforme no intervalo [0,1], e o segundo filho é gerado trocando-se os coeficientes de m_i e p_i.

A proposta acima não admite valores de a_i fora do intervalo $[m_i, p_i]$, mas esta extrapolação pode ser conseguida com o uso do ***crossover BLX*** - α (Blender crossover).

No crossover do tipo BLX - α, os genes filhos são formados da seguinte maneia:

$$a_i = m_i + \lambda(p_i - m_i) \quad \textbf{(eq. 15)}$$

Onde desta vez, λ é escolhido da distribuição uniforme no intervalo $[-\alpha, 1+\alpha]$, e o outro filho pode ser gerado da mesma maneira que anteriormente.

Outros operadores que podem ser utilizados são aqueles ditos aritméticos, que se baseiam em conceitos de combinação linear de vetores. Os mais comuns são o ***crossover média*** e o crossover ***média geométrica***, e ambos geram apenas um filho.

Nos crossovers média e média geométrica, aos genes do filho gerado são definidos como mostrado abaixo:

$$\text{Média} \qquad a_i = (m_i + p_i)/2$$

$$\text{Média geométrica} \quad a_i = \sqrt{m_i p_i}$$

Muitos outros operadores de recombinação podem ser encontrados na literatura.

Entre os operadores de mutação o mais simples de ser utilizado, é a ***mutação aleatória ou randômica***, que a simples substituição de um gene (que tenha passado no teste de probabilidade) por um número escolhido aleatoriamente num intervalo definido pelo problema.

3.7 Critérios de Parada

Depois de várias gerações, onde as respectivas populações são submetidas seqüencialmente a processos de avaliação, seleção, reprodução e substituição, deve ser usado algum critério para decidir se o algoritmo já encontrou a melhor solução possível, ou verificar se caso a evolução continue será feito algum progresso significante ou não. Desta forma, todo algoritmo genético deve ter um critério de parada.

Alguns critérios de critérios de parada são rotineiramente apresentados na literatura, dentre os quais, alguns são mencionados a seguir.

1. Interromper o algoritmo assim que ele atinja um número predeterminado de gerações (este é o mais simples deles).

2. Se o valor ótimo é conhecido, a parada é determinada assim que o algoritmo atingir este valor, durante o processo de evolução (é difícil possuir o valor ótimo do problema, previamente).

3. Parar o processo quando não houver melhora significante no valor da aptidão do melhor cromossomo, ou da média das aptidões dos cromossomos, para algumas gerações consecutivas, ou seja, verificar se o algoritmo convergiu.

Para o caso de representação binária, pode se considerar que um algoritmo convergiu usando o seguinte critério: admitir que uma variável ou gene já convergiu quando 90% da população tem o mesmo valor para esta variável, e considerar que o algoritmo como um todo já encontrou a convergência quando 90 a 95% das variáveis já podem ser consideradas convergentes.

3.8 Algoritmos Genéticos Paralelos

3.8.1 Introdução

O avanço contínuo da capacidade de processamento de computadores pessoais vem possibilitando que a modelagem computacional aplicada à problemas de engenharia se torne cada vez mais complexa e mais sofisticada. Contudo, a inegável e crescente necessidade da aplicação de métodos de busca e otimização para esta classe de problemas tem sido suprida com sucesso pela utilização de algoritmos robustos, frequentemente probabilísticos e que dependem de centenas de repetições de análises com alto custo computacionais. Em alguns casos, mesmo dispondo de toda capacidade de processamento de um computador pessoal, tornasse inviável a aplicação destes métodos comprovadamente bem sucedidos.

Atualmente, a solução de problemas complexos de engenharia aponta para o surgimento de uma tendência comum de associação da tarefa computacionalmente onerosa de busca e otimização a utilização de sistemas de computação paralela. Logo existe uma preferência natural para a escolha de algoritmos de busca e otimização facilmente paralelizáveis, dentre os quais um dos mais utilizados tem sido os Algoritmos Genéticos Paralelos (AGP).

Este capítulo faz uma breve discussão sobre as principais formas de paralelização de um algoritmo genético típico, sem o detalhamento pormenorizado dos códigos e rotinas necessários a esta atividade.

Analisando a estrutura de algoritmo genético podemos ratificar está idéia observando que cada indivíduo da população tem uma qualidade (fitness) avaliada de forma independente de qualquer outro fator ou indivíduo da população, e ainda que as operações genéticas efetuada sobre os indivíduos independem da ordem de aplicação, ou ainda da ordem dos indivíduos que se submeterão a estas operações.

Os estudos sobre algoritmos genéticos paralelos [16, 17] vêm mostrando comumente que o sucesso de cada implementação não se relaciona apenas com o algoritmo escolhido, mas também com sua parametrização, o tipo de problema ao qual será aplicado e ainda com o tipo de equipamento e técnicas de paralelização que servirão de plataforma para a execução do algoritmo. Assim, poucas variações nos parâmetros de controle do algoritmo ou na formulação do problema, ou ainda na plataforma de execução podem viabilizar a utilização de algoritmos inicialmente com baixo desempenho.

3.8.2 Modelos de Algoritmos Genéticos Paralelos

A idéia básica atrás da maioria dos programas paralelos é dividir uma tarefa em partes e solucionar as partes simultaneamente usando múltiplos processadores [31]. Esta abordagem de "dividir-e-conquistar" pode ser aplicada em algoritmos genéticos de muitos modos diferentes, e a literatura contém muitos exemplos de implementações paralelas [17, 27, 28]. Alguns métodos de paralelização usam uma única população, enquanto outros dividem a população em várias subpopulações relativamente isoladas. Alguns métodos podem explorar arquitetura de computadores maciçamente paralelos, enquanto outros são melhores adaptados para multicomputadores com menos e elementos de processamento contudo de maior desempenho [29, 30].

De acordo com [**Erro! Indicador não definido.**, 17, 27, 28], existem três tipos (modelos) principais de Algoritmos Genéticos Paralelos:

Algoritmos genéticos paralelos podem ser implementados em redes de estações ou mesmo em um único computador de natureza paralela, dependendo do modelo de paralelização adotado. Alguns dos principais modelos de algoritmos genéticos paralelos são resumidamente discutidos a seguir.

O mais simples dos modelos de paralelização é conhecido como tipo "***mestre- escravos***" ou global. Este modelo de paralelização admite uma única população que é submetida a operadores de seleção, crossover e mutação em um processador principal (mestre), o qual em seguida subdivide esta população, enviando estas subdivisões para diversos processadores (escravos). Os processadores escravos ficam encarregados de proceder com a avaliação dos indivíduos (que demanda alto custo computacional) e retornar o resultado ao processador principal, para se possa continuar com as próximas gerações. O modelo de paralelização tipo mestre – escravo pode ser ilustrado pela figura seguinte.

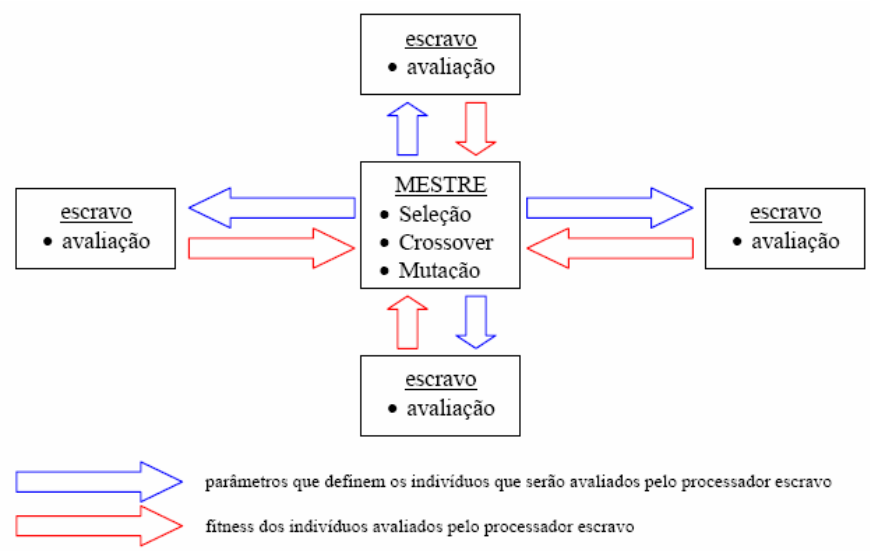


Figura 13: Modelo Mestre x Escravos

Outro modelo é conhecido como "***modelo de ilhas***" ou algoritmo genético com granularidade baixa. Este modelo admite a evolução isolada de subpopulações, delegando assim a diversos processadores, não só a função de avaliação de indivíduos, mas também as funções de reprodução e seleção destes.

As subpopulações ou ***demes*** (colônia de células), após um dado número de gerações, trocam entre si os melhores indivíduos antes de continuar a evolução isoladamente, é o que se conhece como ***migração***. Este processo de troca de indivíduos entre processadores vizinhos (interconectados) é repetido a cada número de gerações pré-determinado.

Segundo a literatura, o modelo de paralelização em ilhas é o mais utilizado, pelo motivo de sua implementação ser simples, diferenciando dos algoritmos genéticos seqüenciais apenas no fato de rodar cada algoritmo responsável por uma sub-população em um processador distinto, além da implementação de rotinas que coordenem o processo de migração. Este modelo de paralelização pode ser implementado até mesmo num simples computador convencional. Representativamente, o modelo de paralelização em ilhas é melhor compreendido com o auxílio da figura abaixo.

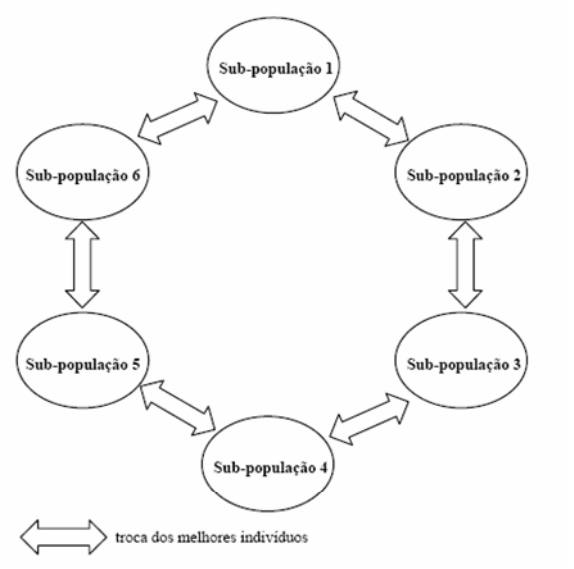


Figura 14: Modelo de ilhas

Por fim, vale lembrar os algoritmos genéticos paralelos de granularidade alta, ou granulação fina, que constituem um processo de paralelização bastante sofisticado. Neste modelo, cada indivíduo é posto num processador distinto, onde são avaliados, e os processadores geralmente ficam arranjados numa malha 2-D. Os processos de seleção e reprodução são realizados localmente, entre cada processador e os processadores vizinhos (conectados a ele). Este método pode ser exemplificado esquematicamente através da figura seguinte.

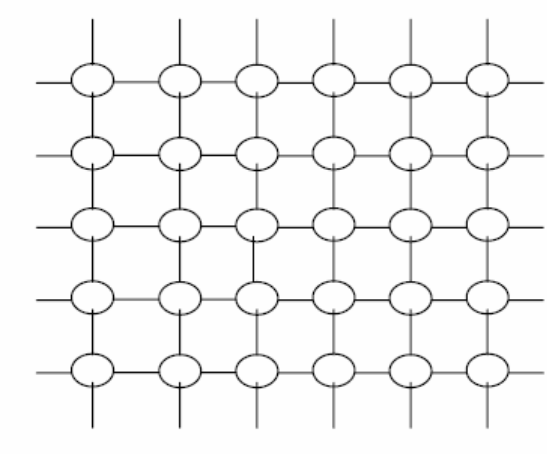


Figura 15: Modelo de granularidade alta

Pode ainda existir um algoritmo genético paralelo misto, onde se combinam esses métodos. Por exemplo, pode-se implementar um modelo de ilhas, onde em cada ilha rode um algoritmo genético com granulação fina, ou que cada ilha evolua usando um algoritmo do tipo "mestre-escravos".

4 Computação Paralela

4.1 Ambiente de Processamento Paralelo

Este capítulo tem como objetivo apresentar os conceitos básicos relacionados à programação em um ambiente paralelo apresentados de forma resumida, dando foco nas aplicações de Sistemas MPP e Redes de Estações de Trabalho.

4.1.1 Terminologia Básica

- Concorrência – característica de um programa que possibilita a execução de conjuntos de tarefas de forma concorrente, ou seja, traduz a capacidade de execução paralela de grupos de tarefas.
- Processos – tarefas executadas concorrentemente, geralmente gerenciadas por um programa mestre que trocam informações com este e entre si através de comunicação por troca de mensagens.
- Aceleração (speed-up) – relação entre o tempo de execução de um programa em um único processador (execução seqüencial) e o tempo de execução do mesmo programa utilizando vários processadores.
- Escalabilidade – capacidade de um sistema de execução paralela melhorar sua aceleração de maneira diretamente proporcional ao aumento do número de processadores utilizados.
- Sincronização – coordenação dos eventos de troca de mensagem durante a execução dos processos.
- Granularidade – medida da quantidade de processamento direto ocorrido durante uma execução de um programa em ambiente paralelo em relação à quantidade de troca de mensagens realizada entre os processos.
- Balanceamento de carga – distribuição equilibrada de processos entre os processadores utilizados, de forma a garantir a eficiência do sistema paralelo.
- Cluster de PC's – conjunto de máquinas independentes interligadas em rede e configuradas para executar paralelamente uma ou mais tarefas computacionais.

4.1.2 Classificação Básica de Paralelismo

Considerando o foco de observação no objeto paralelizado é possível descrever as seguintes formas básicas de paralelismo.

- Paralelismo de Dados:

Os processadores executam as mesmas instruções ou tarefas sobre conjuntos diferentes de dados. Enquadram-se nesta classificação os sistemas SMP (Symmetric Multi Processing), representados pela figura abaixo, e que possuem mais de um processador em um mesmo computador.

Os processadores funcionam independentemente, mas compartilham os recursos de memória e disco segundo uma política de controle de concorrência, gerenciada diretamente pelo sistema operacional. Desta forma, esta arquitetura é bem transparente ao usuário, ficando a cargo do sistema operacional a maior parte da complexidade. O acesso pelos processadores à memória é feito diretamente sem a necessidade de troca de mensagens, e somente um processador acessa um endereço da memória por vez. Como exemplo de aplicação deste tipo de paralelismo temos o sistema o Cray Y-MP.

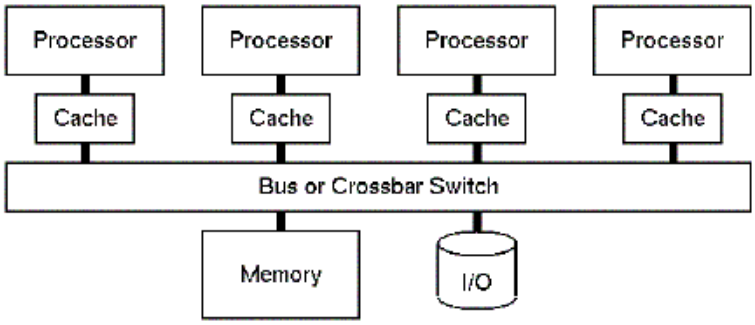


Figura 16: Arquitetura SMP.

- Paralelismo Funcional:

Os processadores executam diferentes instruções ou tarefas sobre conjuntos de dados distintos ou não. É uma boa opção para a paralelização de programas bem modulados onde os blocos de instrução são claramente definidos e podem ser considerados como processos diferentes. Podem ser caracterizados desta forma os sistemas MPP, exemplificados esquematicamente na figura seguinte, onde há pouco ou nenhum compartilhamento de recursos entre processadores, e o controle do paralelismo é realizado pelo programador, que deve coordenar as tarefas e a coerência entre os diversos processos.

Um exemplo típico para este tipo de aplicação de computação paralela são os Clusters de PCs, onde cada nó do sistema MPP é um computador independente, com memória e discos próprios. Os processadores estão conectados em rede e o acesso a cada maquina é feito por passagem de mensagem (Message Passing).

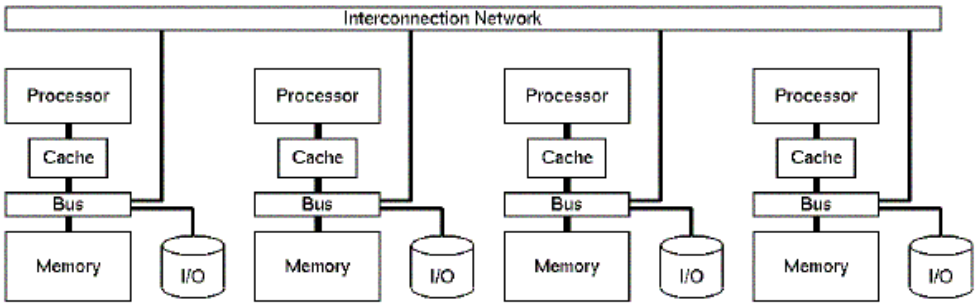


Figura 17:Arquitetura MPP.

O Message Passing é um método de comunicação baseada no envio e recebimento de mensagens através de uma rede de computadores, seguindo regras de protocolo de comunicação entre processadores com memória própria. Como exemplo de Message Passing é possível citar, conforme a referência [18]:

- PVM - Parallel Virtual Machine;
- MPI - Message Passing Interface;
- MPL - Message Passing Library.

Para montar um cluster é preciso basicamente ter vários computadores interligados via rede, uma plataforma para manipulação de processos paralelos (LAM, MPICH), um sistema operacional (Linux,Windows), uma linguagem de programação (Fortran, C++) e uma biblioteca de comunicação (MPI, PVM, MPL).

O baixo custo de aquisição e manutenção deste sistema, aliado ao aumento considerável de desempenho dos computadores pessoais (PCs), e a evolução de técnicas eficientes de conexão em rede, são os atrativos que vem popularizando a aplicação destas arquiteturas dentro da engenharia. O desenvolvimento e aperfeiçoamento de plataformas de manipulação de processos paralelos e das bibliotecas de comunicação também tem tido um papel fundamental no desenvolvimento destes sistemas.

4.2 Biblioteca de comunicação paralela MPI

O MPI (Message Passing Interface), um dos modelos de Message Passing mais empregado atualmente nas diversas áreas da computação paralela para ambiente de memória distribuída, foi introduzido pelo MPI Fórum em maio de 1994.

O MPI é um produto resultante de um fórum aberto de desenvolvimento constituído por 40 organizações de pesquisadores, empresas, usuários e vendedores que definiram a sintaxe, semântica e o conjunto de rotinas padronizadas. A bibliografia disponível sobre o MPI é bastante vasta e de fácil acesso, contudo as principais utilizadas neste trabalho são as referências [19] e [20]. A versão utilizada no presente trabalho é a "MPI2" e foi finalizada em julho de 1997.

Uma das principais características do MPI é a robustez em vários ambientes, já que foi desenvolvido para executar eficientemente em máquinas diferentes. Somente o funcionamento lógico das operações tem formato inalterável, ficando toda implementação a cargo do próprio desenvolvedor que usa as características de cada máquina para gerar um código mais personalizado.

O MPI consiste em um conjunto de bibliotecas ou funções que auxiliam na comunicação entre processos. Apesar do grande número de funções disponíveis, na prática, pode ser construído um programa simples e eficaz usando poucas delas. A Figura 18 relaciona as seis rotinas fundamentais do MPI que serão descritas em linhas gerais a seguir.

Figura 18: Principais rotinas MPI.

A primeira função (MPI_INIT) inicializa o MPI, ou seja, prepara o ambiente de programação para reconhecer todas as outras funções MPI. Desta forma será sempre a primeira função a ser solicitada na programação paralela usando MPI.

Em seguida, a função MPI_COMM_SIZE contabiliza o número de tarefas ou processos, definido na linha de comando de execução. Esses processos são associados a um comunicador e são capazes de comunicar apenas entre os processos pertencentes ao seu comunicador. Inicialmente, todos os processos são membros de um grupo com um comunicador já pré-estabelecido denominado MPI_COMM_WORLD. Os processos têm uma única identificação denominada de rank e atribuída pelo sistema quando o processo é inicializado.

A identificação única de cada processo é reconhecida com a utilização da função MPI_COMM_RANK. Após estas etapas pode-se realizar a troca de mensagens entre os processos associados ao comunicador utilizando as funções MPI_SEND e MPI_RECV coordenadamente. A interrupção final de todos os processos e respectiva finalização d o ambiente paralelo é determinada pela utilização da função MPI_FINALIZE.

Toda troca de mensagem em MPI possui um formato semelhante ao mostrado abaixo:

FUNÇÃO(endereço, contador, tipo de dado, destino ou origem, etiqueta, comunicador, erro)

Onde a função pode corresponder a um comando de envio ou recebimento de mensagem e sua utilização exige a informação dos seguintes parâmetros;

- ✓ endereço: localização da memória (buffer) onde está armazenada a mensagem a ser enviada ou recebida;
- ✓ contador: especifica o tamanho da mensagem a ser enviada ou recebida;
- ✓ tipo de dado: : especifica o tipo de dado a ser enviado ou recebido, normalmente, devem ser iguais nas chamadas de envio e recebimento, exceto, como por exemplo, quando o dado é definido do tipo MPI_PACKED.
- ✓ destino ou origem: identificação do rank do processo receptor ou emissor;
- ✓ etiqueta: identificação da mensagem;
- ✓ comunicador: define um contexto e grupo de comunicação;
- ✓ erro: código de erro, retorna 0 em caso de sucesso ou código de erro em caso de falha na comunicação.

5 Desenvolvimento do algoritmo

O código do algoritmo genético foi escrito em linguagem de programação orientada à objetos, criando além da solução numérica, também um ambiente gráfico amigável de pré e pós processamento dos dados do algoritmo genético. A interface relativa ao algoritmo genético está integrada a uma interface maior, destinada ao tratamento de dados para o programa de simulação numérica de linhas e unidades flutuantes com código em Fortran chamado PROSIM [21]. Este programa realiza as análises estáticas ou dinâmicas não lineares mencionadas anteriormente com a utilização do MEF no domínio do tempo, e considera todos os carregamentos antes mencionados.

O algoritmo genético implementado inicializa uma população randômica de indivíduos com cada um dos seus parâmetros sorteados entre os limites máximos e mínimos previamente definidos. O esquema de seleção usado é da "roda da roleta" e as gerações subseqüentes podem ser constituídas por um procedimento para substituição de cromossomos com ou sem considerações de elitismo, geracional ou ainda steady-state.

O algoritmo inclui apenas codificação binária para os indivíduos, mutação simples com a alteração de um bit aleatório, e crossover de único-ponto de corte. O critério de parada básico implementado leva em conta apenas um número máximo de gerações definido, e não pode ser desativado. Outros critérios de parada, baseados na convergência ou na estagnação do algoritmo foram implementados e serão comentados a seguir.

A interface gráfica que mostra em tempo real, durante todo o processo, uma representação da configuração geométrica do melhor indivíduo de cada geração e um gráfico de acompanhamento de convergência, será apresentada detalhadamente mais adiante.

5.1 Função de Avaliação

É sabido que a composição de custos de uma configuração de risers para exploração de petróleo offshore não é facilmente representada, pois a rigor deve contemplar várias fases como: transporte, lançamento e instalação, dentre outras. Contudo, de modo a simplificar a avaliação das soluções candidatas, a função objetivo utilizada neste trabalho contempla apenas o custo de aquisição dos componentes da linha. Acredita-se que a parcela considerada represente bastante adequadamente a diferença de custo entre cada trecho e, conforme será descrito a seguir, fica claro que não é realmente necessário conhecer o custo real de aquisição de cada trecho de linha e sim a relação entre eles.

A *função objetivo (F_{OBJ})* da otimização é representada por duas parcelas. A primeira é um somatório do produto do comprimento *(L_i)* de cada trecho de linha da configuração pelo seu relativo índice custo unitário *(Ic_i)*. A segunda parcela é o produto do volume dos flutuadores também pelo seu respectivo índice de custo unitário. Os índices de custo podem ser descritos como a relação entre o custo unitário de cada respectivo trecho de linha e o menor custo unitário entre os três trechos. Assim podemos escrever:

$$F_{OBJ} = \left(\sum_{i=1\ldots n} IC_i \times L_i \right) + \left(V_{flut} \times IC_{flut} \right) \qquad \textbf{(eq. 16)}$$

De acordo com o descrito anteriormente, para evitar problemas numéricos, a função objetivo foi utilizada de forma adimensional pelo seu máximo possível. Assim, pode escrever-se a função objetivo adimensional, da seguinte forma:

$$f = \frac{\left(\sum_{i=1\ldots n} IC_i \times L_i \right) + \left(V_{flut} \times IC_{flut} \right)}{\left(\sum_{i=1\ldots n} IC_i \times L^{\max}{}_i \right) + \left(V^{\max}{}_{flut} \times IC_{flut} \right)} \qquad \textbf{(eq. 17)}$$

A função de avaliação das soluções candidatas é então composta da função objetivo adimensional e um somatório de funções de penalidade relativas as restrições de comportamento aplicadas ao problema, definidas a seguir. Assim, a função de avaliação para um problema de minimização, usando um algoritmo de maximização, pode ser escrita na seguinte forma:

$$fitness = \frac{1}{f + \sum P_j} \qquad \textbf{(eq. 18)}$$

Onde Pj são as respectivas penalidades aplicadas.

Com base nas equações 10 e 18, podemos fazer as seguintes observações:

- O valor máximo de *f* será 1, quando a configuração analisada for constituída pelos comprimentos máximos dados como limite superior do espaço de busca;
- O valor mínimo de *f* será atingido quando a configuração analisada for constituída pelos comprimentos limites do espaço de busca;
- A função de *fitness* terá o valor máximo possível quando *f* for mínimo, e a solução analisada não for restringida.

5.2 Restrições

São impostas ao problema físico em questão, várias restrições ou condições limites de comportamento, contudo, criar um sistema computacional que avaliasse de forma completa e definitiva um sistema de risers seria uma tarefa bastante difícil de ser realizada tendo compromisso simultâneo com a qualidade dos resultados e com um custo computacional viável.

Assim, para simplificar o processo de avaliação das soluções candidatas, apenas um tipo de função de penalidade que pode ser aplicada a várias restrições conforme será descrito a seguir.

Forma da Função de Penalidade

Podemos observar através da equação (18) que o denominador da *fitness* é a soma de duas parcelas: uma relativa à função *f*, e outra relativas às penalidades.

Como o valor máximo de *f* é 1, caso as penalidades propostas forem muito maiores que este valor, o algoritmo pode ignorar o objetivo original de minimização passando a priorizar qualquer solução não restrita, mesmo que ele tenha o pior custo possível. Assim definiu-se o valor máximo unitário para cada uma das funções de penalidade, ou seja, a função de penalidade deve cortar o eixo das ordenadas em 1.

Tendo esta prioridade, uma curva composta foi implementada para ser utilizada na penalização de soluções que rompem os critérios de projeto, com um trecho de reta horizontal de ordenada nula e um trecho de curva cúbica, conforme mostrado a seguir.

$$P_q = \begin{cases} k \cdot \left[(1 - x^3)\right], & se \quad x < 1 \\ 0, & se \quad x \geq 1 \end{cases} \qquad \textit{(eq. 19)}$$

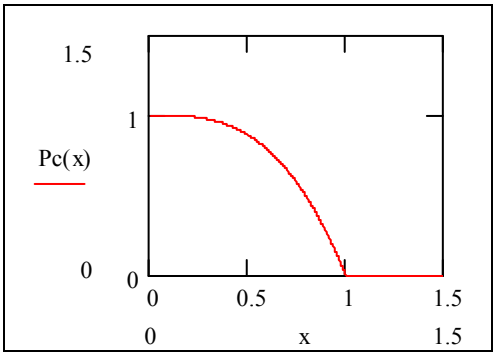


Figura 19: Função cúbica de penalidade

Em casos de configurações com limite de restrição muito rígido, ou seja, quando para valore x muito próximo de 1 pela esquerda deste valor, o valor de penalidade assumido tornasse muito pequeno, o algoritmo pode passar a ignorar as penalidades, dando como ótima uma solução restrita. O coeficiente k foi inserido para forçar o surgimento de soluçõesque não violem nenhum critério de projeto, podendo ser atualizados iterativamente ao longo da busca para que se imprima maior rigor na avaliação de novas soluções geradas.

No algoritmo proposto o valor de k é acrescido de 10% cada vez que o melhor solução apresentada ainda viola um dado critério de projeto

Restrições de comportamento

A função de penalidade descrita acima é aplicada a seis restrições de comportamento: uma restrição de tensão ao longo do riser, duas restrições ao ângulo de topo, uma restrição aplicada a variação do ângulo de "buit-in" da configuração, uma restrição a tração de topo e uma restrição ao nível de tração ao longo do riser.

Tensões

Com relação ao critério de avaliação de tensões, a tensão equivalente de Von Mises (σ_e) deve respeitar o limite máximo definido segundo a API conforme citado anteriormente.

A função de penalidade definida anteriormente pode ser então utilizada corretamente fornecendo a ela o valor de ordenada conforme definido abaixo, de forma que para x > 1 as soluções não são restritas quanto ao critério de tensão:

$$x = \frac{1.5 \times C_a \times C_f \times \sigma_y}{\sigma_e} \qquad \textit{(eq. 20)}$$

Ângulo de Topo

O ângulo de topo da configuração pode ser limitado segundo critérios funcionais tanto por um limite superior quanto por um superior. Estas restrições podem ser atendidas aplicando a função de penalidade usando consecutivamente as seguintes ordenadas:

$$x = \left(\frac{\alpha_{\max}}{\alpha} \right) \quad e \quad x = \left(\frac{\alpha}{\alpha_{\min}} \right) \qquad \textit{(eq. 21)}$$

Onde α é o ângulo de topo da configuração analisada, αmax é o limite máximo e αmin é o limite mínimo para o ângulo de projeto.

Variação do Ângulo de "built-in"

É também comum, para risers rígidos, a especificação de uma "flex joint" para a extremidade superior do riser, que impões uma limitação de deflexão angular. Ou seja, limita mais especificamente a variação angular de topo relativa à definição inicial na posição de projeto (*built-in angle*). Assim podemos definir a ordenada a ser fornecida para a mesma função de penalidade utilizada anteriormente seguinte forma:

$$x = \frac{\Delta\alpha}{\Delta\alpha_{flex}} \qquad \textit{(eq. 22)}$$

Onde Δα é a variação máxima no ângulo de built-in da configuração analisada e Δαflex é a variação admissível no ângulo de built-in, limitada pela flex-joint.

Tração de Topo

É também muito comum limitar-se a tração no topo do riser a um valor máximo. Para este fim basta aplicar a função de penalidade previamente definida a seguinte ordenada:

$$x = \frac{T_{\max}}{T_{topo}} \qquad \textit{(eq. 23)}$$

Onde Ttopo é a tração atuante no topo da configuração e Tmaxé a tração limite máxima no topo.

Tração ao longo do Riser

Por último, pode ser interessante limitar a tração ao longo do riser a um valor mínimo. Para utilizar novamente a mesma função de penalidade, basta então acrescentar um critério condicional. Sendo Tmin a tração mínima recomendada e T a menor tensão axial atuante ao longo do riser, a ordenada da função de penalidade pode ser definida da seguinte forma:

Se Tmin > 0 e T > 0 - $x = T / T_{\min}$

Se Tmin < 0 e T < 0 - $x = T_{\min} / T$

Se Tmin > 0 e T < 0 - $x = 0$

Se Tmin < 0 e T > 0 - $x = 1$

Existe ainda uma restrição absoluta, que retira do processo de otimização as configurações em que a menor distância entre o trecho inicial de catenária (entre a conexão de topo até o início dos flutuadores) e o solo ficar abaixo de um mínimo valor pré-estabelecido.

5.3 Modelos de análise

As penalidades são estabelecidas a partir de resultados de análises estruturais para cada indivíduo. Dentro da interface que envolve o Algoritmo Genético e o Programa de Análise de Linhas e Unidades Flutuantes, podemos empregar dois métodos diferentes:

- Um método baseado em uma formulação analítica que emprega as equações da catenária;
- Um método numérico, baseado no MEF (Método dos Elementos Finitos).

A formulação baseada nas equações de catenária tanto pode ser empregada para fornecer os resultados aproximados, quanto para gerar as malhas e outros dados iniciais para as rotinas do MEF.

Para tratar a restrição de ângulo de topo, mostrada no item anterior, não há a necessidade de se executar análise pelo MEF da configuração do sistema de risers, nem mesmo na posição média (Mean), já que este é o único dado coletado nesta posição, e o ângulo de topo pode ser informado com suficiente precisão a partir de uma análise simples pelas equações de catenária.

Para a avaliação das tensões de Von Mises surgem duas hipóteses: é possível fazer uma avaliação simplificada, com base nos resultados das equações de catenária ou considerar os resultados mais criteriosos obtidos com o método dos elementos finitos.

Na avaliação simplificada, os resultados de força axial e curvatura são utilizados para o cálculo de tensões da seguinte forma:

tensão normal (σ_n):

$$\sigma_n = \frac{F_x}{A_x} \qquad \textit{(eq. 24)}$$

Onde F_x é a força axial e A_x é a área da seção transversal do riser.

tensão de flexão (σ_f)

$$\sigma_f = \frac{E \cdot \frac{d_{ext}}{2}}{R} \qquad \textit{(eq. 25)}$$

Onde E é o módulo de elasticidade do material, d_{ext} é o diâmetro externo da seção, e R é o raio de curvatura no ponto analisado do riser. Sendo R o inverso da curvatura.

Assim, a tensão equivalente de Von Mises foi considerada, aproximadamente, como o somatório das tensões normal e de flexão, ou seja:

$$\sigma_e = \sigma_n + \sigma_f \qquad \textit{(eq. 26)}$$

Vale lembrar que esta aproximação não é válida caso seja requerida uma avaliação precisa dos valores de tensão. Contudo, como o objetivo desta avaliação é o de fornecer um critério para o algoritmo classificar qualitativamente as soluções candidatas, considera-se válido o procedimento na síntese de soluções potencialmente boas.

Na formulação de solução pelo MEF é possível considerar todos os tipos de carregamento aos quais o sistema estará exposto: cargas gravitacionais, correnteza, onda e movimentos impostos pelo flutuante. Surge então a possibilidade de escolha entre análises estáticas ou análises dinâmicas.

Em casos reais, onde o objetivo é a capacitação de um projeto de sistema de risers para utilização prática são exigidas sempre análises dinâmicas. Contudo durante o processo de evolução do algoritmo em busca da solução ótima são geradas soluções candidatas que nem sempre atendem aos requisitos de projeto, mesmo quando expostas apenas aos carregamentos estáticos especificados. Desta forma, analisar estas configurações dinamicamente seria desperdiçar tempo de processamento com soluções certamente não viáveis.

Mas, devemos estar cientes de que uma pequena modificação numa configuração que foi penalizada, mesmo que os critérios tenham sido violados estaticamente, pode tornar a mesma uma boa solução quando exposta aos carregamentos completos. Assim, as soluções penalizadas estaticamente não devem ser excluídas do processo evolutivo já que constituem material "genético" valioso para as próximas gerações.

Para que não se gaste tempo computacional inadequadamente, contudo sem desprezar soluções estaticamente penalizadas, foi admitido um critério de análise global misto na evolução do algoritmo. Por este critério, uma configuração somente é levada á análise dinâmica caso seja viável estaticamente, ou seja, para que se dispare uma análise dinâmica inicialmente a configuração deve ter sido analisada estaticamente nas posições Near e Far e não ter violado nenhuma das restrições. Caso durante as análises estáticas a configuração viole uma das restrições adotadas, ela não será analisada dinamicamente, mas será mantida na população com seu valor de avaliação relativo aos valores de penalidade que obteve estaticamente.

5.4 Critério de convergência

Além do critério de convergência clássico, que interrompe as iterações do algoritmo quanto atingido um número predeterminado de gerações, ainda estão implementados na ferramenta desenvolvida mais dois critérios, ambos baseados nos valores de avaliação dos indivíduos calculados em gerações anteriores.

O primeiro critério de convergência desenvolvido interrompe a evolução quando o valor das médias das avaliações realizadas em cada geração permanece maior ou igual a um percentual (p) do valor da avaliação do atual melhor indivíduo durante as (n) últimas gerações. Com esta opção ativa o valor de (p) deve se aproximar 100% , caso contrário o algoritmo pode ter a execução interrompida ainda com a média instável, o que levaria a admitir a convergência com uma população ainda bastante heterogenia. Quando o valor de (p) for 100% somente se admite convergência quando toda a população obtiver o mesmo valor de avliação. Quanto maior o número de gerações (n) atingido pelo critério mais segurança se tem na convergência adequada do algoritmo, por outro lado se um número pequeno de gerações for considerado é possível que se encerre uma execução que ainda apresenta possibilidades de melhora. Vale ainda alertar que valores altos de mutação podem levar a não convergência por este critério, sendo a execução interrompida somente quando atingido o número máximo de gerações.

O segundo critério desenvolvido encerra a execução quando a variação percentual do valor da média de uma geração, em relação ao valor médio da geração anterior for inferior a (1-p) durante (n) gerações consecutivas. Desta forma a execução é interrompida sempre que a população estiver estável, não se importando se a estabilidade fica em torno de um valor bom ou ruim de avaliação, apenas identifica que o algoritmo tem poucas chances de promover uma boa investigação do espaço de busca. Este critério é bastante útil para análises de configurações caras computacionalmente, onde é interessante investir em tempo de análise apenas em execuções de performance ascendente com o número de gerações. Também nesta opção valores altos de mutação podem levar a não convergência, e novamente a execução somente seria interrompida quando atingido o número máximo de gerações.

Ainda é possível combinar os dois critérios acima, de forma que a execução seria interrompida apenas caso as condições impostas por ambos critérios sejam satisfeitas plenamente.

6 Descrição da Ferramenta

6.1 Configuração inicial

Antes de dar início ao processo de síntese e otimização de configuração de riser, é necessário definir uma "configuração base", que servirá de ponto de partida para a geração de cada indivíduos do processo de síntese. A criação e edição da configuração básica de um riser em catenária simples ou lazy-wave pode ser realizada através da tela de edição de linhas, ilustrada na figura abaixo.

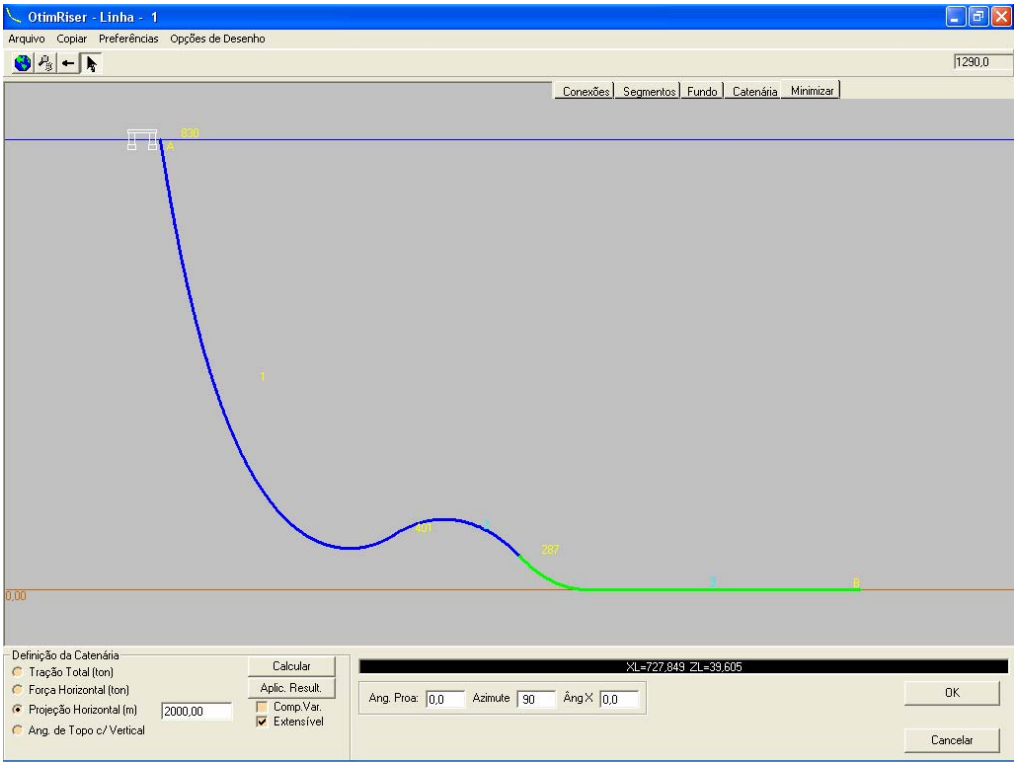


Figura *20*: Tela de Edição de Linhas

Observa-se que a geometria da configuração básica pode ser definida em termos de quatro parâmetros diferentes: tração total, força horizontal, projeção horizontal ou ângulo de topo com a vertical. Existe ainda nesta mesma tela quatro abas, as quais serão descritas a segir.

6.1.1 Conexões

Na aba *conexões*, ilustrada na figura a seguir, é possível definir as coordenadas das conexões de fundo e topo. Observa-se que, de acordo com o parâmetro escolhido para a geração da catenária, pode não ser necessário definir as coordenadas de uma das conexões, que serão ajustadas automaticamente para atender o critério de geração.

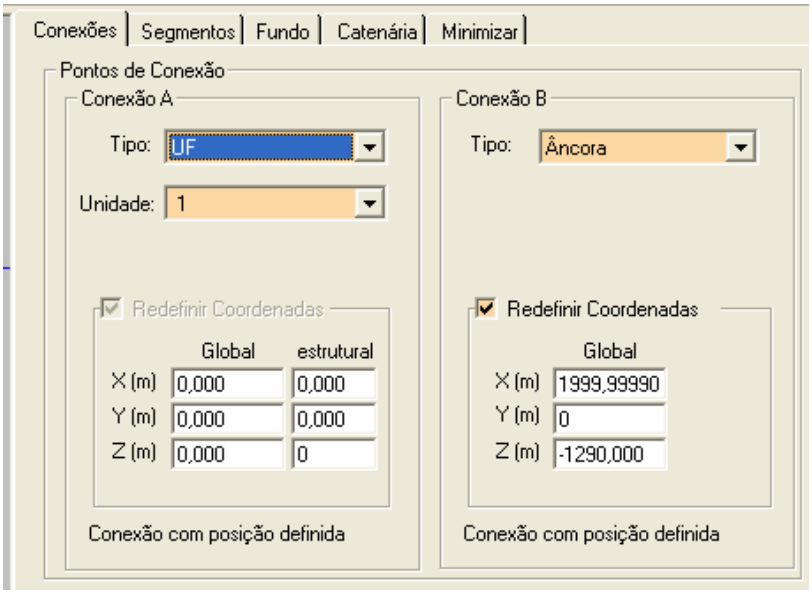


Figura 21: Conexões

6.1.2 Segmentos

***Lay-Out* da aba Segmentos**

O lay-out da aba *segmentos* é ilustrado na Figura a seguir. A partir desta aba é possível definir as propriedades de cada tipo de segmento que irá compor a configuração básica do *riser*, e definir outras propriedades da configuração básica, incluindo comprimento, tipo e gradação da malha de cada segmento; densidade do fluido interno e pressão no topo do *riser*.

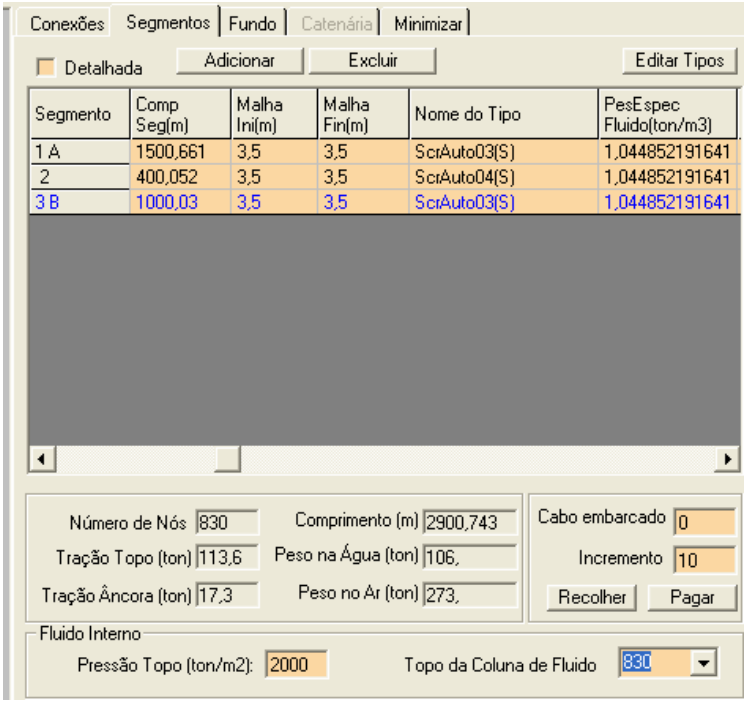


Figura 22: Aba "Segmentos"

Biblioteca de Tipos de Segmentos

Como mencionado anteriormente, a partir da aba *segmentos* é possível definir as propriedades de cada tipo de segmento que irá compor o *riser*. No caso de um riser lazy-wave, por exemplo, podem ser definidos dois tipos de segmentos, um para o corpo do riser, outro para o trecho com flutuadores.

Para isso, o usuário deve clicar no botão "***Editar Tipos***", ativando uma tela específica para a definição das propriedades físicas, geométricas e hidrodinâmicas de cada segmento. Nesta tela, os dados de segmentos podem ser editados e visualizados em dois modos: o modo *detalhado* e o modo *tabela*, como ilustrado nas Figuras a seguir.

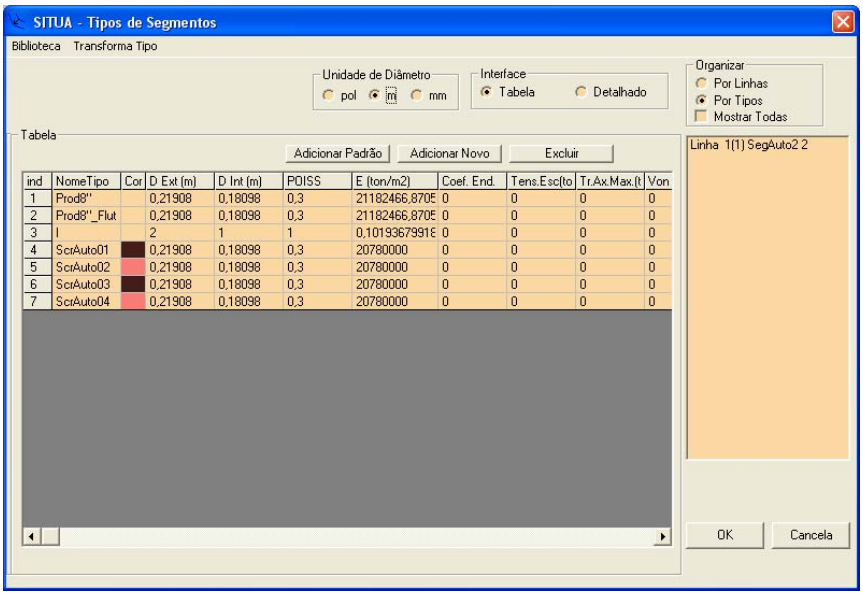


Figura *23*: Tela "Tipos de Segmentos" - modo Tabela

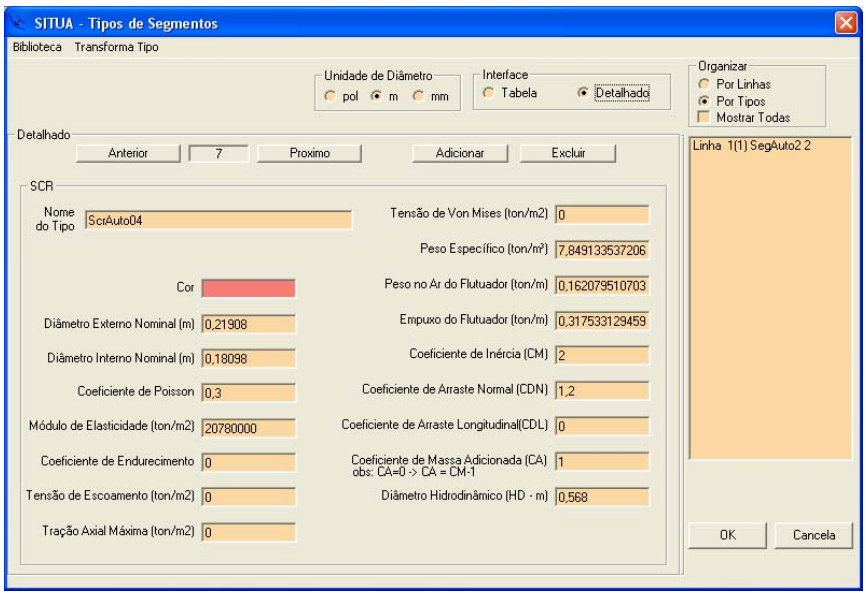


Figura *24*: Tela "Tipos de Segmentos"- modo Detalhado

Definição dos Segmentos da “configuração básica”

Uma vez definidos os tipos de segmentos, através da aba *segmentos* é possível montar a configuração básica do riser definindo o comprimento e gradação da malha de cada segmento da “configuração básica”, e associando o nome do tipo de segmento definido anteriormente na tela *Tipos de Segmentos*. Na aba *segmentos* é possível também definir outros dados básicos operacionais do *riser*, tais como a densidade do fluido interno e a pressão no topo do *riser*.

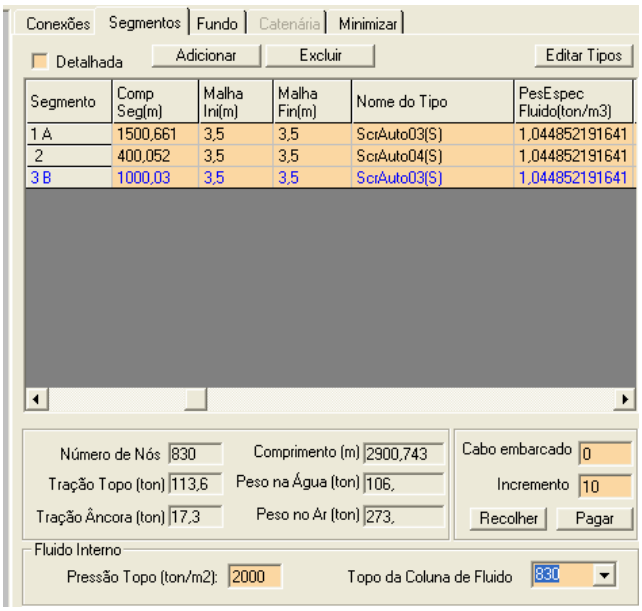


Figura *25*: Segmentos

6.1.3 Dados de Fundo

A aba relativa ao fundo permite a definição de todas as variáveis relativas a modelagem de contato com o solo, como ilustrado na Figura abaixo.

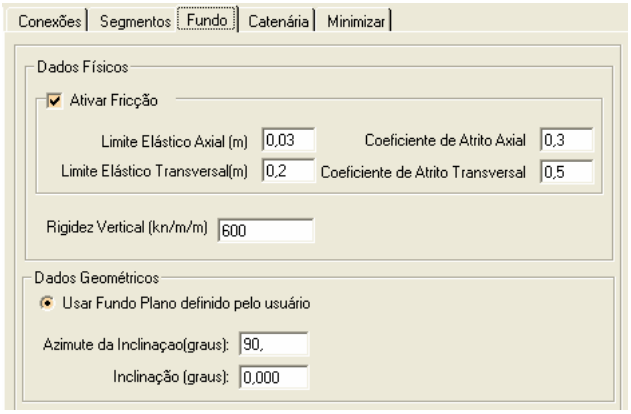


Figura *26:* Dados de Fundo

6.1.4 Resumo da Configuração em Catenária

Finalmente, a aba "***Catenária***", ilustrada na figura abaixo, apresenta um resumo da configuração gerada pelo pré-processador baseado nas equações da catenária, a partir dos dados fornecidos nas abas anteriores.

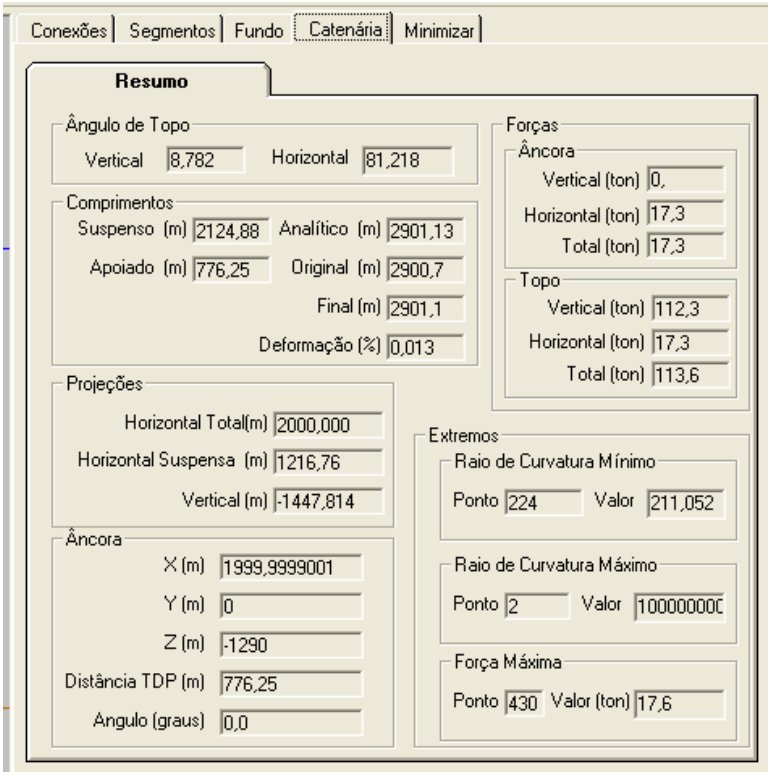


Figura 27: Resumo de Catenária

6.2 Controle do Processo de Síntese e Otimização

Toda a interface gráfica relativa à definição dos parâmetros e ao acompanhamento da evolução do processo de síntese ficam contidos em um formulário com quatro abas descrito a seguir.

6.2.1 Variáveis de Projeto e Restrições

Na primeira aba, mostrada na figura a seguir, existem dois quadros relacionados a diferentes etapas de descrição do problema de otimização: a definição do tipo de configuração a ser estudada (trechos com flutuabilidade) e suas respectivas variáveis de projeto, e definição das restrições a serem adotadas durante o processo.

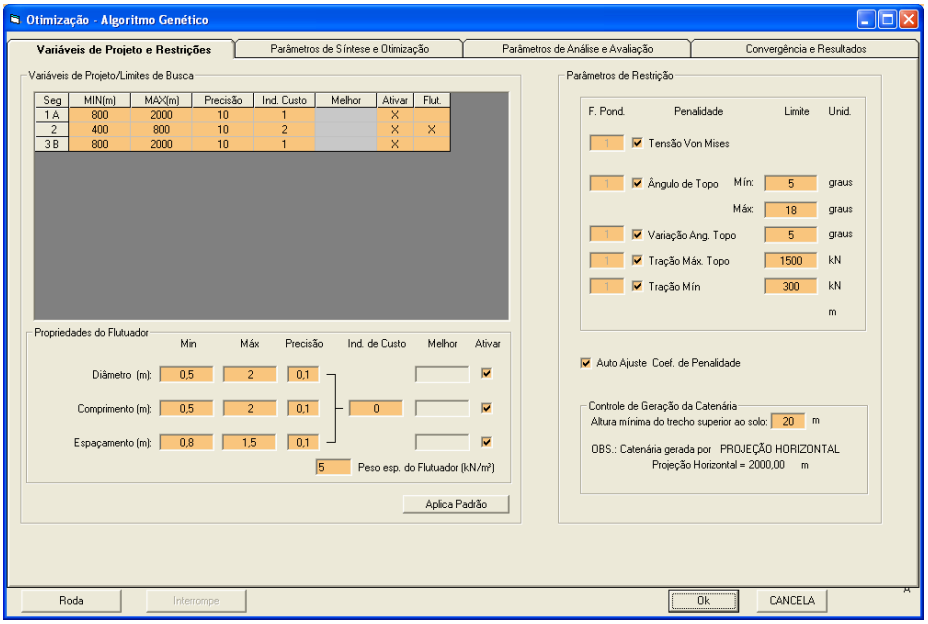


Figura 28: Formulário principal de síntese e otimização

As variáveis livres e os limites de busca

No quadro relacionado ao ajuste de limites de busca, mostrado na figura a seguir, são definidos os espaços de busca de cada variável do problema, cada um composto de limites inferior e superior do espaço e sua precisão associada. É também neste quadro que devem ser definidas as relações de custo de cada componente da configuração.

Além disso, é possível, através das opções de ativação colocada à direita de cada variável de projeto, optar em excluir alguma variável da busca, tornando-a assim um parâmetro fixo, assumido igual ao valor descrito na configuração básica da linha.

A opção "*aplicar padrão*" apenas aplica valores de custo e precisão previamente determinados, de maneira a facilitar repetidas utilizações da ferramenta.

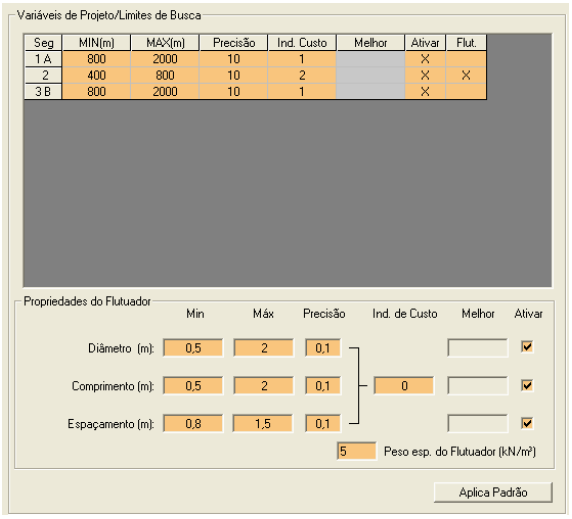


Figura 29: Variáveis de Projeto / Limites de Busca

Restrições

Este quadro permite fornecer todos os valores relacionados às várias restrições ou funções de penalidade criadas para a avaliação dos indivíduos no processo de otimização, e ainda. escolher qual restrição de comportamento estará ativa durante um processo. Ainda nesta mesma aba é possível também optar pelo ajuste automático dos fatores de ponderação de cada penalidade, ou simplesmente atribuir um valor fixo para cada um destes coeficientes.

É necessário esclarecer que caso a opção de geração da catenária marcada na tela de definição da configuração base seja por ângulo de topo, a penalidade relativa a esta variável fica automaticamente desativada.

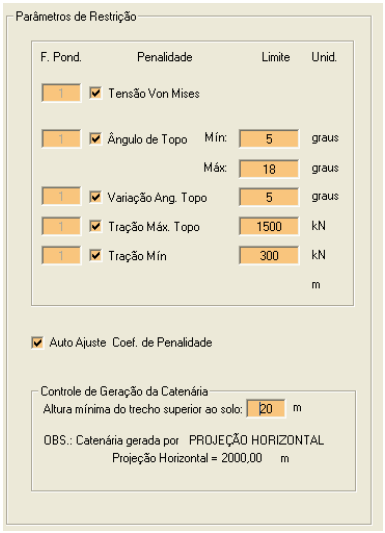


Figura 30: Parâmetros de Restrição

6.2.2 Parâmetros de Síntese e Otimização

Nesta segunda aba ilustrada na figura abaixo, dividida em três quadros, é possível ajustar por completo o funcionamento da ferramenta do ponto de vista dos critérios adotados para a evolução algoritmo genético propriamente dito.

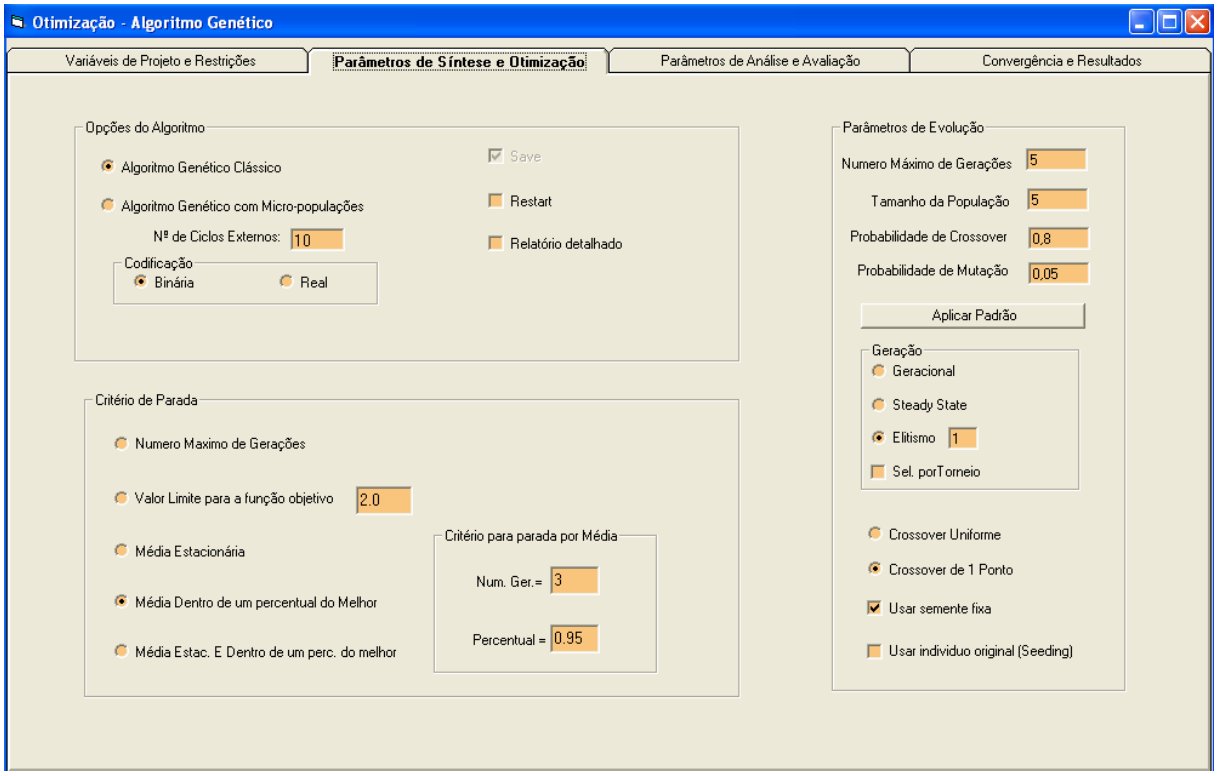


Figura 31: Parâmetros de Síntese e Otimização

Opções do Algoritmo

No quadro "Opções do Algoritmo" é possível optar pelo ***Algoritmo Genético Clássico*** ou pelo ***Algoritmo Genético com Micro-populações*** sendo obrigatório neste caso definir o número de ciclos externos do algoritmo e ainda optar pelo tipo de codificação adotada, e por fazer o acompanhamento com relatório detalhado ou não.

Neste mesmo quadro, a opção "Save" tem o objetivo de armazenar intermitentemente os dados de uma evolução para que ela possa ser recomeçada posteriormente após um ponto de parada determinado. A opção "Restart"por sua vez tem a função de recomeçar uma evolução guardada pela primeira opção.

Parâmetros de evolução

Neste quadro é possível fazer uma opção exclusiva entre os três tipos implementados para substituição de gerações: ***Geraciomal***, ***Steady State*** ou ***Elitism***o. Como padrão, a substituição pode ainda ser feita por ***Ranking***, ou por ***Torneio*** caso seja marcada a caixa de verificação referente a esta opção. É possível ainda fazer a opção entre ***Crossover Uniforme*** ou de ***Um Ponto***. Caso seja requerida uma comparação mais exata entre duas evoluções do algoritmo, é possível também optar por fixar a "semente" da geração randômica. É possível ainda manter na primeira geração a configuração original que serve como fonte de dados básicos de geração da linha, marcando a caixa de verificação "Usar indivíduo original".

O botão "***Aplicar Padrão***" atribui às caixas de texto relativas ao tamanho da população, e probabilidades de crossover e mutação, valores estabelecidos na literatura como eficientes para a maioria dos casos, de acordo com o tipo de algoritmo escolhido. No entanto tais valores podem sempre ser editados livremente pelo usuário.

Critérios de Parada

A opção referente à escolha do critério de parada adotado pelo algoritmo é excludente podendo ser feitas as seguintes opções.

A opção "*n. máximo de gerações*" refere-se ao critério clássico de parada, cujo objetivo é interromper as iterações do algoritmo quanto atingido um número predeterminado de gerações, que deve ser definido na primeira caixa de texto do quadro de parâmetros de evolução.

A opção "*limite para função objetivo*" deve ser escolhida quando é conhecido um valor máximo ou simplesmente satisfatório para a função objetivo do problema, valor este que deverá ser preenchido junto à opção referida.

A opção pelo terceiro critério de parada interrompe a evolução quando o valor das médias das avaliações realizadas em cada geração permanece maior ou igual a um percentual (p) do valor da avaliação do atual melhor indivíduo durante as (n) últimas gerações. Isso corresponderia, no caso relacionado na figura acima, a interromper a evolução quando o valor das médias das avaliações realizadas em cada geração permanece maior ou igual a 95% do valor da avaliação do atual melhor indivíduo durante as 6 últimas gerações

A quarta opção encerra a execução quando a variação percentual do valor da média de uma geração, em relação ao valor médio da geração anterior for inferior a (1-p) durante (n) gerações consecutivas. Por exemplo, caso seja fornecido o valor percentual de 95% e o número de gerações igual a 6, a execução seria interrompida caso a variação percentual do valor da média de uma geração, em relação ao valor médio da geração anterior se mantiver inferior a (1-95)%, ou seja, 5%, durante 6 gerações consecutivas.

A escolha pela última opção da interface faz com que a execução seja interrompida se ambos os critérios definidos na terceira e quarta opções sejam atendidos.

Caso seja feita a opção pelo segundo, terceiro ou quarto critério de parada e as condições estabelecidas por esta opção não sejam atendidas até atingido o número máximo de gerações anteriormente definido, a execução será interrompida impreterivelmente.

6.2.3 Parâmetros de Análise e Avaliação

Nesta aba podem ser definidos o tipo de análise a ser empregada na aquisição dos valores relacionados com as restrições de comportamento, bem como o carregamento ao qual estarão submetidas às configurações candidatas ao longo do processo de síntese e otimização, conforme descrito a seguir.

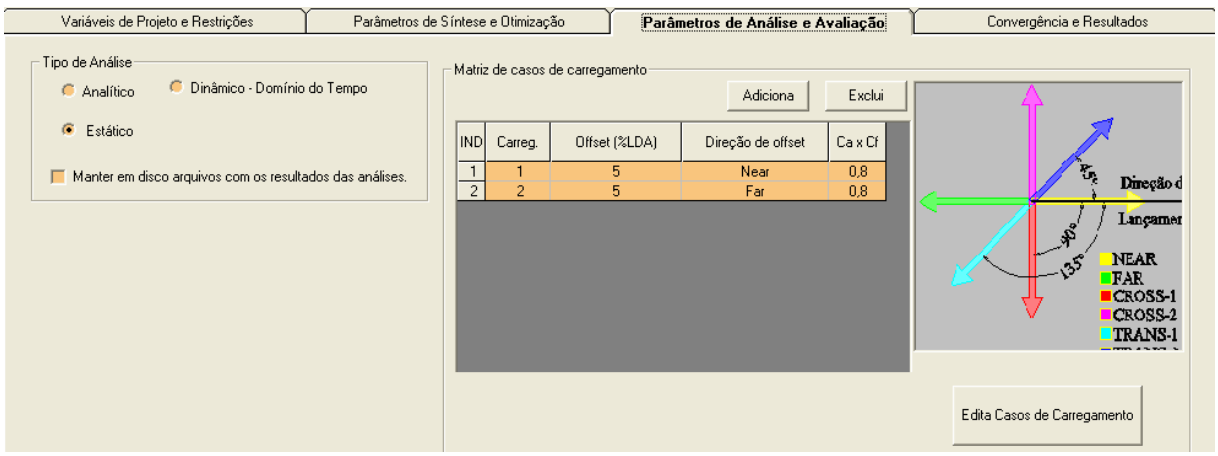


Figura 32: Parâmetros de Análise e Avaliação

Tipo de Análise para Avaliação

A escolha pelo método de análise adotado é feita de forma excludente e as opções são explicadas a seguir.

As duas primeiras opções correspondem a soluções estáticas, que não levam em conta o comportamento dinâmico do *riser*, e que portanto não são indicadas para a otimização propriamente dita, admitindo-se seu uso apenas para fins de estimar limites de busca para uma posterior análise mais exata do problema. A primeira opção corresponde ao uso de um algoritmo de solução analítica baseado nas equações da catenária; a segunda opção equivale ao procedimento de solução baseado em Elementos Finitos, considerando apenas análises estáticas. A última opção refere-se ao uso de procedimentos de solução baseado em Elementos Finitos e considerando análises estruturais dinâmicas.

Matriz de Casos de Carregamento

Neste quadro é possível criar uma matriz de casos de carregamento que irá compor a seqüência de análises seguida para as opções que incluem análise por elementos finitos para avaliar cada indivíduo do processo de otimização. Durante a avaliação de um indivíduo do processo de síntese, serão considerados nas análises todos os casos da matriz que estiverem marcados como “ativos”. São admitidos quantos casos de carregamento forem necessários, contudo foram definidas seis direções básicas de aplicação de offset: Near, Far Cross-1, Cross-2, Trans-1, Trans-2.

Os casos de carregamento que compõem a matriz, numerados na primeira coluna desta tela, são criados através da edição de casos de carregamento. Conforme mostrado a seguir, é possível fazer conbinações de componentes de carregamento ambiental de onda, correnteza e offset.

Na figura a seguir a aba ativa mostra a definição dos dados de correnteza. Estes dados podem ser definidos de acordo com a Especificação Técnica (ET) de Dados Meteo-oceanográfico previamente armazenada em um arquivo que é disponibilizado separadamente da interface (e portanto pode ser omitido caso razões de sigilo impeçam sua distribuição). Os dados de correnteza podem ainda serem definidos completamente a critério do usuário. É possível nesta aba definir, editar ou excluir quantas correntezas de projeto forem necessárias.

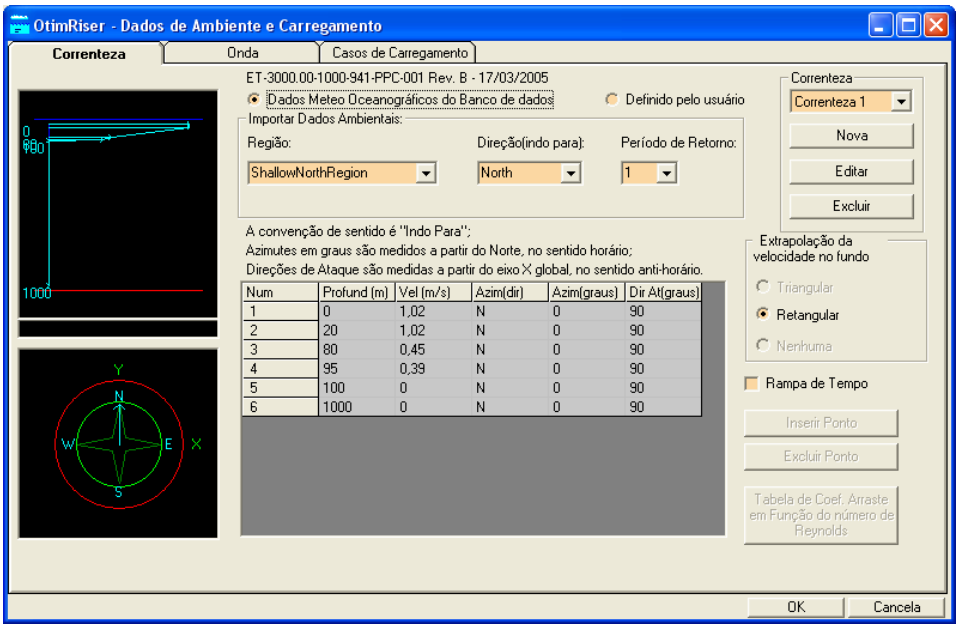


Figura 33: Dados de Correnteza

Na figura seguinte, que ilustra a aba relativa ao carregamento de onda, observa-se que também é possível optar entre definir manualmente um carregamento qualquer ou recorrer à ET de dados meteoceanográficos. Assim como no caso da correnteza, podem ser criadas e editadas tantas ondas quantas forem necessárias. O usuário pode utilizar mar regular ou mar irregular, sendo que nesta última alternativa o mar poderá ser unidirecional ou bidirecional.

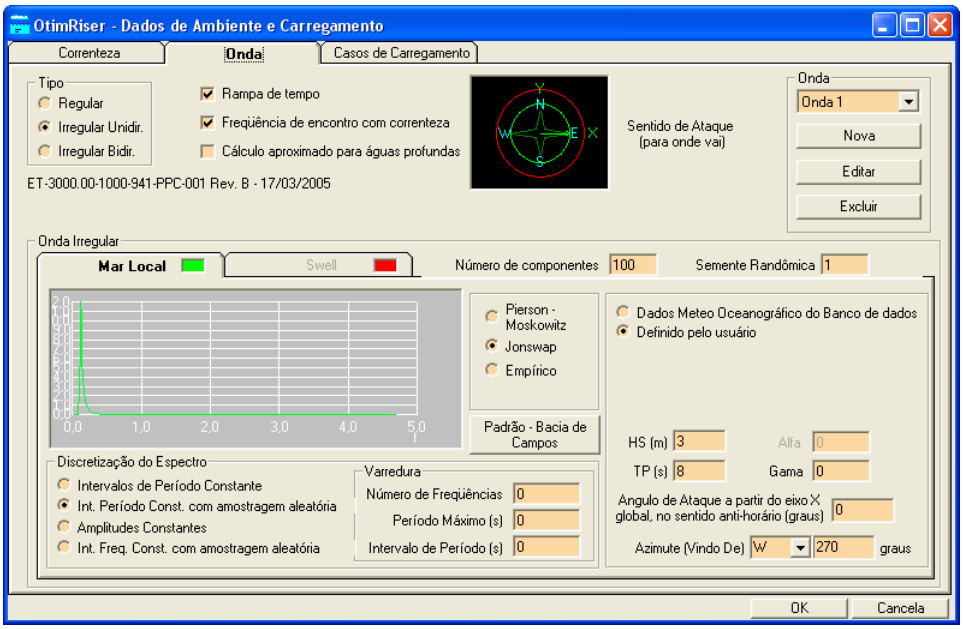


Figura 34: Dados de onda

Depois de definidos todos os dados de cada componente do carregamento estas podem ser combinadas formando casos de carregamento, utilizando-se a aba mostrada na figura a seguir.

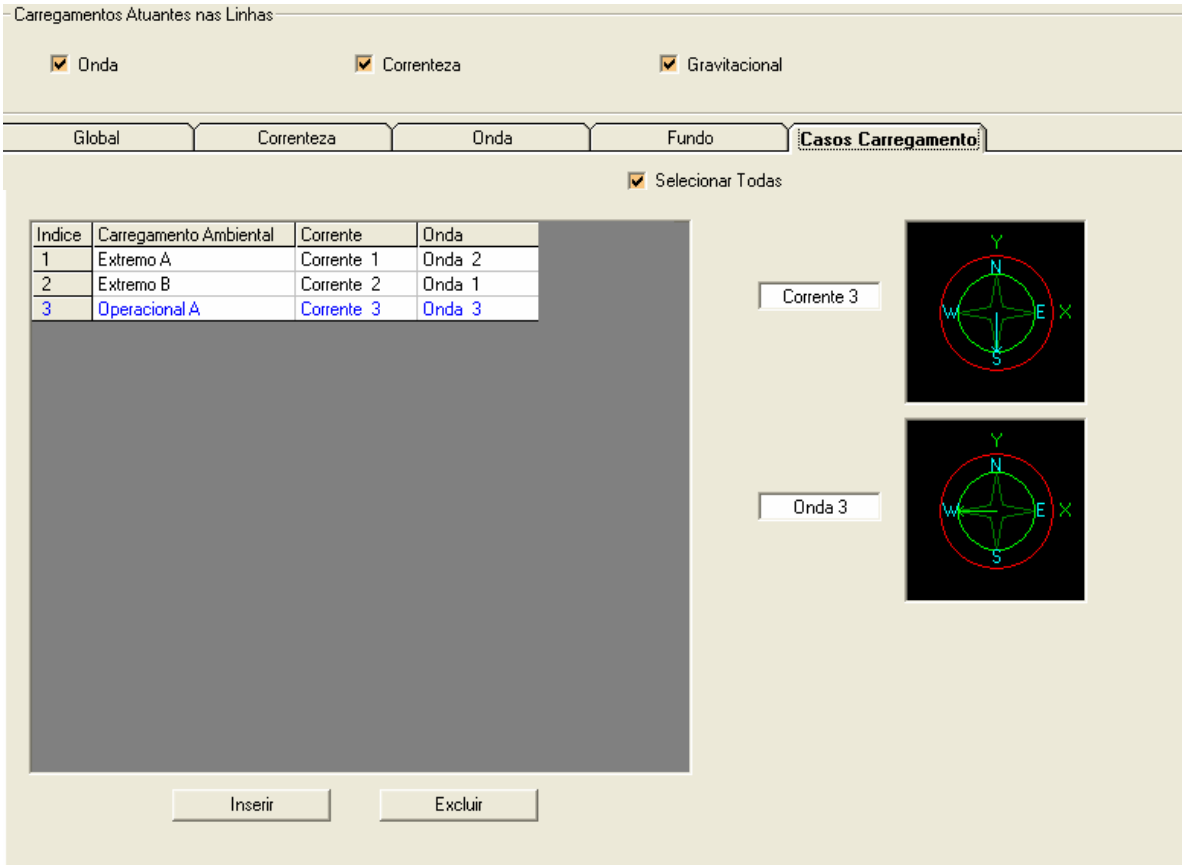


Figura 35: Dados de onda

6.3 Convergência e Resultados

Finalmente, a quarta aba é responsável pelo acompanhamento da evolução, convergência e apresentação dos resultados do processo de otimização, conforme ilustra a figura a seguir.

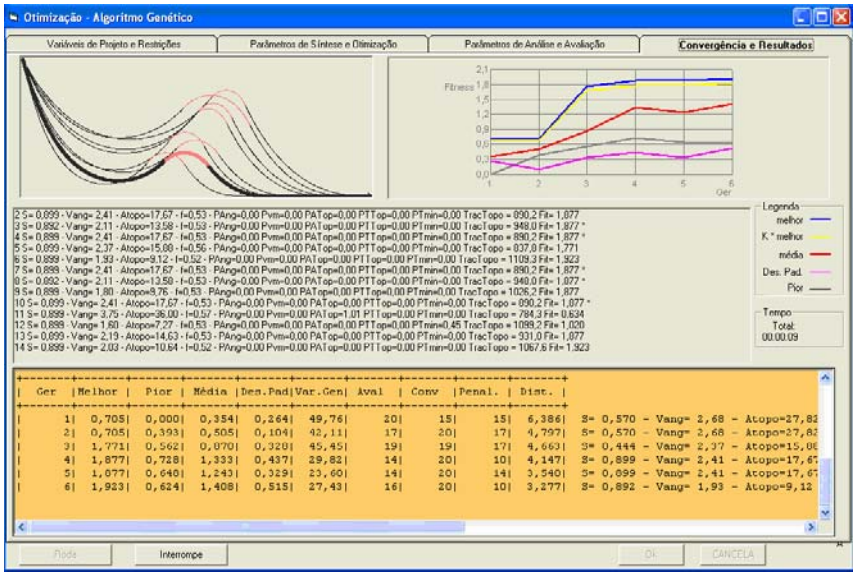


Figura 36: Convergência e Resultados

O quadro disposto no canto superior esquerdo apresenta, em tempo real durante todo o processo, uma representação esquemática da configuração geométrica de cada indivíduo da geração atual e do melhor indivíduo da geração anterior, sendo que este último aparece em linhas mais espessas de forma a ganhar destaque, como ilustrado na figura a seguir.

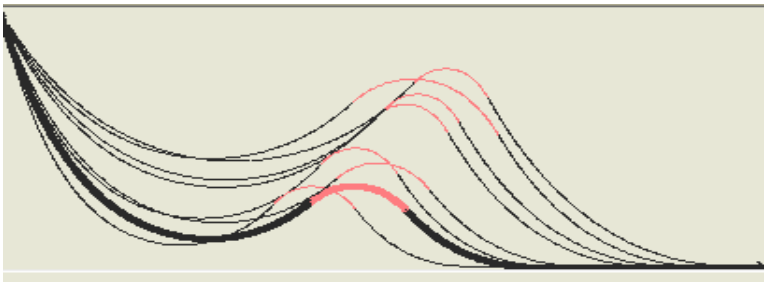


Figura 37: Representação esquemática das configurações

O quadro no canto superior direito apresenta um gráfico de convergência, que mostra valores de avaliação no eixo vertical e o índice das gerações no eixo horizontal, que também é atualizado em tempo real, como ilustra a figura seguinte.

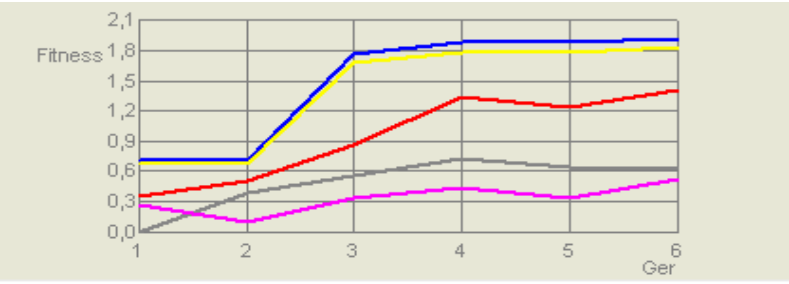


Figura 38: Gráfico de convergência

A curva em azul mostra o valor do melhor indivíduo de cada geração, e a curva amarela, sempre paralela a anterior, mostra apenas o limite de convergência, ou seja, equivale ao valor da curva azul multiplicado pelo valor percentual do critério de convergência. A curva vermelha e a curva magenta mostram consecutivamente média e desvio padrão das avaliações de todos os indivíduos da população em cada geração, incluindo os não convergentes, que recebem avaliação igual a zero. A curva preta representa o valor de avaliação do pior indivíduo da população a cada geração, também incluindo possíveis não convergentes.

O quadro intermediário mostra, também em tempo real, informações sobre cada avaliação: fator de utilização (inverso do fator de segurança), variação de ângulo de topo, ângulo de topo e o valor da função de avaliação, como pode ser visto na figura abaixo.

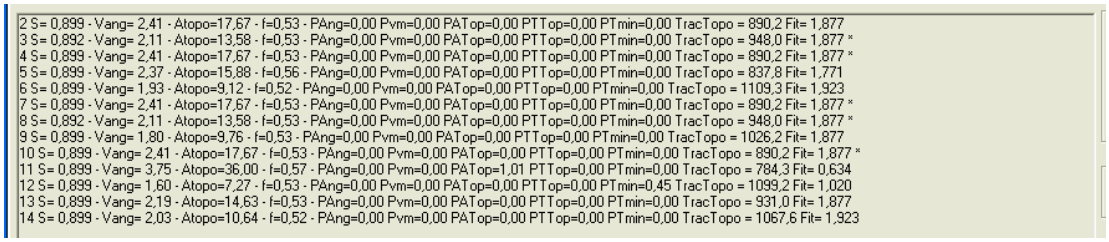
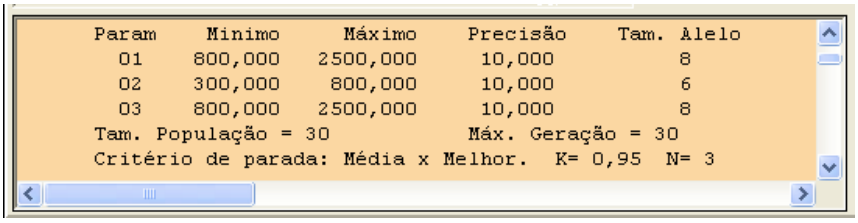


Figura 39: Gráfico de convergência

No quadro inferior é disposto um resumo do relatório de todo o processo evolutivo. O relatório completo é gravado em arquivo e pode incluir a descrição de cada solução gerada ou apenas a melhor solução de cada geração, fazendo respectivamente a opção por relatório detalhado ou não.

No relatório é mostrado ainda um resumo das restrições laterais do problema e demais parâmetros definidos para a busca e evolução do algoritmo. É registrada também uma descrição estatística do processo de otimização inteiro e de cada geração separadamente contendo as informações como mostradas nas figuras seguintes. As figuras adiante mostram o resumo do relatório comentado acima.

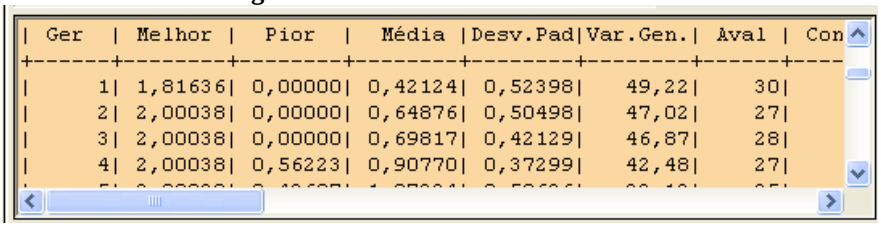


Param	Minimo	Máximo	Precisão	Tam. Alelo
01	800,000	2500,000	10,000	8
02	300,000	800,000	10,000	6
03	800,000	2500,000	10,000	8

Tam. População = 30 Máx. Geração = 30

Critério de parada: Média x Melhor. K= 0,95 N= 3

Figura 40: Resumo – limites de busca



Ger	Melhor	Pior	Média	Desv.Pad	Var.Gen.	Aval	Con
1	1,81636	0,00000	0,42124	0,52398	49,22	30	
2	2,00038	0,00000	0,64876	0,50498	47,02	27	
3	2,00038	0,00000	0,69817	0,42129	46,87	28	
4	2,00038	0,56223	0,90770	0,37299	42,48	27	

Figura 41: Resumo – informações das gerações

6.4 Processamento Paralelo

Uma possibilidade poderosa em termos de desempenho da ferramenta é o processamento paralelo utilizando cluster de computadores. Esta opção é disponível apenas caso a escolha do tipo de análise estrutural para avaliação das configurações empregue análises dinâmicas pelo MEF.

A estratégia escolhida para a utilização da ferramenta em ambiente paralelo é inovadora, considerando que toda a interface gráfica ainda continua disponível durante a análise.

O algoritmo de paralelização escolhido é bem simples e segue o padrão "MESTRE-ESCRAVOS" conforme esquematizado na figura abaixo:

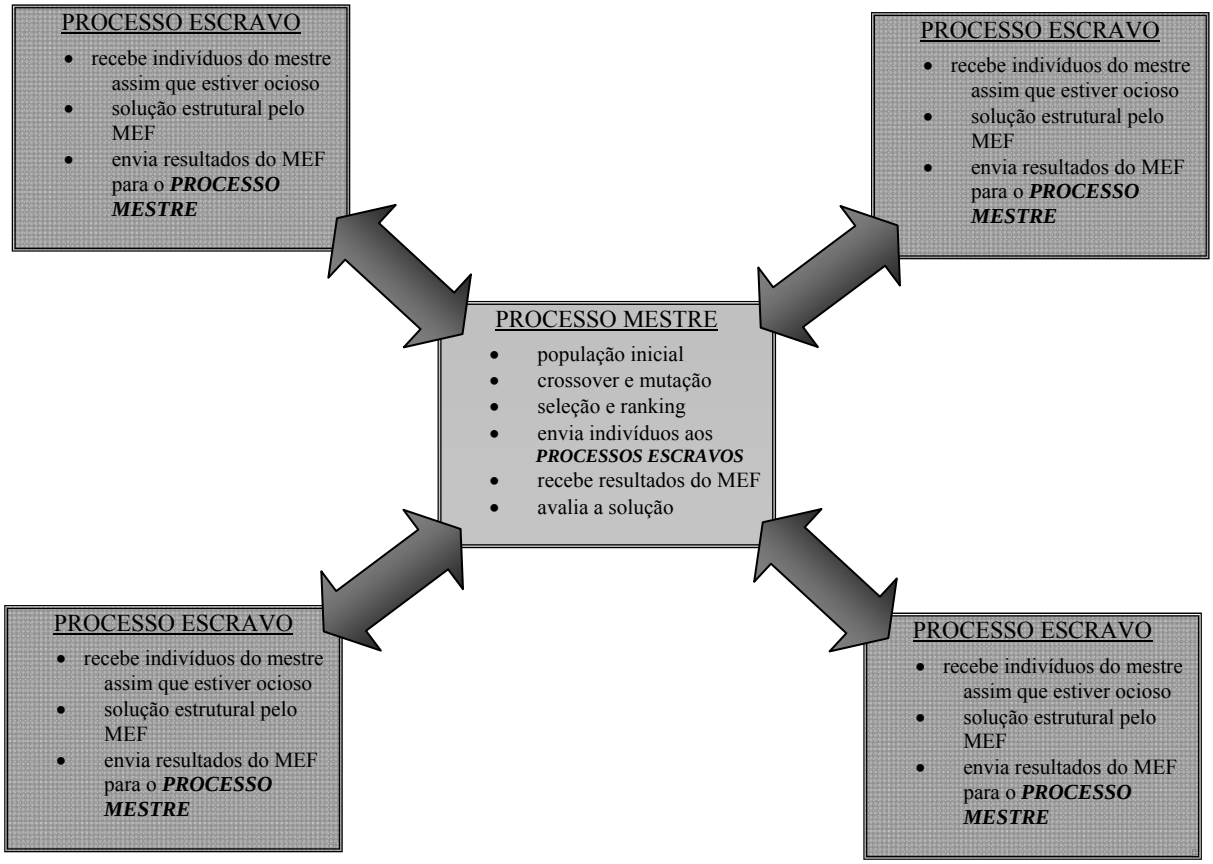


Figura 42: Diagrama esquemático da paralelização

Desta forma, todo o desenvolvimento, ou evolução do algoritmo fica por conta do processo mestre, enquanto o processamento de custo computacional oneroso fica a cargo somente dos processos escravos.

É importante ressaltar que a interface entre mestre e escravos e feita de forma sincronizada, de maneira que um processo escravo somente fica ocioso caso o processo mestre não tenha mais indivíduos para enviar durante uma dada geração. Neste caso o processo escravo ocioso aguarda o início de uma nova geração e conseqüente redistribuição dos indivíduos pelo processo mestre, ou simplesmente aguarda o a finalização do ambiente paralelo.

O processo mestre é a interface propriamente dita, e cada processo escravo é um executável independente e sem interface gráfica alguma. É fundamental ainda lembrar que o MPI tem as opções de codificação apenas em Fortran ou Linguagem C e a interface gráfica do algoritmo genético é totalmente desenvolvida em Visual Basic. Logo, a solução para que a interface não fosse abandonada durante o processamento paralelo foi concentrar toda a codificação do MPI do processo mestre dentro de uma DLL (Dynamic Link Library) escrita em Fortran. Assim, o algoritmo dos processos escravos também foi escrito em Fortran, mas não existe a necessidade de concentrá-lo em uma DLL, sendo feita então a opção por deixar a codificação deste em um programa executável padrão.

Além dos comandos básicos do MPI citados no item 4.2, para gerenciar o trafego de informação entre o processo mestre e os processos escravos, de maneira a deixar estes últimos ociosos pelo menor tempo possível, foram usados apenas os comandos MPI_WAIT e MPI_BARRIER.

Estes comandos foram utilizados em conjunto com um sistema de “flags” coordenados que mantêm o processo mestre informado do status de cada processo escravo (ocupado ou ocioso). Desta forma se houver um indivíduo pendente de análise num determinado momento, o processo mestre o envia para o processo escravo que estiver com o status de ocioso primeiro.

7 Aplicações

7.1 Introdução

Neste capítulo serão apresentadas aplicações do processo de síntese e otimização, procurando ilustrar a funcionalidade e possibilidades da ferramenta computacional desenvolvida.

O estudo de casos foi realizado em três etapas com finalidades específicas. A primeira etapa considera a utilização da ferramenta com a intenção de definir limites de aplicabilidade para uma configuração pré-definida, em um determinado cenário de projeto. Foram então definidos para esta finalidade os quatro casos, A1, A2, A3 e A4, descritos posteriormente.

Na segunda etapa de estudo de casos tem o objetivo de mostrar a possibilidade de utilização da ferramenta na definição dos materiais empregados no projeto de uma determinada configuração. Desta forma, dois casos de estudo descritos a seguir e nomeados como B1 e B2 foram analisados.

A terceira etapa de estudo de casos foi desenvolvida com o objetivo de ilustrar a utilização de análises dinâmicas paralelizadas na avaliação de configurações. Com esta finalidade foram então avaliados os dois últimos casos também descritos posteriormente e nomeados de C1 e C2.

7.2 Primeira etapa de estudo de casos.

Os quatro casos analisados nesta etapa constituem configurações do tipo Lazy-Wave e todos utilizam os mesmos dados de configuração básica e carregamento ambiental apresentados nas tabelas a seguir.

7.2.1 Dados gerais

Tabela 4: Dados da configuração básica da primeira etapa de estudo de casos

MATERIAL		
Densidade	7.800	kg/m³
Peso Específico	77	kN/m³
Tensão de Escoamento	413	MPa
Tensão admissível	277	MPa
Módulo de elasticidade	207.800	MPa
CARACTERÍSTICAS GEOMÉTRICAS DA SEÇÃO TRANSVERSAL		
Espessura	0,019	m
Diâmetro Externo	0,219	m
Diâmetro Interno	0,181	m
DADOS DOS FLUTUADORES		
Peso	0,162	ton/m
Empuxo	0,3175	ton/m
Diâmetro externo	0,568	m
DADOS OPERACIONAIS		
Peso específico do fluido interno	5	kN/m³
Pressão de operação no topo do riser	19,62	MPa
Lâmina d'água	1290	m
Azimute	90	graus
Projeção horizontal	2000	m
Posição da conexão de superfície em relação a linha d'água	0	m

7.2.2 Dados ambientais

Apenas análises estáticas foram empregadas nesta etapa de estudo e casos, logo apenas carregamento de correnteza e offset estático são considerados.

Mesmo sabendo que não é corrente prática de projeto utilizar a combinação de um mesmo perfil de correnteza para todas as direções de offset estático, esta opção pode ser considerada aceitável por se tratar de um estudo acadêmico com o foco em desenvolvimento metodológico.

Tabela 5: Correnteza aplicada aos casos da primeira etapa de estudo de casos

Profundidade (m)	Vel. (m/s)	Azimute (DIR)	Azimute (graus)	Direção de ataque (graus)
0	1.23	N	0	90
100	1.23	N	0	90
350	1.16	N	0	90
500	0.77	N	0	90
100	0.69	N	0	90
1250	0.52	N	0	90
1500	0.36	N-NE	22.5	67.5

Notas:

1. a convenção de sentido é "indo para".
2. azimutes são medidos a partir do norte e no sentido horário.
3. direções de ataque são medidas a partir do eixo x global e no sentido anti-horário.

Para esta etapa de estudo de casos considerou-se como carregamentos atuantes a combinações do perfil de correnteza exposto anteriormente com os deslocamentos estáticos de topo nas direções NEAR, FAR, CROSS1 e TRANS1, conforme ilustrado na figura abaixo. O valor do deslocamento adotado em todas as direções foi o equivalente a 5% da lâmina d'água.

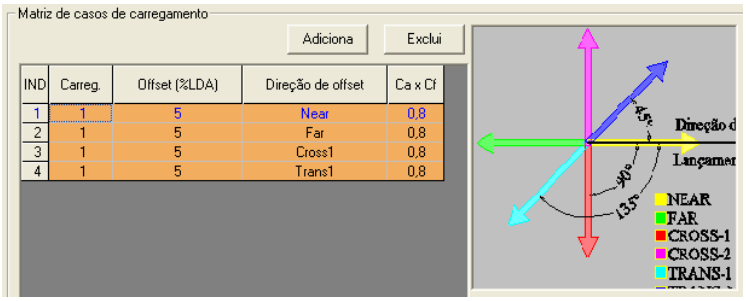


Figura 43: Direções de offset estático

7.2.3 Descrição dos casos em estudo da primeira etapa

A partir do modelo básico definido anteriormente para a primeira etapa foram definidos quatro seguintes casos de estudo O caso ***A1*** é exatamente o modelo básico ao qual são aplicados os limites de busca de variáveis de projeto e restrições de comportamento que podem ser vistas na Figura 44.

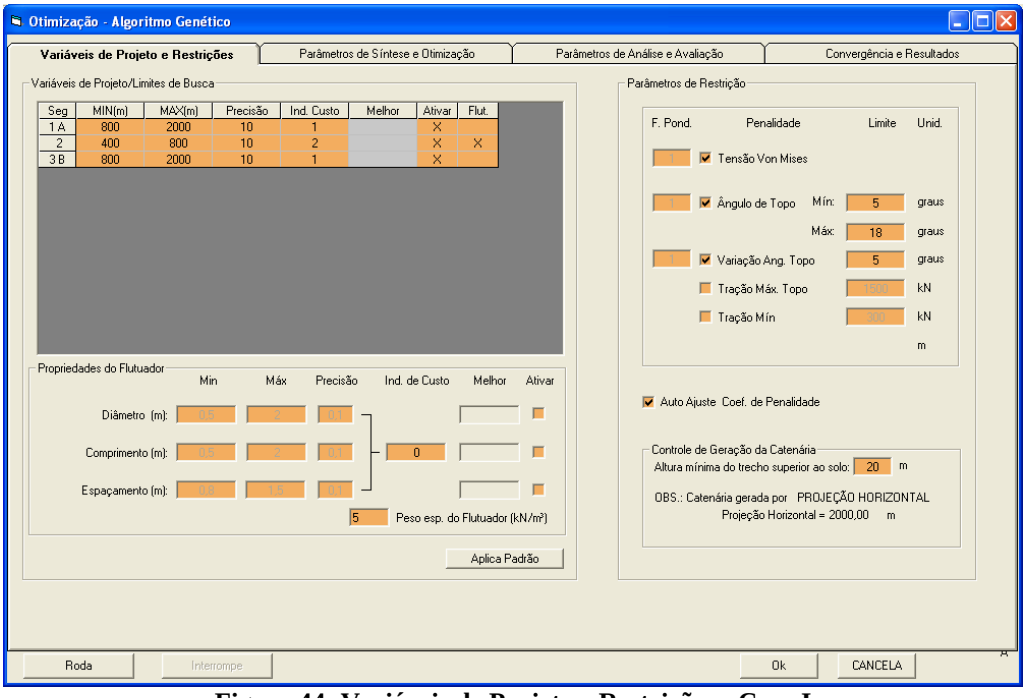


Figura 44: Variáveis de Projeto e Restrições - Caso I

Vale observar que no caso de estudo ***A1*** não é admitida variação para os parâmetros que definem os flutuadores, além de não estarem ativas as restrições de tração máxima no topo e tração mínima.

O ***Caso A2*** mantém os o espaço de busca para as variáveis já contemplada no Caso A1, contudo admite variação dos parâmetros de definição dos flutuadores com novas variáveis de busca. Logo, o algoritmo passa a ignorar os dados de flutuadores definidos na configuração básica e adota os limites de busca definidos para as variáveis inseridas na tela de variáveis de projeto, conforme vistos na Figura 45.

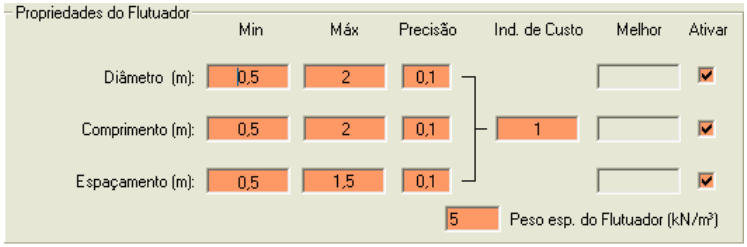


Figura 45: Variáveis de Projeto e Restrições - Caso II

O ***Caso A3*** tem todas as variáveis de busca do Caso A2, com os mesmos limites de busca, mas agora são ativadas também as restrições de tração mínima e tração máxima no topo. Os valores limites para estas restrições, também informados na interface de variáveis de busca e restrições: Atração máxima no topo ficou limitada a 1500 kN e a tração mínima deve neste caso de estudo ser maio que 200kN.

O ***Caso A4*** tem as mesmas variáveis e as mesmas restrições do Caso A3, mas tem agora seu limite máximo de tração no topo um pouco mais aliviado, passando de 1500kN para 1700kN.

7.2.4 Observações baseadas na primeira etapa de estudo de casos

Foram realizadas algumas execuções do procedimento de síntese para cada caso, de forma a investigar se o resultado apontado não representa uma região de ótimo local. Todas as execuções foram realizadas com uma população de 30 indivíduos e admitindo um número máximo de gerações também igual a 30.

Foi adotado o critério de parada "Média-Melhor", onde o processo é interrompido quando por ***N*** gerações a média das avaliações da população se mantiver maior ou igual a um percentual ***K*** do valor de avaliação do melhor indivíduo. Nas execuções da primeira etapa foram adotados ***N***=5 e ***K***=0.95.

Em nenhuma das execuções realizadas foi necessário efetuar o número máximo de gerações pré-definido, ou seja, todas terminaram com a satisfação do critério de parada utilizado.

Na Figura 46 disposta a seguir são mostradas as curvas de convergência para cada caso, onde podemos observar que na geração de parada do algoritmo, o critério de parada é atingido e ainda toda a população da última geração converge para o mesmo ponto.

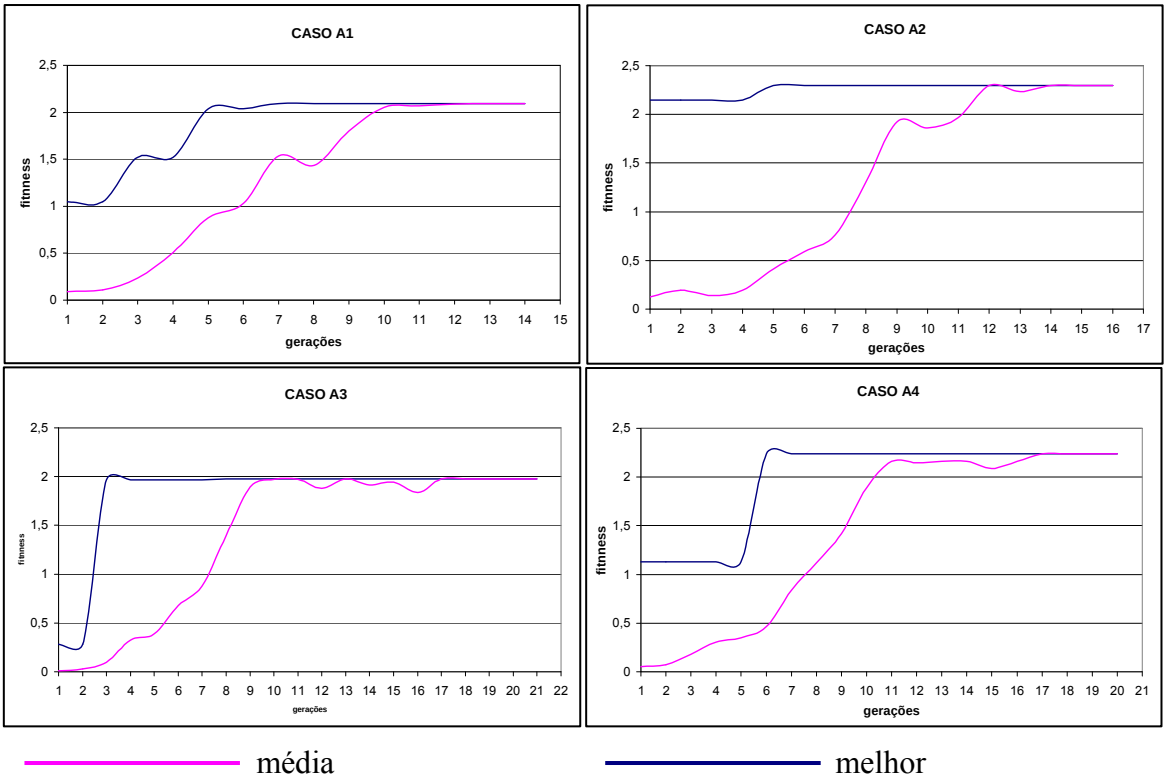


Figura 46: Gráficos de convergência de cada caso

A figura 47 ilustra as configurações indicadas para cada caso de estudo da primeira etapa como sendo a solução ótima do problema, através de um esquema geométrico simplificado.

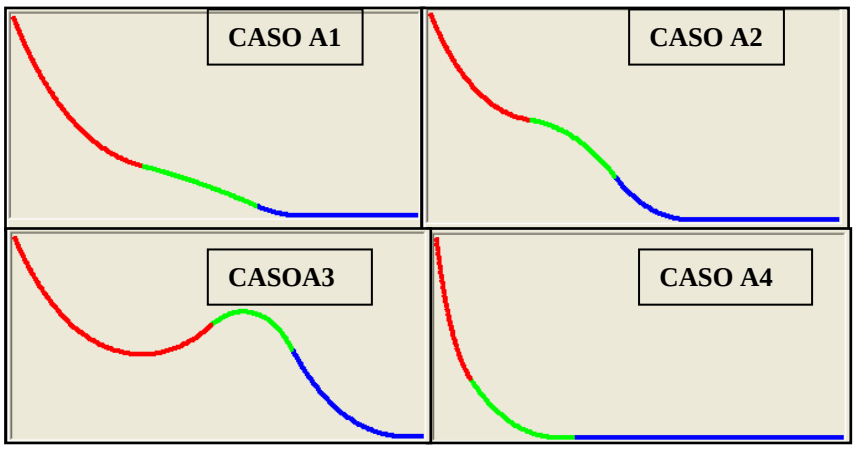


Figura 47: Variáveis de Projeto e Restrições - Caso IV

A tabela a seguir mostra um resumo das configurações alcançadas para os processos de busca de cada caso de estudo.

Tabela 6: Resumo - Melhores indivíduos de cada caso

CASO	L1(m)	L2(m)	L3(m)	$ø_{Bun}$(m)	L_{Bun}(m)	Esp_{Bun}(m)	fitness
A1	809	622	1197	-	-	-	2,09
A2	1206	565	857	1,3	0,7	2,0	2,30
A3	923	540	1452	1,4	1,4	1,9	1,98
A4	1329	673	932	0,7	1,0	1,9	2,24

Para entender o comportamento de cada caso de estudo é necessário recorrer a uma análise mais criteriosa das condições de contorno de cada problema, ou seja, através seus limites de busca e suas restrições de comportamento.

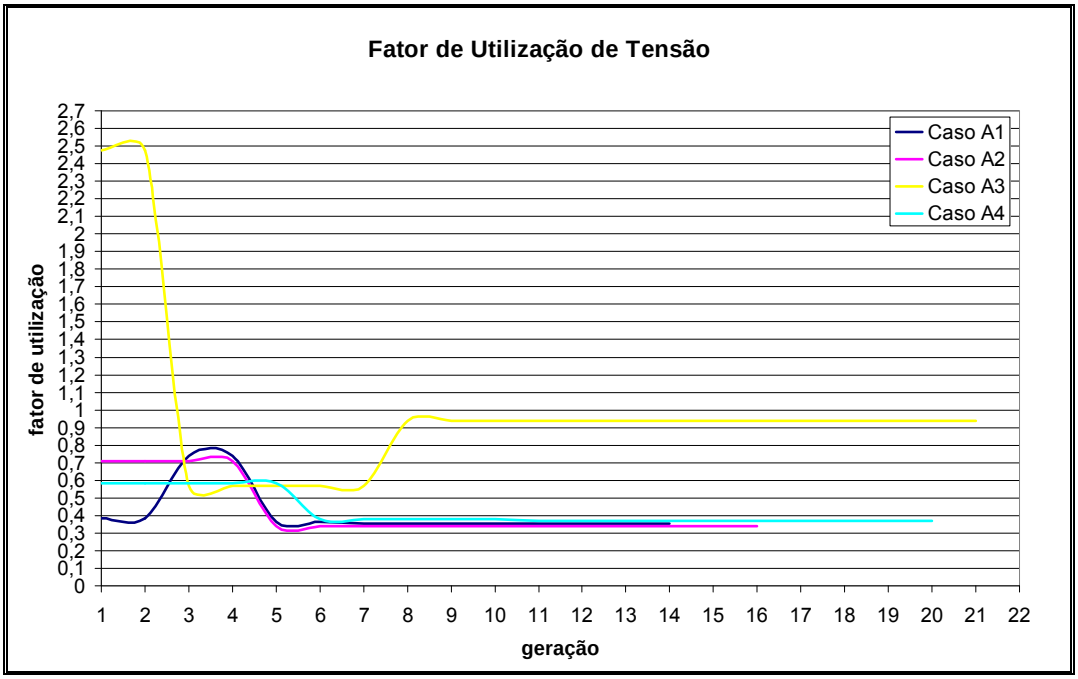


Figura 48: Fator de utilização de Tensão de Von Mises

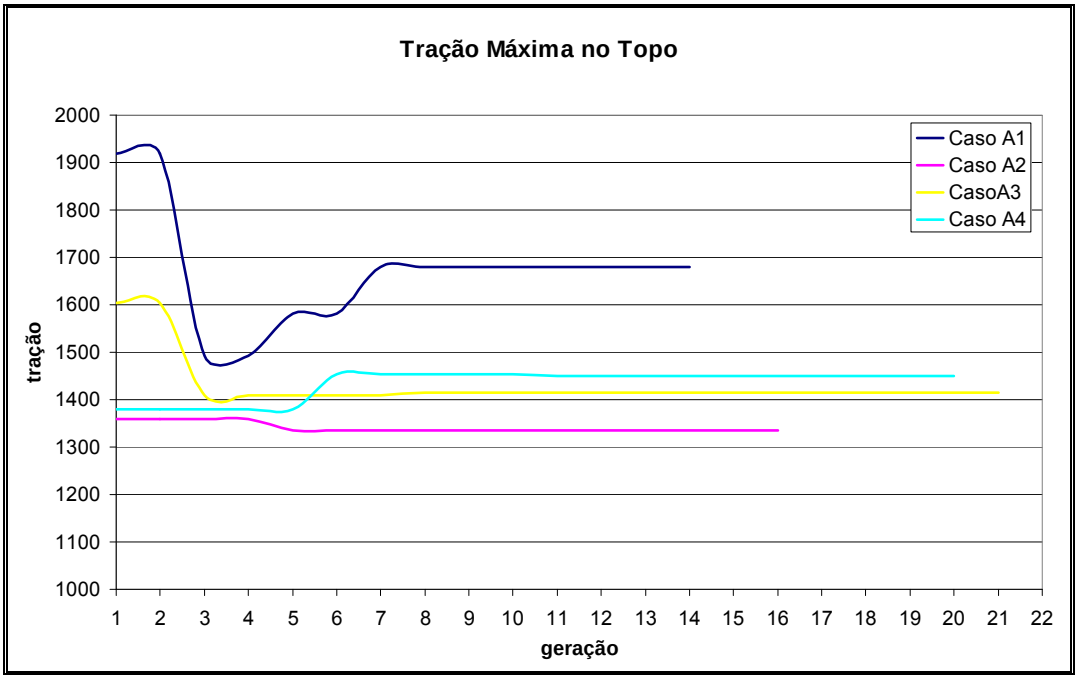


Figura 49: Tração Máxima no topo

Observando para cada caso a evolução do fator de utilização de tensões de Von Mises e da tração máxima de topo do melhor indivíduo ao longo das gerações, mostrados nas Figuras 48 e 49, é possível fazer a seguir fazer as seguintes observações.

- Sem a variação dos parâmetros dos flutuadores, ou seja, no Caso A1, é possível perceber que a configuração somente se consegue direcionar a busca para a região do ótimo elevando muito o nível de tração no topo.
- Admitindo para a mesma configuração base a variação dos parâmetros do flutuador, ou seja, no Caso A2, notamos que o mesmo nível de tensões de Von Mises foi alcançado, mas agora com níveis muito menores de tração de topo. Nota-se ainda um aumento na função de avaliação de cerca de 10%.
- Inserida a restrição de tração no topo, ou seja, no Caso A3, o algoritmo é forçado a dar mais flutuabilidade ao trecho intermediário e lançar mão de maiores comprimentos para alcançar simultaneamente bons níveis de tração no topo e de tensão de Von Mises. Em virtude deste comportamento a função de avaliação cai cerca de 14%.
- O Caso A4 é bastante interessante contudo não inesperado, pois ao se relaxar a restrição de tração no topo a região ótima se desloca certamente nesta direção, levando a configuração a manter a geometria aproximada de uma configuração tipo *free-hanging*.

Considera-se então que a ferramenta desenvolvida é capaz de traduzir corretamente o comportamento físico do problema descrito de forma a criando condições para tomada de decisão numa fase inicial de projetos de sistemas de riser.

7.3 Segunda etapa de estudo de casos.

De forma a ilustrar a possibilidade de utilização da metodologia para estudar a aplicação de materiais diferentes ao longo de um riser e ainda a possibilidade de estudar configurações sem flutuadores, na segunda etapa são analisados 2 casos de estudo, B1 e B2, dando agora ênfase a utilização de configurações *Free-Hanging* compostas por um ou mais materiais. Os dados do modelo básico são mostrados a seguir, bem como a descrição de cada Caso de Estudo.

Como a objetivo desta etapa de estudo de casos não é focado no desempenho computacional, mas sim em exemplificar mais uma possibilidade de aplicação da ferramenta, novamente são empregandas apenas análises estáticas; assim, é importante ter este fato em mente ao interpretar os resultados obtidos pelo procedimento de síntese e otimização.

7.3.1 Dados gerais e dados ambientais

Tratando-se agora de um estudo de configurações free-hanging, não são mais necessários dados de flutuador, e o restante dos dados gerais são mantidos iguais aos da primeira etapa. É apenas necessário acrescentar os dados do seguimento de titânio que compõe um dos casos agora estudados, conforme a tabela a seguir.

Tabela 7: Dados para trecho de titânio usado na segunda etapa de estudo de casos

Dados do duto de titânio	
Peso Específico	43,46 kN/m³
Tensão de escoamento	560 MPa
Módulo de elasticidade	114 GPa

Foi considerada a mesma correnteza atuante na primeira etapa, contudo ela agora é combinada com os seguintes apenas com os casos NEAR e FAR mostrados na figura 43.

7.3.2 Descrição dos casos em estudo da segunda etapa

O primeiro caso estudado nesta etapa, ***Caso B1***, é bastante simples, utilizando apenas um único segmento de aço com malha constante de 5 metros de comprimento. As características da configuração básica que serve de banco de dados para as gerações do algoritmo de síntese e otimização estão dispostas na figura 50.

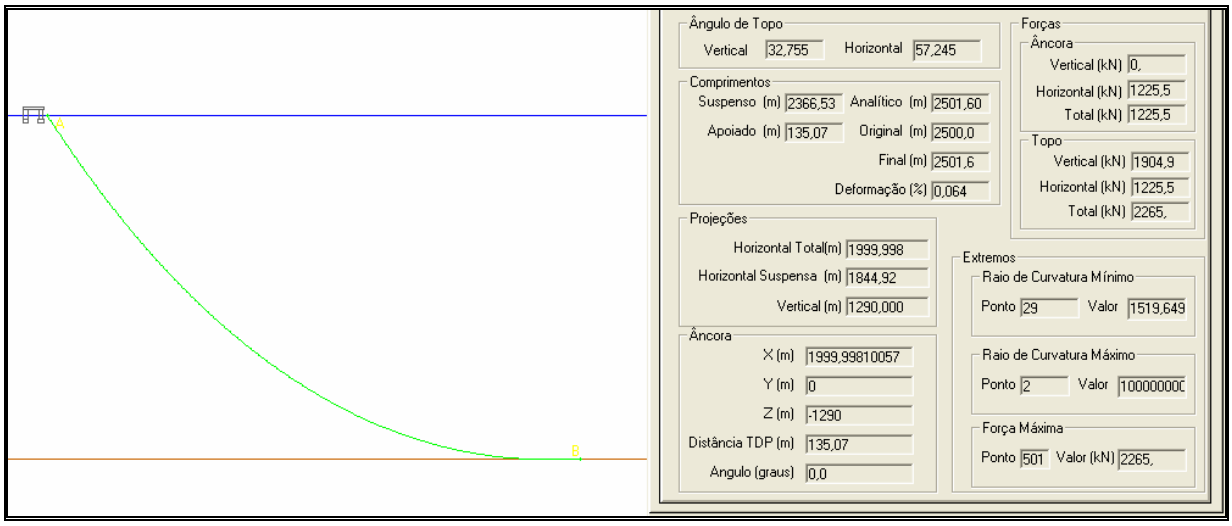


Figura 50: Dados Básicos do caso de estudo B1

Podemos notar que esta é uma configuração completamente descompromissada com a funcionalidade, pois como dito anteriormente pretende apenas compor o banco de dados básico para o algoritmo, sendo usada como ponto de partida para o processo de síntese e otimização.

Os limites de busca utilizados para este caso foram de 2500m (comprimento original da com figuração base) até 4000m com precisão de 5m. As restrições aplicadas ao comportamento das configurações são vistas na Figura 51

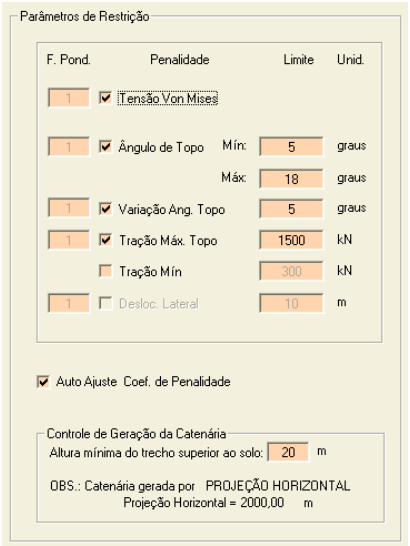


Figura 51: Restrições e parâmetros de evolução aplicados ao caso de estudo 1

O segundo caso de estudos desta etapa , ***Caso B2***, parte de uma configuração básica parecida com a do caso B1, mas agora composta por 4 segmentos intercalando titânio e aço a partir do topo da configuração, como ilustra a Figura 52.

Segmento	Nome	Comp Seg(m)	Malha Ini(m)	Malha Fin(m)	Nome do Tipo
1 A	Tit - topo	625,	5	5	TITANIUM(S)
2	Aço - topo	625,	5	5	Scr - d = 021808m(S)
3	Tit-TDP	625,	5	5	TITANIUM(S)
4 B	Aço - fundo	625,	5	5	Scr - d = 021808m(S)

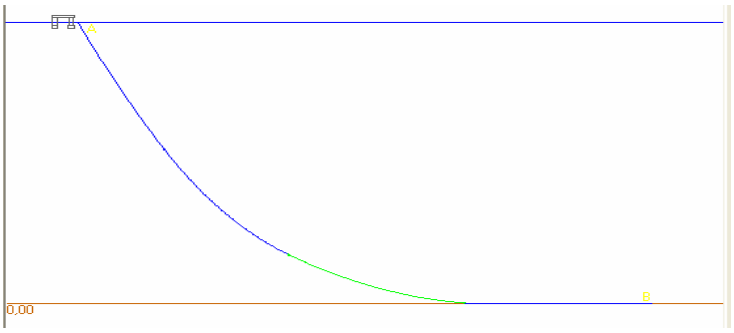


Figura 52: Configuração básica - caso 2

Como podemos notar na figura acima, não houve a preocupação de posicionar corretamente os trechos de titânio, e nem tampouco definir comprimentos racionais para estes, apenas a seqüência de segmentos é coerente com o interesse da aplicação. Espera-se que uma boa distribuição surja do resultado do processo de síntese e otimização.

Os parâmetros de evolução do algoritmo foram mantidos os mesmos do caso anterior, contudo foram definidas novas restrições aplicadas ao comportamento das configurações e novos limites de busca que podem ser vistos nas Figuras 53 e 54.

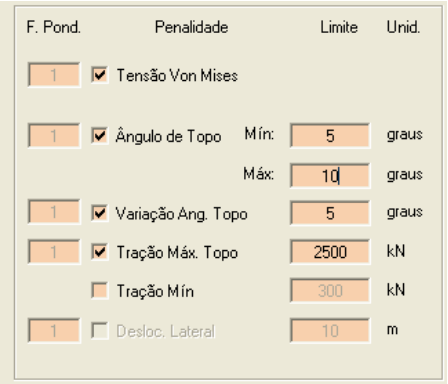


Figura 53: Critérios de Restrição – Caso B2

Seg	MIN(m)	MAX(m)	Precisão	Ind. Custo	Melhor	Ativa	Flutua
1 A	50	650	1	1000		X	
2	650	2000	1	1		X	
3	50	650	1	1000		X	
4 B	650	2000	1	1		X	

Figura 54: Limites de Busca – Caso B2

7.3.3 Observações baseadas na segunda etapa de estudo de casos

Ao caso B1 foi atribuído um número máximo de gerações de 50, um tamanho de população de 20 indivíduos e taxas de cruzamento e mutação de 0.8 e 0.1 respectivamente.

Este caso de estudo convergiu conforme a Figura 55, chegando a uma configuração que não viola nenhum dos critérios de projeto com 14 gerações e com o histórico de análises descrito pela Tabela 8.

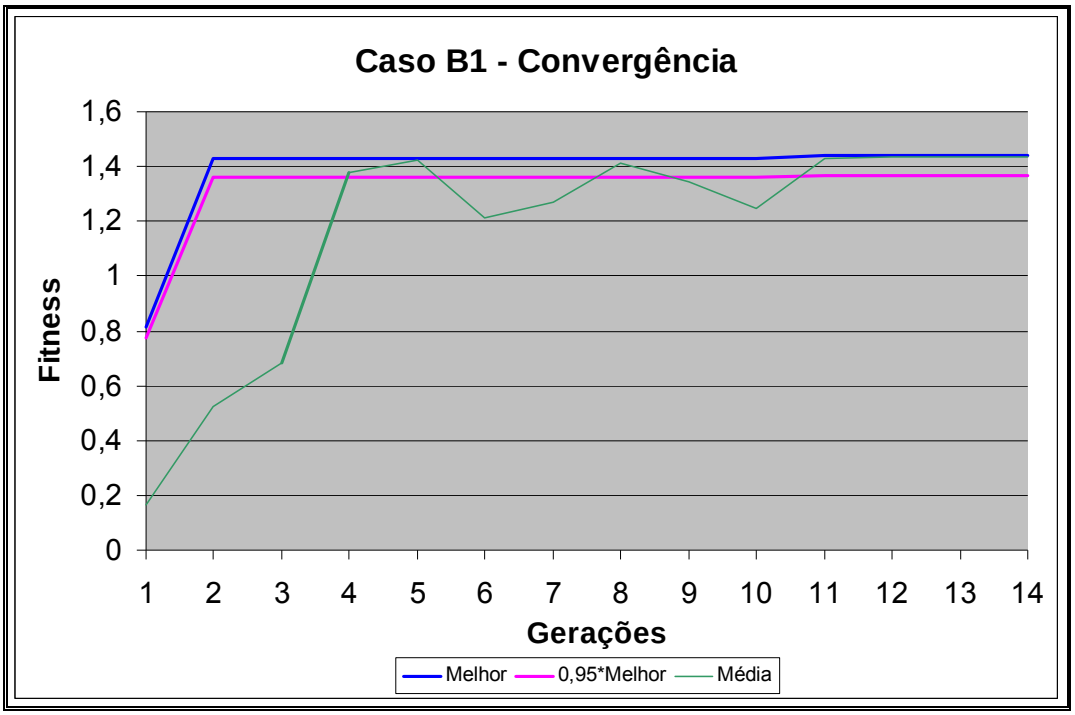


Figura 55: Caso B1 - Convergência

Tabela 8: Histórico de análises do caso B1

Ger	*Num. Aval*	*Não Conv*	*Ind.Penalizados*
1	20	15	5
2	16	9	7
3	15	5	10
4	11	0	11
5	9	1	8
6	9	4	5
7	7	2	5
8	3	0	3
9	4	1	3
10	5	2	3
11	2	0	2
12	2	0	2
13	3	0	3
14	3	1	2
Σ	109	40	69

As características da configuração sugerida pelo algoritmo são mostradas na Figura 56, onde podemos destacar o ângulo de topo de aproximadamente 14º, a tração no topo total de 1380kN e o novos comprimentos apoiado de 1120m e total de 2780m aproximadamente.

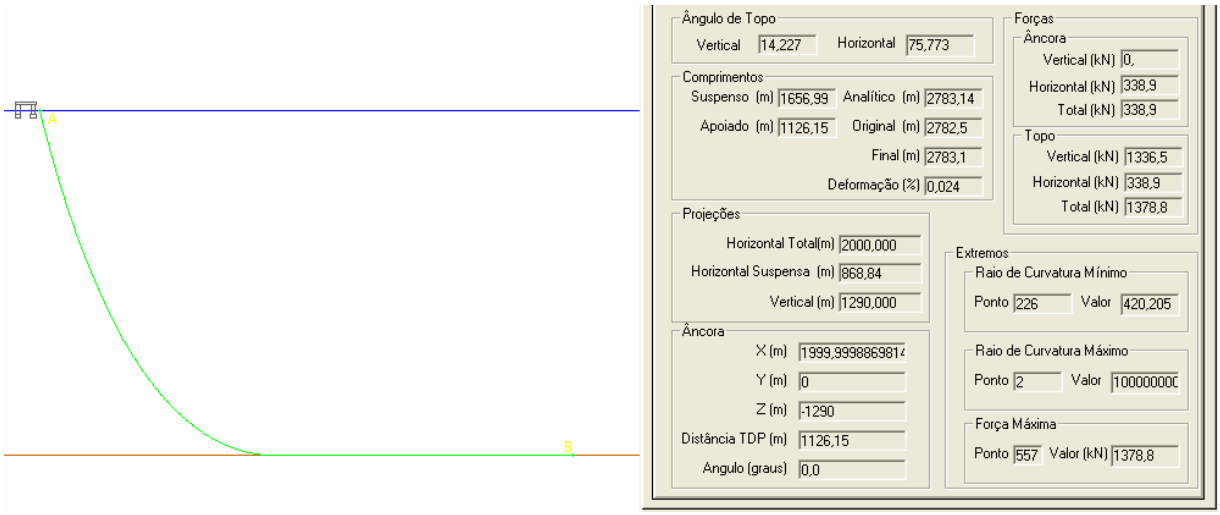


Figura 56: Características da configuração sugerida para o caso 1

O fator de tensão de Von Mises da configuração sugerida ficou em torno de 30% da tensão admissível, ou seja, já considerada a aplicação dos fatores Ca e Cf na tensão de escoamento do material. Novamente, é importante ter em mente que neste procedimento as avaliações empregaram apenas análises estáticas.

O caso de estudo B2 convergiu conforme a Figura 57, chegando a uma configuração que não viola nenhum dos critérios de projeto com 19 gerações e com o histórico de análises descrito pela Tabela 9.

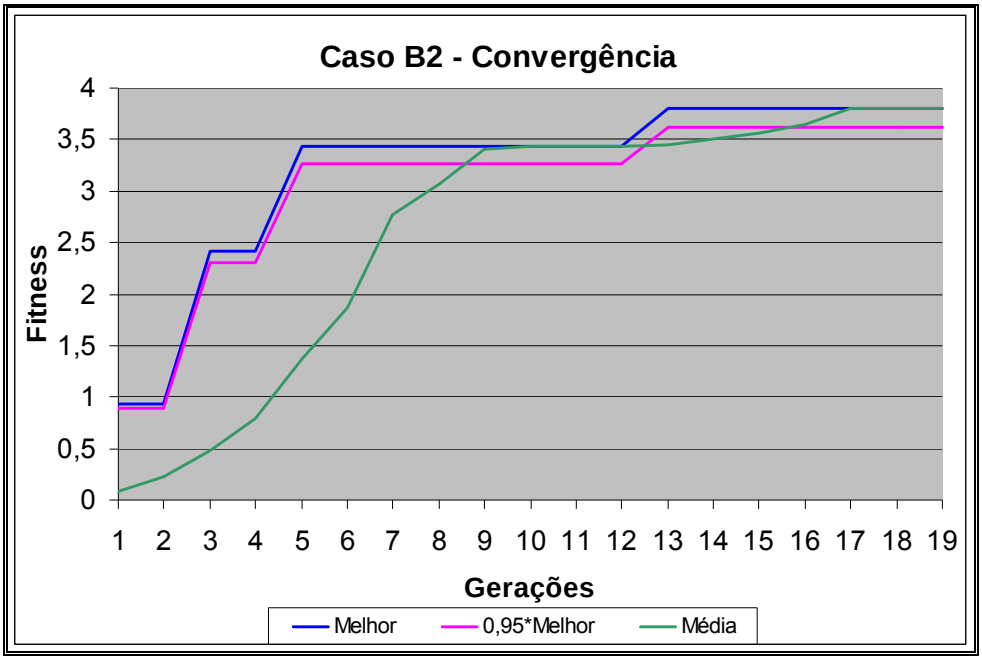


Figura 57: Caso B2 – Convergência

Tabela 9: Histórico de análises do caso B2

Ger	*Num. Aval*	*Não Conv*	*Ind. Pen*
1	***20***	***18***	***2***
2	***17***	***14***	***3***
3	***15***	***8***	***7***
4	***13***	***4***	***9***
5	***7***	***0***	***7***
6	***8***	***0***	***8***
7	***6***	***0***	***6***
8	***4***	***0***	***4***
9	***3***	***0***	***3***
10	***1***	***0***	***1***
11	***2***	***0***	***2***
12	***2***	***0***	***2***
13	***5***	***0***	***5***
14	***7***	***1***	***6***
15	***7***	***0***	***7***
16	***7***	***0***	***7***
17	***5***	***0***	***5***
18	***5***	***1***	***4***
19	***4***	***0***	***4***
Σ	***138***	***46***	***92***

Os comprimentos dos segmentos sugeridos pelo algoritmo podem ser vistos na Figura 58 e as demais características da configuração são mostradas na Figura 59, onde podemos destacar o ângulo de topo de aproximadamente 10º, a tração no topo total de 1100kN e o novos comprimentos apoiado de 1426m e total de 2919m.

Segmento	Nome	Comp Seg(m)	Malha Ini(m)	Malha Fin(m)	Nome do Tipo
1 A	Tit - topo	288,	5	5	TITANIUM(S)
2	Aço - topo	1835,	5	5	ScrAuto03(S)
3	Tit-TDP	51,	5	5	TITANIUM(S)
4 B	Aço - fundo	745,	5	5	ScrAuto03(S)

Figura 58: Caso B2 - Comprimentos sugeridos pelo algoritmo

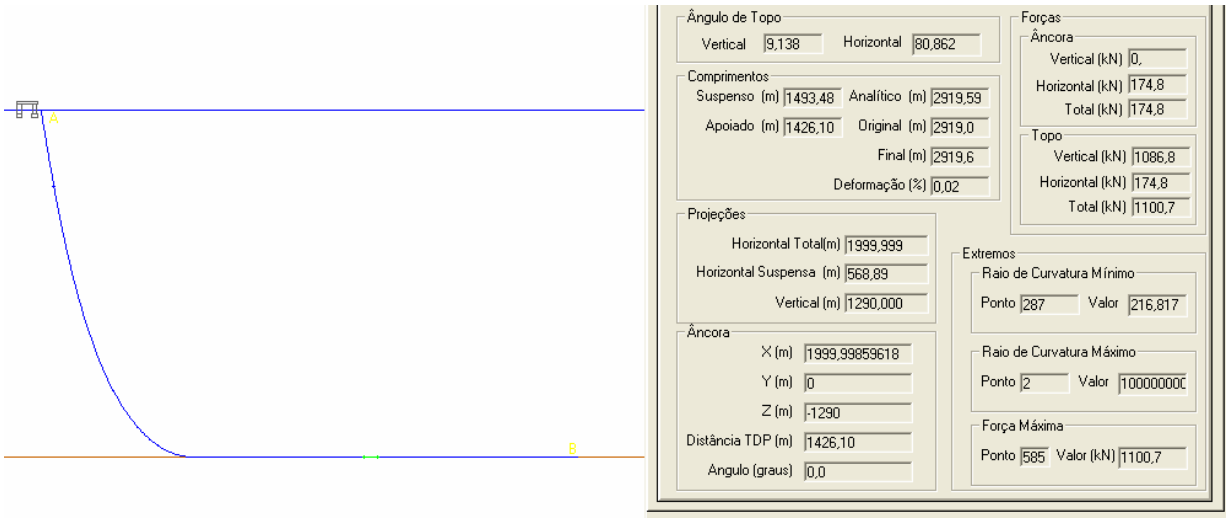


Figura 59: Características da configuração sugerida para o caso 2

O fator de tensão de tensão de Von Mises da configuração sugerida ficou em torno de 35% da tensão admissível. Podemos observar que o trecho de titânio destinado ao TDP (touchdown point) teve o comprimento mínimo definido para ele, e não ficou posicionado na região correta. Este resultado sugere que a aplicação deste material no trecho de TDP não seria indicada neste caso, e este segmento pode ser excluído.

7.4 Terceira etapa de estudo de casos.

Com o objetivo de alcançar todo o potencial da ferramenta desenvolvida, a última etapa do estudo contempla dois casos, ***C1*** e ***C2***, capazes de explorar todo grau de complexidade do problema físico proposto e ainda mostram o limite de desempenho computacional disponível na mesma.

Na verdade os dois casos tratam da mesma configuração básica do tipo lazy-wave, submetida aos mesmos carregamentos ambientais, agora incluindo análises dinâmicas. Nas análises dos dois casos são adotados os mesmos parâmetros de configuração do algoritmo de otimização e de restrições e comportamento. O que diferencia os dois casos é apenas o tipo de processamento adotado. O Caso C1 executa o algoritmo no modo de processamento seqüencial enquanto o Caso C2 utiliza o modo de processamento paralelo disponível na ferramenta.

É importante dizer que tanto utilizando o processamento seqüencial quanto o paralelo, a seqüência de verificações mostrada a seguir é assumida de maneira a evitar o gasto de tempo de processamento com configurações não convergentes.

1. Toda configuração candidata gerada pelo algoritmo de síntese deve admitir uma solução pela equação de catenária, que será utilizada apenas na geração da malha de elementos finitos, para posteriormente ser analisada estaticamente.
2. Caso não se consiga calcular a solução da configuração gerada equação da catenária, automaticamente é atribuída a ela a avaliação nula e nenhum outro processamento é realizado.
3. As configurações que admitem solução pela equação da catenária devem ser submetidas aos procedimentos de análise estática, que caso não tenham a convergência alcançada, resultam numa avaliação nula e nenhum outro processamento é realizado posteriormente.
4. Aquelas configurações que mesmo admitindo a convergência nas duas primeiras etapas não atinjam a convergência durante as análises dinâmicas, ainda receberão avaliação nula.

Vale lembrar que cada configuração é analisada considerando o procedimento descrito acima para cada caso da matriz de casos de carregamento. Assim, se para um único caso da matriz de casos de carregamento a configuração não admitir convergência, ela receberá avaliação nula e os casos restantes da matriz não serão avaliados.

7.4.1 Dados gerais e dados ambientais

O riser analisado nos casos de estudo ***C1*** e ***C2*** é constituído do mesmo material e a mesma seção transversal do riser tratado nos casos de estudo da primeira etapa, contudo agora existem duas combinações de corrente e onda atuando nas direções de offset mostradas na figura abaixo (***Carreg. 1 e 2***).

IND	Carreg.	Offset (%LDA)	Direção de offset	Ca x Cf
1	1	5	Near	0,8
2	2	5	Far	0,8

Figura 60: Matriz de casos de carregamento para a terceira etapa de estudo de casos

As correntezas que compões os casos de carregamento 1 e 2 estão mostradas nas tabelas a seguir:

Tabela 10: Correnteza aplicada ao caso de carregamento (1) da terceira etapa de estudo de casos

Profundidade (m)	Vel. (m/s)	Azimute (DIR)	Azimute (graus)	Direção de ataque (graus)
0	1.14	E	90	0
100	0.99	E	90	0
350	1.02	N	0	90
500	1.14	N	0	90
100	0.65	N	0	90
1250	0.52	N	0	90
1500	0.38	N-NE	22.5	67.5

Notas:

1. a convenção de sentido é “indo para”.
2. azimutes são medidos a partir do norte e no sentido horário.
3. direções de ataque são medidas a partir do eixo x global e no sentido anti-horário.

Tabela 11: Correnteza aplicada ao caso de carregamento (2) da terceira etapa de estudo de casos

Profundidade (m)	Vel. (m/s)	Azimute (DIR)	Azimute (graus)	Direção de ataque (graus)
0	1.09	W	270	180
100	0.64	W	270	180
350	0.56	N	0	90
500	0.77	N	0	90
100	0.73	N	0	90
1250	0.56	N	0	90
1500	0.48	N	0	90

Notas:

1. a convenção de sentido é “indo para”.
2. azimutes são medidos a partir do norte e no sentido horário.
4. direções de ataque são medidas a partir do eixo x global e no sentido anti-horário.

O carregamento de onda considera ondas regulares, com parâmetros que podem ser observados Figuras 61 e 62. O movimento da plataforma devido à onda é aplicado no topo do riser através da interpolação em um RAO fictício, porém de grandezas consistentes com uma embarcação do tipo FPSO.

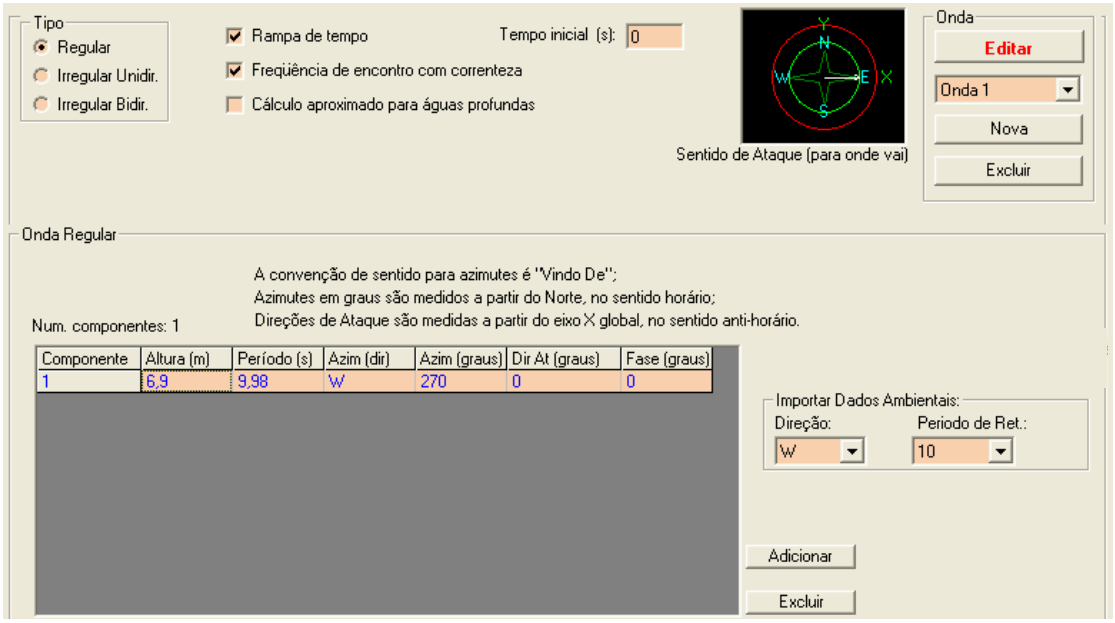


Figura 61: Definição da onda para o caso de carregamento (1) da terceira etapa de estudo de casos

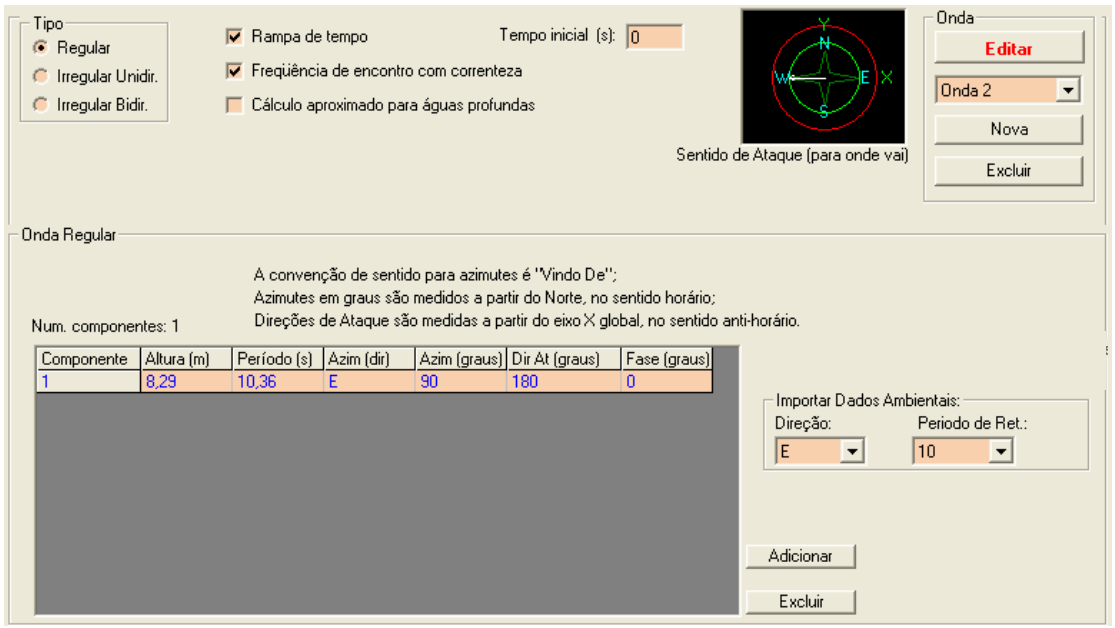


Figura 62: Definição da onda para o caso de carregamento (2) da terceira etapa de estudo de casos

7.4.2 Descrição dos casos em estudo da terceira etapa

Os limites de busca para as variáveis de projeto e os parâmetros de restrição de comportamento utilizados nos casos de estudo C1 e C2 estão mostrados na figura 63.

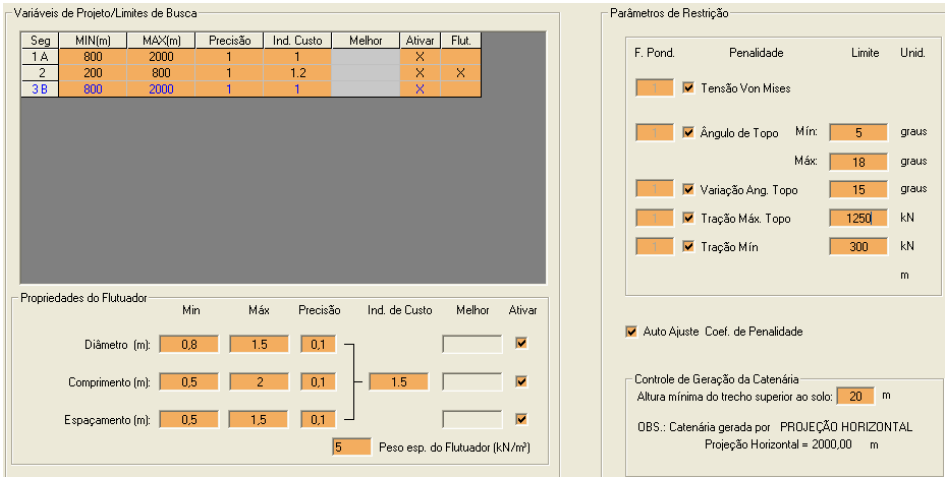


Figura 63: Variáveis de projeto do e parâmetros de restrição para a terceira etapa de estudo de casos

7.4.3 Parâmetros de Síntese e Otimização

Os ajustes dos parâmetros de síntese e otimização adotados para os casos de estudo da terceira etapa são mostrados na Figura 64.

Vale observar que o número de indivíduos da população está intimamente ligado a diversidade do estudo e a capacidade de exploração do algoritmo. Por outro lado, fazendo o uso de análises dinâmicas, o custo computacional associado às avaliações da função objetivo cresce significativamente. Isto poderia sugerir a redução do número de indivíduos, para evitar custo computacional excessivo, contudo este aumento de custo pode ser compensado por um esperado aumento no número de indivíduos que não tenham a análise concluída por problemas de convergência.

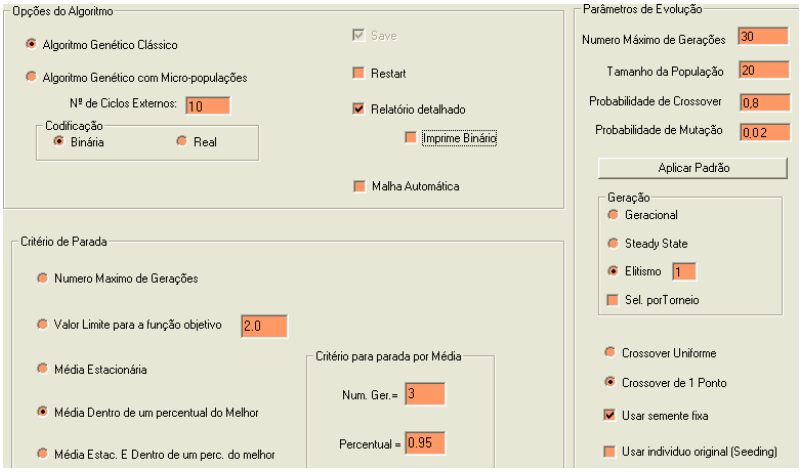


Figura 64: Parâmetros de síntese e otimização para a terceira etapa de estudo de casos

7.4.4 Observações baseadas na terceira etapa de estudo de casos

De forma a verificar a consistência da estratégia de paralelização do código de síntese e otimização, a geração dos indivíduos nos dois casos da última etapa de estudos foi propositalmente realizada com a mesma semente aleatória. Desta forma, é possível garantir que a primeira geração de soluções candidatas, que é gerada aleatoriamente, será a mesma nos dois casos.

Os resultados alcançados para os dois casos desta etapa foram os mesmos, e estão apresentados a seguir.

O acompanhamento da convergência do procedimento de otimização para os casos C1 e C2 é apresentado na figuras 65.

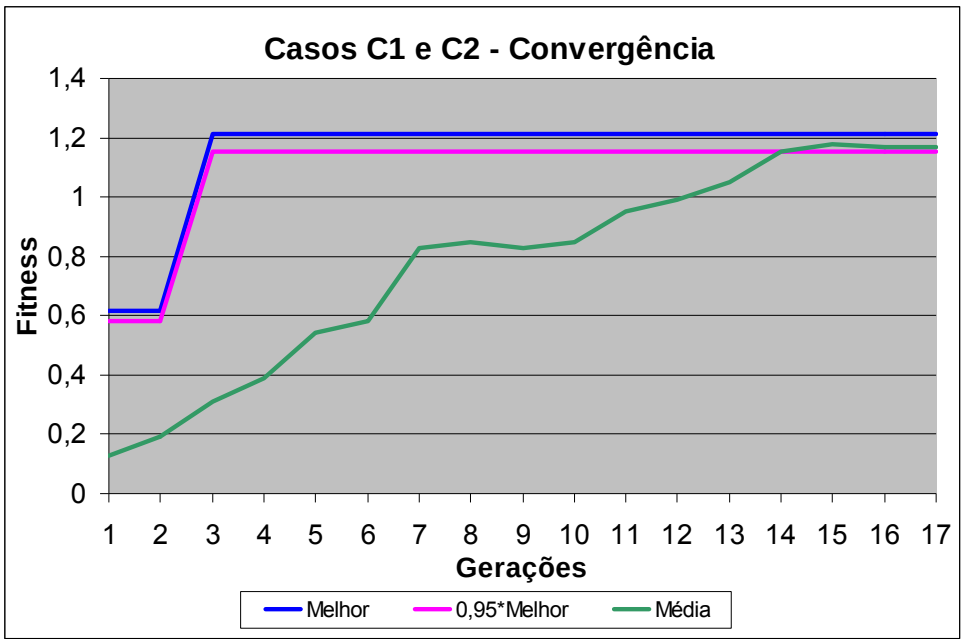


Figura 65: Convergência - Casos C1 e C2

A Tabela 12 indica os valores dos parâmetros da configuração que se obteve como resultado.

Tabela 12: Configurações obtidas – Casos C1 e C2

Limites de Busca			Melhor Configuração (m)
Variável	min(m)	max(m)	Casos C1 e C2
Comprimento superior	800	2000	1418
Comprimento com Flutuadores	200	800	428
Comprimento inferior	800	2000	845
Diâmetro dos Flutuadores	0,8	1,5	1,0
Comprimento dos Flutuadores	0,5	2,0	1,3
Espaçamento dos flutuadores	0,8	2,0	0,9

A melhor configuração indicada pelo algoritmo para os casos ***C1*** e ***C2*** é apresentada na figura 66.

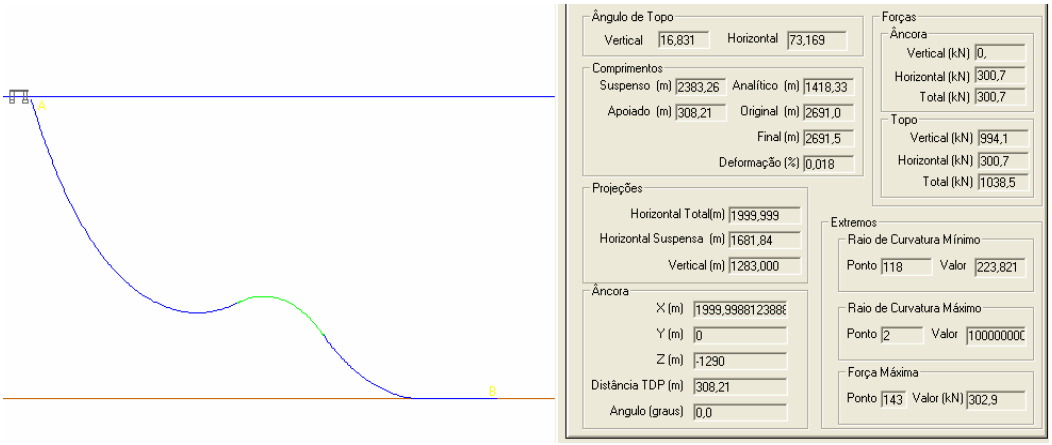


Figura 66: Configuração - Casos C1 e C2

A configuração alcançada para os casos C1 e C2 pode ser considerada satisfatória, pois agrega as melhores características de um modelo Lazy-wave enquanto respeita todos os critérios de projeto, representados pelas restrições, sob todas as condições impostas pela matriz de casos de carregamento.

A estatística das rodadas é mostra a tabela a seguir.

Tabela 13: Parâmetros estatísticos das rodadas dos Casos C1 e C2

Parâmetro Estatístico	Casos C1 e C2
Número total de avaliações da função de avaliação:	142
Número total de indivíduos penalizados:	95
Número total de indivíduos não penalizados:	47
Número total de indivíduos com falha de convergência:	47

Assim, é possível afirmar que a paralelização do algoritmo de síntese e otimização foi executada de forma a não interferir na evolução do algoritmo. Desta forma, a vantagem da implementação em paralelo foi permitir obter uma configuração ótima em um tempo muito menor. De forma a analisar de maneira mais consistente o desempenho da implementação do algoritmo de síntese e otimização em ambiente de computação paralela, seria necessário empregar uma estratégia de medição de tempos de processamento, comunicação e espera ao longo das execuções da ferramenta desenvolvida, para todos os processos envolvidos.

8 Conclusões e Recomendações para Trabalhos Futuros

Através dos casos estudados podemos concluir que a metodologia de busca e otimização de configurações de riser foi implementada com sucesso na ferramenta desenvolvida visto que os resultados alcançados são coerentes do ponto de vista do comportamento físico, apresentando variações esperadas de acordo com as variáveis de busca incluídas e com as restrições consideradas em cada caso.

No entanto, é preciso atentar para o fato de que mesmo tendo validado uma solução para os critérios de restrição adotados, isto não garante de que o mesmo ocorrerá quando a mesma solução for submetida à critérios não contemplados na busca, como por exemplo critérios de fadiga.

O objetivo da metodologia desenvolvida não é o de automatizar a espiral tradicional de projeto, substituindo o trabalho do projetista e sim criar a possibilidade de estabelecer de forma ágil diretrizes para as tomadas de decisão na faze inicial de um projeto. Tomando então este ponto de vista, recomenda-se que em desenvolvimentos futuros sejam incluídos critérios de avaliação mais abrangentes, que possam mesmo que simplificados como, por exemplo, uma avaliação de fadiga simplificada, no domínio da freqüência ou considerações sobre o comportamento de VIV (Vortex Induced Vibration). Até mesmo algumas considerações subjetivas incluídas na avaliação podem ser muito úteis na rotina do projeto.

Acredita-se que substituindo a função de avaliação descrita anteriormente pela equação (18) por um conjunto extenso de regras de lógica nebulosa seja possível manter as considerações relacionadas às restrições impostas devidas ao comportamento estrutural adotadas no presente trabalho a ainda incluir critério de qualificação das configurações de diferentes naturezas. Desta forma os resultados proporcionados pela metodologia proposta poderiam ter seu potencial de aplicabilidade prática na indústria amplificado consideravelmente.

Nos casos de estudo apresentados, algumas vezes o algoritmo precisou ajustar os valores de coeficiente de ponderação de algumas penalidades para conseguir achar soluções não restritas. Este fato indica que a solução do problema físico proposto para este conjunto de variáveis, condições de carregamento e de restrições de comportamento fica certamente bem próximo da fronteira factível do problema.

Apesar do uso do ajuste iterativo dos valores de coeficiente de ponderação das penalidades durante a evolução do algoritmo mostrar-se eficiente, a adoção de estratégias mais completas e subjetivas de avaliação, como indicado anteriormente, pode dificultar a ponderação das restrições e qualificações de comportamento de comportamento. Desta forma, talvez seja necessário atribuir a tarefa de ponderação sobre o comportamento das soluções candidatas a um sistema inteligente, como por exemplo, “redes neurais”.

Para esta tarefa, talvez o sistema de avaliação das configurações deva ser inicialmente uma função simples abrangendo somente as principais características e comportamentos da configuração em questão, conforme foi apresentado no presente trabalho. Desta forma, as primeiras populações geradas pelo algoritmo podem servir de base para treinamento de uma rede neural que irá calibrar os coeficientes de ponderação de um sistema de avaliação de lógica nebulosa.

Uma outra observação importante deve ser feita sobre a acurácia dos resultados produzidos. A fim de tornar as análises mais confiáveis numericamente, a implementação de uma estratégia de adaptação de malha de elementos finitos para cada configuração gerada pelo algoritmo de otimização é recomendada. Assim, será possível além de definir vários materiais por linha também garantir uma modelagem numérica de elementos finitos consistente para todos os casos estudados.

De forma a analisar de maneira mais consistente o desempenho da implementação do algoritmo de síntese e otimização em ambiente de computação paralela, recomenda-se também que seja definida uma estratégia de medição de tempos de processamento, comunicação e espera ao longo das execuções da ferramenta desenvolvida.

Visto que a estratégia de paralelização adotada no desenvolvimento da metodologia não proporciona nenhuma alteração na evolução do algoritmo, pode se disser que ela foi ideal para a implementação inicial, onde era extremamente necessário verificar a consistência dos resultados gerados pelo processo. Contudo, outras estratégias de paralelização típicas para algoritmos genéticos podem ser ainda mais promissoras. Por exemplo, uma estratégia de evolução paralela no modelo de ilhas pode criar novas direções de exploração. Desta forma cada sub-população evoluída separadamente pode resultar em configurações bastante diferentes a ainda assim muito bem sucedidas frente aos critérios de avaliação, sem comprometer o desempenho computacional resultante da paralelização.

Baseando-se então nas observações e recomendações anteriores, é possível confirmar a motivação inicial do estudo, ratificando a utilização com sucesso de metodologias baseadas em estratégias de “soft computing” na síntese e otimização de configurações de sistemas de risers, como alternativas aos tradicionais e demorados estudos paramétricos manuais rotineiramente utilizados nas fases preliminares de projeto, já que pequenas mudanças em algum parâmetro que definem uma configuração podem levá-la a falha.

9 Referências Bibliográficas

[1] Jacob B.P, Lima B.S.L.P., Reyes M.C.T. Estudos paramétricos em configurações alternativas de SCRs conectadas a FPSO's. LAMCSO/COPPE/Universidade Federal do Rio de Janeiro - Petrobras ET-150822, Outubro 1998.

[2] Jacob BP, Lima BSLP, Reyes MCT, Torres ALFL, Mourelle MM, Silva RMC. Alternative Configurations for Steel Catenary Risers for Turret-Moored FPSO's. Proceedings of the 9th International Offshore and Polar Engineering Conference, Brest-France, 1999; II: 234-239.

[3] Jacob BP, Lima BSLP, Reyes MCT. Estudos paramétricos da configuração Lazy-Wave para SCR em FPSO's. LAMCSO/COPPE/ Universidade Federal do Rio de Janeiro - Petrobras PEC-361, Janeiro 2001.

[4] Lagaros ND, Papadrakakis M, Kokossalakis G. Structural Optimization Using Evolutionary Algorithms. Computers and Structures 2002; 80: 571-589.

[5] Michalewics Z, Dasgupta D, Riche RG, Schoenauer M. Evolutionary Algorithms for Constrained Engineering Problems. Computers & Industrial Engineering 1996; 30-4: 851-870.

[6] Chung TS, Li KK, Xie JD, Tang GQ. Multi-objective transmission network planning by a hybrid GA approach with fuzzy decision analysis. International Journal of Electrical Power and Energy Systems 2003; 25-3: 187-192.

[7] Andersson J. Multiobjective Optimization In Engineering Design, PhD thesis, Institute of Technology, Sweden, 2001.

[8] Man, KF, Tang KS, Kwong S. Genetic Algorithms: Concepts and Designs. Springer-Verlag London, 1999.

[9] Holland J. Adaptation in natural and artificial systems, Ann Arbor: The University of Michigan Press, 1975.

[10] Goldberg DE, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, 1989.

[11] Zadeh LA. Fuzzy Sets. Information and Control 1965; 8: 338-353.

[12] Dester WS, Blockley DI. Hazard Engineering. Structural Safety 1994; 16: 3-12.

[13] Shapiro AF. The merging of neural networks, fuzzy logic and genetic algorithms. Insurance: Mathematics and Economics 2002, 31: 115-131.

[15] Lacerda, E. & Carvalho, A. (1999). Introdução aos algoritmos genéticos, Anais do XIX Congresso Nacional da Sociedade Brasileira de Computação, Vol. 2, pp. 51.126.

[**Erro! Indicador não definido.**] Lemonge, A. C. de Castro, Aplicação de Algoritmos Genéticos em Otimização Estrutural, Tese de D.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 1999.

[**Erro! Indicador não definido.**] Silva, A. J. Moreira, Implementação de um Algoritmo Genético utilizando o Modelo de Ilhas, Tese de M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2005.

[**Erro! Indicador não definido.**] Cantú-Paz, E. e Goldberg, D. E., Eficient parallel genetic algorithms : Theory and practice, Technical report, University of Illinois at Urbana-Champaign, 1999.

[16] Haupt, S. E. e Haupt, Randy L., Practical genetic algorithms, John Wiley & Sons, Inc, 2004.

[17] Panagiotis Adamidis, Review of Parallel Genetic Algorithms Bibliography, Technical report, Aristotle University of Thessaloniki, Greece, 1994.

[18] Franco, Luciano Donizetti, Implementação Computacional em Ambiente Paralelo de Memória Distribuída para Análise Acoplada de Sistemas Offshore [Rio de Janeiro] 2004 – Tese de Mestrado.

[19] PACHECO, P. S.,MING, W. C., Introduction to Message Passing Programming: MPI User Guide in FORTRAN. 1997. Disponível em: http://www.hku.hk/cc/sp2/training.html.

[20] MPICH2 User's Guide¤ Version 1.0.3 Mathematics and Computer Science Division Argonne National Laboratory – November, 2005.

[14] Design of Risers for Floating Production Systems (FPSs) and Tension-Leg Platforms (TLPs) – Recommended Practice 2RD – First Edition, June 1998. American Petroleum Institute.

[21] Jacob, B.P, "Programa PROSIM: Simulação Numérica do Comportamento de Unidades Flutuantes Ancoradas, Versão 2.2ª – Manual de Entrada de Dados", COPPE/UFRJ, Programa de Engenharia Civil, Rio de Janeiro, 2004.

[22] Radcliffe, N.J., Formal Analysis and Random Respectful Recombination (1991) In Proceedings of the fourth international conference on genetic algorithms (pp. 222-229). San Mateo, CA: Morgan Kaufmann.

[23] Afonso C. C. Lemonge and Helio J. C. Barbosa, “An adaptive penalty scheme for genetic algorithms in structural optimization” - INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING Int. J. Numer. Meth. Engng 2004; 59:703–736 (DOI: 10.1002/nme.899).

[24] Helio J.C. Barbosa a,*, Afonso C.C. Lemonge, “A new adaptive penalty scheme for genetic algorithms” – Computer Science 2003, Elsevier Inc.

[25] Michalewicz Z, Dasgupta D, Le Riche RG, Schoenauer M. Evolutionary algorithms for constrained engineering problems. Computers and Industrial Engineering 1996; 30(2):851– 870.

[26] Z. Michalewicz, M. Schoenauer, Evolutionary algorithms for constrained parameter optimization problems, Evolutionary Computation 4 (1) (1996) 1–32.

[27] Gordon, S. V. & Whitley, D. (1993). Serial and parallel genetic algorithms as function optimizers, V International Conference on Genetic Algorithms, Morgan Kaufmann, pp. 177.183.

[28] Goodman, E. D., Lin, S. & Punch, W. F. (1994). Coarse-grain parallel genetic algorithms: Categorization and new approach, VI IEEE Symposium on Parallel and Distributed Processing.

[29] Cantú-Paz, E. (1997). A survey of parallel genetic algorithms, Technical report, Computer Science Department and Illinois Genetic Algorithms Laboratory - University of Illinois at Urbana-Champaign.

[30] Cantú-Paz, E. (1998). Topologies, migration rates and multi-population parallel genetic algorithms, Technical report, Department of Computer Science and Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign.

[31] Cantú-Paz, E. (1995). A summary of research on parallel genetic algorithms, Technical report, Computer Science Department and Illinois Genetic Algorithms Laboratory - University of Illinois at Urbana-Champaign.

[32] Wright, A. H. (1990). Genetic algorithms for real parameter optimization., FOGA, pp. 205. 218.

[33] Grama, A., Gupta, A., Karypis, G. & Kumar, V. (2003). Introduction to Parallel Computing, 2nd edn, Addison Wesley.

[34] Gropp, W., Lusk, E. & Skjellum, E. (1994). Using MPI : Portable Parallel Programming with the Message Passing Interface, Cambridge,MA,MIT.

[35] Pacheco, P. (1996). Parallel Programming with MPI, Morgan Kaufmann Publishers.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo