Integração de Ferramentas Computacionais para a Solução de Problemas de Otimização em Engenharia

Tales Lima Fonseca

JUIZ DE FORA

MARÇO, 2016

Integração de Ferramentas Computacionais para a Solução de Problemas de Otimização em Engenharia

Tales Lima Fonseca

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Departamento de Mecânica Aplicada e Computacional

Bacharelado em Ciência da Computação

Orientador: Afonso Celso de Castro Lemonge

Co-orientador: Heder Soares Bernardino

Co-orientadora: Patricia Habib Hallak

JUIZ DE FORA

MARÇO, 2016

Integração de Ferramentas Computacionais para a Solução de Problemas de Otimização em Engenharia

Tales Lima Fonseca

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Afonso Celso de Castro Lemonge
Doutor em Engenharia Civil

Patricia Habib Hallak
Doutora em Engenharia Civil

Carlos Cristiano Hasenclever Borges
Doutor em Engenharia Civil

Leonardo Goliatt da Fonseca
Doutor em Modelagem Computacional

JUIZ DE FORA

04 DE MARÇO, 2016

Mãe, pai, irmão, amor, sem vocês nada disso seria possível.

Resumo

Problemas de otimização são comuns em diversas áreas. Na engenharia, por exemplo, estes estão sendo modelados considerando complexidades que, até então, eram descartadas. Isso, com o intuito de simplificar as simulações durante o processo de busca das soluções ótimas. Os softwares comerciais, boa parte deles, consolidados no mercado passam a ser alternativas atraentes nas análises dos problemas de otimização que demandam análises mais complexas. O uso dos mesmos evita a implementação de códigos computacionais que suportam modelagens complexas como as requeridas em vários problemas de otimização em dinâmica dos fluidos, por exemplo. Pretende-se neste trabalho realizar a integração de ferramentas computacionais para a solução de problemas de otimização em engenharia. Utilizou-se o softwares comercial Abaqus® como ferramenta de simulação computacional e a linguagem de programação Python, além disso, um algoritmo genético será usado como ferramenta de busca juntamente com um método de penalização adaptativa. Os experimentos iniciais são realizados em problemas de otimização comumente encontrados na literatura, mostrando a integração desejada. Com isso, espera-se estudar os problemas de otimização de maior complexidade, que é o objetivo principal deste trabalho.

Palavras-chave: Otimização, Algoritmos Genéticos, Python, Abaqus®.

Abstract

Optimization problems are common in many areas. In engineering, for exemple, these are modeled considering complexities that, until then, were discarded. That, in order to simplify the simulations during the process of searching for optimal solutions. Commercial software, most of them, consolidated in the market become attractive alternatives in the analysis of optimization problems that require more complex analysis. The use thereof prevents the implementation of computer code that support complex modeling as required for various optimization problems in fluid dynamics, for exemple. It is intended in this work realize the integration of computing tools to solve engineering optimization problems. We used the commercial software Abaqus® as computational simulation tool and the Python programming language, besides that, a genetic algorithm is used as a search tool together with an adaptive penalty method. Initial experiments are performed in optimization problems commonly found in the literature, showing the desired integration. Thus, it is expected studying optimization problems of greater complexity, which is the main objective of this work.

Keywords: Optimization, Genetic Algorithms, Python, Abaqus®.

Agradecimentos

Quero agradecer, em primeiro lugar, a Deus, pela força e coragem durante toda esta longa caminhada.

Aos meus pais, Vilma e Jânio, ao meu irmão Douglas e a minha namorada Janaína, que com muito carinho e apoio, não mediram esforços para que eu chegasse até esta etapa de minha vida.

Aos professores Afonso Lemonge, Heder Bernardino e Patricia Hallak em especial pela amizade e principalmente, pela excelente orientação durante a realização deste trabalho.

Aos professores do Departamento de Ciência da Computação e de Mecânica Aplicada e Computacional pelos seus ensinamentos, que durante esses anos, contribuíram de algum modo para o nosso enriquecimento pessoal e profissional.

Aos meus amigos de curso, João Gabriel, João Marcos, Jonata Jefferson, Lucas Berg e Viviane Galvão pela amizade e companheirismo durante o curso. Em especial, gostaria de agradecer à Bárbara Amaral, João Paulo e principalmente Ronaro Picoli, por todas as conversas, apoios, perguntas respondidas e etc.

"– Gostei da capa – disse ele. – Não entre em pânico. Foi a primeira coisa sensata e inteligível que me disseram hoje."

Douglas Adams (O Guia do Mochileiro das Galáxias)

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

AG	Algoritmo Genético
APM	Adaptive Penalty Method
CFD	Computational Fluid Dynamics
DCC	Departamento de Ciência da Computação
MAC	Departamento de Mecânica Aplicada e Computacional
ODB	Output Database
SSD	Solid-State Drive
UFJF	Universidade Federal de Juiz de Fora

1 INTRODUÇÃO

1.1 Apresentação do tema

Engenheiros normalmente buscam criar projetos com desempenho máximo e/ou um custo mínimo dos mesmos. Estes requisitos podem ser alcançados modelando-se estes projetos como problemas de otimização.

Além da experiência do projetista ou conjunto de projetistas, os processos de otimização auxiliam consideravelmente a busca das melhores soluções almejadas para os projetos. Por outro lado, não se deve desprezar a experiência dos projetistas que podem introduzir conhecimento nos processos de otimização, baseados nas suas práticas consolidadas facilitando o processo de busca.

Assim, um projeto otimizado é aquele que apresenta o melhor custo e o melhor desempenho possíveis. A busca das melhores soluções entre tantas combinações possíveis demandaria um processo de tentativa e erro ou busca exaustiva que na sua grande maioria é extremamente custoso podendo demandar um tempo indisponível. Para contornar estes problemas surgiram, então, os algoritmos de otimização. Não pretende-se apresentar neste texto uma classificação extensa dos algoritmos de otimização. Mais adiante será ilustrada uma visão geral das principais técnicas de otimização.

1.2 Problemas de otimização

Os problemas de otimização em engenharia, na sua grande maioria, estão sujeitos à restrições que devem ser atendidas pelas soluções encontradas nos processos de otimização. As restrições atendidas garantem a viabilidade das soluções, não colocando em risco o projeto adotado após a otimização e com a máxima economia possível.

É fácil observar que atender a todas as restrições, desde as mais fáceis até di-

fíceis, com o menor custo possível são aspectos conflitantes na formulação do problema de otimização. Por exemplo, minimizar o volume de uma viga sujeita a deslocamentos e tensões máximas implica em diminuir massa, diminuir custo e diminuir rigidez. Diminuir rigidez implica em aumento de deslocamentos e tensões.

Para ilustrar, na otimização estrutural, estas restrições referem-se tipicamente a deslocamentos máximos, tensões máximas, frequências naturais de vibração mínimas entre outras tantas.

Dentre os diversos problemas encontrados, existem os problemas que possuem uma grande complexidade para alcançar as soluções ótimas ou as soluções próximas das ótimas, devido ao seu alto nível de detalhamento e/ou na dificuldade de resolvê-lo. Em razão da necessidade de considerar cada vez mais as condições reais de um projeto na modelagem do problema de otimização em engenharia, os métodos clássicos de otimização passaram a ter aplicação restrita. Esta complexidade pode demandar funções objetivo e restrições não diferenciáveis limitando, então, a aplicação de vários métodos clássicos de otimização.

Vários algoritmos de otimização dependem da avaliação de um conjunto de soluções candidatas durante a busca e um exemplo são os algoritmos populacionais. Estes, em geral, são algoritmos de ordem zero que não demandam derivadas das funções objetivo e nem das restrições. Dependendo do grau de complexidade do problema a busca pode demandar um alto custo computacional.

Por outro lado, tal complexidade pode desencorajar o desenvolvimento, a implementação e o uso de ferramentas computacionais domésticas demandando-se o uso de ferramentas comerciais consolidados no mercado. Nem sempre são disponibilizados nestas ferramentas comerciais os desejados módulos de otimização o que não impede o seu uso como simulador das soluções candidatas na otimização. Os problemas que serão abordados neste trabalho apresentarão um razoável grau de complexidade que motivará a integração de ferramentas computacionais domésticas e comerciais como será apresentado mais adiante neste texto.

1.3 Justificativa

É de grande importância que um projeto consiga garantir um bom desempenho durante a sua vida útil, independente do seu grau de complexidade. Além disso, a busca por projetos otimizados se torna altamente atraente em razão da forte possibilidade de minimização dos custos dos materiais utilizados sem perda de segurança.

Ressalta-se que as questões ambientais são importantes neste processo nos quesitos referentes à diminuição do uso exagerado de matéria prima cada vez mais escassa.

Ainda existem problemas de otimização encontrados na engenharia com um alto grau de complexidade que até hoje, foram resolvidos através de consideráveis simplificações no modelo ou simplesmente não foram abordados devido a falta de ferramentas capazes de realizar simulações mais próximas de um projeto real.

1.4 Objetivos

Pretende-se neste trabalho investigar a integração de ferramentas computacionais para a solução de problemas complexos de otimização encontrados na engenharia, assim como problemas estruturais e de dinâmica dos fluidos, utilizando o softwares comercial Abaqus® como ferramenta de simulação computacional. Utilizou-se a linguagem de programação Python para a integração e a implementação do algoritmo de otimização, o qual se refere a um Algoritmo Genético.

1.5 Organização do texto

O Capítulo 2 é destinado a revisão bibliográfica, apresentando uma breve classificação dos métodos de otimização, os conceitos sobre a teoria da evolução e uma introdução aos algoritmos genéticos juntamente com trabalhos relevantes encontrados na literatura. Além disso, o uso de simuladores para tratamento de problemas complexos, a formulação

de problema de otimização e, por fim, uma breve descrição sobre tratamento de restrições.

No Capítulo 3 encontram-se informações sobre a metodologia utilizada para todo o desenvolvimento deste trabalho, destacando os problemas de otimização em engenharia e a integração das ferramentas computacionais para otimização, mostrando detalhadamente todo o processo entre o simulador e o algoritmo de otimização implementado. O Capítulo 4 apresenta os experimentos numéricos realizados durante o desenvolvimento deste trabalho e, por fim, as conclusões são apresentadas no Capítulo 5 com propostas de trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

A revisão bibliográfica desta seção focará em uma síntese dos trabalhos de otimização através de uma cronologia que contempla os publicações pioneiras e o estado da arte. Além disso, será feito o levantamento dos algoritmos de otimização, em especial, as metaheurísticas bioinspiradas como os Algoritmos Genéticos. Pretende-se fazer também um levantamento sobre as ferramentas comercias para a simulação dos modelos pretendidos com foco no Software Abaqus® e a sua integração com outros tipos de códigos.

2.1 Uma breve classificação dos métodos de otimização

Não pretende-se neste texto nenhuma tentativa de esgotar a apresentação e discussão de todos métodos de otimização disponíveis para as suas aplicações. Na Figura 2.1, extraída de (Carvalho et al., 2014) e adaptada de (Gandomi et al., 2013), mostra-se uma classificação bastante informativa acerca dos métodos mais usuais e consolidados na literatura. A técnica a ser usada neste trabalho são os Algoritmos Genéticos.

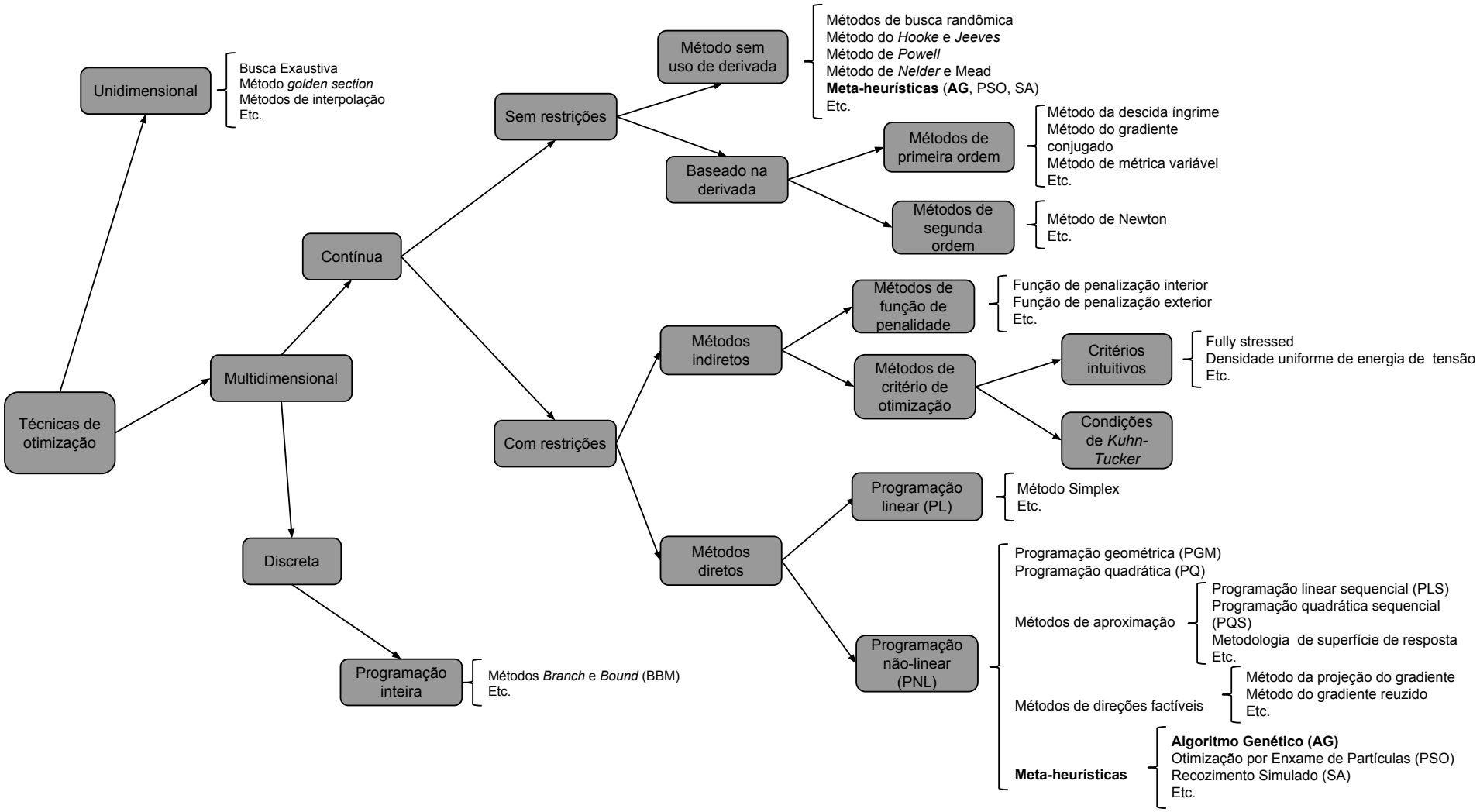


Figura 2.1: Classificação das técnicas de otimização numérica, extraída de (Gandomi et al., 2013) e adaptada de (Carvalho et al., 2014).

2.2 Seleção natural

De acordo com a Teoria da Evolução (Darwin, 1859), a luta pela sobrevivência dos seres vivos é de grande relevância para a tese da seleção natural. Desta forma, qualquer transformação apresentada em um ser vivo tem como o objetivo tentar lhe dar vantagens na sua luta pela sobrevivência. Algumas destas transformações podem ser benéficas e conceder ao indivíduo maiores chances de se manter vivo e reproduzir, ou podem ser transformações indesejáveis causando talvez a sua morte.

Pode-se observar que na natureza, em um cenário de recursos limitados, existe uma competição entre todos seres vivos, acarretando um grande equilíbrio sobre a quantidade de indivíduos que uma região pode conter. Isso nos leva a pensar que nenhum indivíduo pode se reproduzir infinitamente em uma região, pois desta forma a região seria preenchida por indivíduos de uma mesma descendência e com isso não existiria uma seleção natural, contradizendo que os seres lutam para poderem se reproduzir mantendo as suas heranças genéticas. Doenças, predadores, quantidade de recursos são meios de controle populacional e de seleção natural em uma região, onde apenas os seres mais capacitados conseguem sobreviver.

A seleção natural leva a diversas extinções, pois a luta pela sobrevivência tem como objetivo selecionar os indivíduos mais fortes para a sua manutenção na população e diminuindo as chances de sobrevivência dos menos aptos para tal. Logo, pode-se perceber que todos os seres vivos estão em frequente concorrência para sobreviver e qualquer transformação que possa lhe dar vantagens acarreta em maiores chances de se reproduzir e essa seleção deixa a espécie cada vez mais forte, uma vez que a natureza opta apenas em apurar os bons desempenhos.

Toda mudança em um indivíduo é inicialmente local, mas se esta mudança estimular uma melhoria da espécie, em pouco tempo surgiria um pequeno grupo de reprodução que espalharia estas novas características por uma grande região. Isso leva a concluir que pequenas modificações se acumulam caso sejam vantajosas e acabam se realçando de geração em geração, levando esses indivíduos a se adaptarem a qualquer *habitat* caso possuam características que levem a sua sobrevivência. Os descendentes que sofreram

modificações de uma espécie podem se reproduzir e habitar diferentes regiões, levando em conta que pode ocorrer luta por sobrevivência com outros seres que já habitam a região.

Os aspectos relativos à Teoria da Evolução ilustrados aqui foram inspiração a criação dos Algoritmos Genéticos que serão a máquina de busca utilizada para a obtenção das soluções dos problemas de otimização estrutural estudados neste trabalho.

2.3 Algoritmos genéticos

Com o passar dos tempos, a teoria da evolução se transformou nas ideias apresentadas por Darwin sobre a seleção natural junto com a genética, criando então o princípio básico da Genética Populacional, onde a versatilidade entre os indivíduos de uma população é gerada através da recombinação genética dos pais e pelo acontecimento das mutações. Com base nestas informações, foram feitas pesquisas sobre a utilização da teoria da evolução em problemas de otimização, utilizando o processo evolutivo para se encontrar as melhores soluções de um problema e em aprendizado de máquina (Holland, 1975). Foi então definido que cada solução para um determinado problema seria um indivíduo e o conjunto de possíveis soluções seria a população, de forma que as melhores soluções da população possam se "reproduzir", ou seja, sofrem uma combinação gerando uma nova solução para o problema em uma geração seguinte. Logo, os algoritmos genéticos tendem a gerar soluções cada vez melhores a cada geração, convergindo sempre para melhores soluções.

Os algoritmos genéticos foram imaginados por John Holland nos anos 70, na Universidade de Michigan. Ele acreditava na possibilidade de criação de um algoritmo capaz de resolver problemas complexos inspirando-se na teoria da evolução. Ele se concentrou em dois pontos fundamentais: abstrair e explicar rigorosamente os processos adaptativos dos sistemas naturais e fazer implementações computacionais para simular sistemas artificiais mantendo os mecanismos importantes dos sistemas naturais (Goldberg, 1989).

Sem perda de generalidade, os AG's podem ser definidos como procedimentos de busca baseados na genética e seleção natural das espécies. Eles combinam o conceito de sobrevivência artificial por meio de testes de classificação. Os operadores genéticos são

inspirados na natureza formando um mecanismo eficiente de busca. Testes de classificação através da informação das aptidões dos indivíduos na população têm como objetivo identificar as diferenças entre os melhores e os piores indivíduos pertencentes a uma população.

John Holland elencou três aspectos importantes na construção de um AG: o cromossomo, o indivíduo e a população. Ele buscava uma maneira de codificar um cromossomo que representasse um indivíduo e uma população. O alfabeto binário foi a sua primeira representação e é considerada a mais clássica de todas. O alfabeto binário é composto de 0's e 1's representando artificialmente um "código genético".

Um conjunto de indivíduos define uma população e estes são codificados por seus cromossomos e que serão submetidos a avaliações impostas pelo meio em que vivem. Os indivíduos serão classificados por algum critério baseado nos seus desempenhos permitindo a identificação dos mais aptos a sobreviverem e se reproduzirem. Após a classificação, três operações básicas são feitas sobre a população: a seleção, a recombinação e a mutação. Os genitores (pais) são escolhidos pelo operador de seleção em que sobre eles atuaram os operadores de recombinação e mutação. Imaginando-se que os pais são bons e estão entre aqueles que apresentam as melhores aptidões, espera-se que os seus filhos também o sejam, representando boas soluções na próxima geração. Assim, a população evoluirá para uma solução desejada.

Um algoritmo genético pode ser identificado por cinco aspectos fundamentais:

1. uma codificação genética de soluções para o problema;
2. um procedimento para criar uma população inicial de soluções;
3. uma função de avaliação que retorna a aptidão de cada indivíduo;
4. operadores genéticos que manipulam a codificação dos pais durante o processo de reprodução dando origem a novos indivíduos;
5. parâmetros a serem utilizados no algoritmo durante os processos de reprodução e mutação.

E ainda, colocados da seguinte forma (Davis, 1996):

1. Inicializar uma população de cromossomos;
2. Avaliar cada cromossomo na população;
3. Criar novos cromossomos através da troca de material genético entre cromossomos;
4. Remover membros da população para dar lugar a novos cromossomos;
5. Avaliar os novos cromossomos e inseri–los na população;
6. Se o procedimento convergiu, terminar, se não, voltar ao passo 3.

Um pseudo–código que representa um algoritmo AG na sua forma genérica pode ser escrito como:

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

Desta forma, a utilização de algoritmos genéticos para solução de problemas da engenharia se torna uma ferramenta muito atrativa, já que diferente dos métodos clássicos que têm aplicações limitadas, os algoritmos genéticos conseguem identificar soluções globais, podendo convergir para a solução do problema e não para uma solução local. Além disso, eles não possuem domínio de aplicação restrito e pode resolver problemas que envolvam variáveis contínuas e discretas. As funções objetivos não precisam ser contínuas ou diferenciáveis e não é preciso informar um ponto factível para que a procura da solução ocorra. A utilização da programação paralela em algoritmos genéticos é totalmente recomendada. Com isso, pode-se concluir que tem-se uma ótima técnica de otimização

para ser utilizada em problemas encontrados não somente na engenharia estrutural, mas também em diversas áreas.

2.4 Simuladores

Nos algoritmos de otimização faz-se necessário avaliar uma solução candidata para a obtenção da qualidade da mesma, como visto no item anterior, e compará-la com as demais, por exemplo, no algoritmos populacionais. A avaliação das soluções em problemas de otimização encontrados na engenharia nem sempre são triviais. Pode-se demandar uma análise complexa que dependerá de simuladores e, também, demandar um tempo razoável. Estas análises além de retornarem a qualidade da solução em questão informa também se as restrições impostas ao problema são satisfeitas ou não.

Face ao desejável grau de complexidade que torna a simulação de uma solução mais próxima de um projeto real, pode-se demandar o uso de simuladores que suportam modelagens complexas acarretadas por estas análises. Uma possibilidade é a utilização da ferramenta chamada Abaqus®, que é capaz de simular vários problemas da engenharia. Existem vários autores que utilizam simuladores como o Abaqus® para a análise das soluções candidatas em algoritmos de otimização (Johnsen, 2013; Lanes and Greco, 2013), que também é um dos objetivos deste trabalho.

2.5 A formulação do problema de otimização

Um problema de otimização **PO** está sempre associado a um problema de minimização ou maximização de uma ou mais funções. No caso de minimização o **PO** pode ser colocado da seguinte forma:

$$\min f(x) = f(x_1, x_2, x_3, ..., x_n)$$

onde a minimização da função $f(x)$, $x \in \mathbb{R}^n$, é equivalente à maximização da função $g(x)$, onde $g(x) = -f(x)$. Assim, tem-se:

$$\text{mínimo } f(x) = \text{máximo } g(x) = \text{máximo } \{-f(x)\}$$

As funções $f(x)$ e $g(x)$ são conhecidas como *funções objetivo* e representam a quantidade que se deseja minimizar ou maximizar, respectivamente.

O problema de otimização pode ser formulado matematicamente da seguinte forma:

$$\begin{aligned} \text{min/max } f_i(x) \text{ para } i &= 1, \ldots M \\ \text{submetido a } g_i(x) &\leq 0 \text{ para } i = 1, \ldots p \\ h_i(x) &= 0 \text{ para } i = 1, \ldots q \\ x &\in X \subset \mathbb{R}^n \\ X &= \left\{ x \in \mathbb{R}^n : x_i^L \leq x_i \leq x_i^U, \quad i = 1, 2 \ldots n \right\} \end{aligned}$$

O vetor x é designado como o vetor de incógnitas ou vetor de variáveis de projeto, $f_i(x)$ são as funções objetivo e $g_i(x)$ e $h_i(x)$ são as restrições de desigualdade e igualdade, respectivamente, e estas podem ser funções lineares ou não lineares do vetor de variáveis de projeto.

A otimização com restrições é mais complexa e pode requerer estratégias específicas na formulação do problema para que essas sejam satisfeitas. O conjunto X é um paralelepípedo em $\mathbb{R}^n$ definido pelos limites inferior e superior pré-estabelecidos para as variáveis de projeto. Um vetor $x \in X$ que satisfaz a todas essas restrições é chamado de uma solução factível do problema. O conjunto de todas as soluções factíveis é chamado de região *factível*.

A solução do problema de minimização é dita solução ótima e existindo mais de uma solução ótima estas são ditas soluções ótimas alternativas. O conjunto de variáveis de projeto que proporciona o menor valor da função objetivo – no caso de minimização –

entre todas as combinações possíveis entre os valores das variáveis, é chamado de mínimo global. Em geral, é difícil afirmar que tal valor é global devido à possibilidade de existência de vários mínimos locais e, assim, somente um deles será o global. O que pode-se afirmar é que o valor encontrado é mínimo numa vizinhança do espaço de busca.

Um exemplo de minimização de uma função linear com 8 variáveis de projeto e contendo 6 restrições de desigualdade pode ser escrito na forma:

$$\begin{aligned}
&\max f(x) = x_1 + x_2 + x_3 \\
&\text{submetido a} \\
&\qquad -1 + 0.0025\,(x_4 + x_6) \leq 0 \\
&\qquad -1 + 0.0025\,(x_5 + x_7 - x_4) \leq 0 \\
&\qquad -1 + 0.01\,(x_8 - x_5) \leq 0 \\
&\qquad -x_1 x_6 + 833.33252 x_4 + 100 x_1 - 83333.333 \leq 0 \\
&\qquad -x_2 x_7 + 1250 x_5 + x_2 x_4 - 1250 x_4 \leq 0 \\
&\qquad -x_3 x_8 + 1250000 + x_3 x_5 - 2500 x_5 \leq 0
\end{aligned}$$

Com o espaço de busca limitado por:

$$100 \leq x_1 \leq 10000, 1000 \leq x_i \leq 10000\,(i = 2, 3) \text{ e } 10 \leq x_i \leq 1000\,(i = 4, ..., 8)$$

Um outro exemplo de minimização de duas funções quadráticas com 2 variáveis de projeto e contendo 2 restrições de desigualdade pode ser escrito na forma:

$$\begin{aligned}
&\min f_1(x) = 4(x_1^2 + x_2^2) \\
&\min f_2(x) = (x_1 - 5)^2 + (x_2 - 5)^2 \\
&\text{submetido a} \\
&\qquad (x_1 - 5)^2 + x_2^2 - 25 \leq 0 \\
&\qquad -(x_1 - 8)^2 - (x_2 + 3)^2 + 7.7 \leq 0
\end{aligned}$$

Com o espaço de busca limitado por:

$$0 \leq x_1 \leq 5 \text{ e } 0 \leq x_2 \leq 3$$

2.6 Tratamento de restrições

Em geral, problemas de otimização com restrições, comumente encontrados em engenharia, podem demandar dificuldade na busca por soluções viáveis que atendam a todas as restrições.

Existem várias técnicas para o tratamento de restrições e, entre elas, uma bastante utilizada é a que faz uso de funções de penalização. Estas são largamente utilizadas nas metaheurística bioinspiradas que foram criadas para resolver problemas sem restrições. Dessa forma, os problemas com restrições são transformados em problemas sem restrições através da introdução de funções de penalização.

A função aptidão modificada, em um problema sem restrições, pode ser colocada na seguinte forma:

$$F(x) = f(x) + penal(x)$$

Na maioria dos métodos a parcela $penal(x)$ é simplesmente um parâmetro, uma função ou um conjunto de funções de penalização, que diferem entre si de acordo com o método e a maneira em que são aplicados à soluções infactíveis.

Um problema padrão de otimização com restrições em R^n pode ser dado como um problema de minimização de uma dada função objetivo $f(x)$, onde $x \in R^n$ é um vetor de variáveis de projeto/decisão, submetidos a restrições de desigualdades $g_p(x) \geq 0$, $p = 1, 2, \ldots, \bar{p}$ bem como restrições de igualdades $h_q(x) = 0$, $q = 1, 2, \ldots, \bar{q}$. Além disso, as variáveis podem estar submetidas aos limites do espaço de busca $x_i^L \leq x_i \leq x_i^U$ mas, este tipo de restrição já é uma restrição "natural" a ser considerada no algoritmo genético.

Técnicas de penalização podem ser classificadas como *multiplicativas* ou *aditivas*. No caso multiplicativo, um fator positivo $p(v(x), T)$ é introduzido com o objetivo de "amplificar" o valor da função aptidão de um indivíduo infactível em um problema de minimização. Pode-se ter $p(v(x), T) = 1$ para uma solução candidata factível x e $p(v(x), T) > 1$ para uma solução infactível. No caso aditivo, é somado uma valor obtido através do nível de penalização de um indivíduo infactível.

Basicamente, o que é feito é penalizar um indivíduo que não satisfaz alguma

restrição do problema de forma que o indivíduo que violar mais receberá uma penalização maior e o que violar menos recebera uma penalização menor. Por exemplo, em um problema de maximização, os indivíduos inviáveis receberam um valor de forma a diminuir o valor da sua função objetivo e em problemas de minimização, os indivíduos inviáveis receberam um valor de forma a aumentar o valor da sua função objetivo. Estas técnicas podem ser estáticas, dinâmicas ou adaptativas.

Neste trabalho será usada uma técnica de penalização adaptativa que é conhecida como APM - *Adaptive Penalty Method* (Barbosa and Lemonge, 2002). O método consiste em adaptar o valor dos coeficiente de penalização de cada restrição do problema utilizando as informações obtidas através da população. Desta forma, para cada geração do algoritmo genético, é possível penalizar soluções infactíveis de acordo com o conjunto de possíveis soluções atuais, ou seja, de acordo com a população existente na geração. Caso a solução avaliada seja factível, o método retornará exatamente o valor da função objetivo, mas caso contrário, se a solução for infactível, deve-se realizar algumas verificações.

Para avaliar uma solução infactível, deve-se verificar se a sua função objetivo é maior do que a média das função objetivo da população. Se ela for maior, deve-se indicar que a solução infactível tem um valor igual a sua própria função objetivo acrescida das penalizações de cada restrição violada. Por outro lado, caso ela seja menor, deve-se indicar que a solução tem um valor igual à média da função objetivo da população acrescida das penalizações de cada restrição violada. O APM é formulado pelas equações a seguir.

Em (Barbosa and Lemonge, 2002), Barbosa e Lemonge propuseram um método livre de parâmetros a serem definidos pelo usuário. O método adaptativo usa informações da polulação como a média da função objetivo e o nível de violação de cada restrição durante a evolução.

A função aptidão pode ser escrita como:

$$F(x) = \begin{cases} f(x), & \text{se } x \text{ é factível,} \\ \overline{f}(x) + \sum_{j=1}^{m} k_j v_j(x) & \text{caso contrário} \end{cases}$$

onde

$$\overline{f}(x) = \begin{cases} f(x), & \text{se } f(x) > \langle f(x) \rangle, \\ \langle f(x) \rangle & \text{caso contrário} \end{cases} \tag{2.1}$$

a $\langle f(x) \rangle$ e a média da função objetivo na população atual.

O parâmetro de penalização é definido em cada geração por:

$$k_j = |\langle f(x) \rangle| \frac{\langle v_j(x) \rangle}{\sum_{l=1}^{m} [\langle v_l(x) \rangle]^2} \tag{2.2}$$

e $\langle v_l(x) \rangle$ é a violação da l-th restrição média sobre a população atual. Denotando por *pop* o tamanho da população, pode-se escrever:

$$k_j = \frac{|\sum_{i=1}^{pop} f(x^i)|}{\sum_{l=1}^{m} [\sum_{i=1}^{pop} v_l(x^i)]^2} \sum_{i=1}^{pop} v_j(x^i) \tag{2.3}$$

A ideia do método é de que valores de coeficientes de penalização devem ser distribuídos de uma forma em que as restrições mais difíceis de serem satisfeitas devem ter valores relativos de coeficietes de penalização mais altos.

3 INTEGRAÇÃO DAS FERRAMENTAS

O estudo desenvolvido neste trabalho envolve uma abordagem para solução de problemas através de um levantamento de informações teóricas básicas na literatura que serão suficientes para a implementação e solução dos problemas propostos.

O enfoque principal desse trabalho é a utilização de métodos de otimização voltada diretamente para problemas encontrados na engenharia. Desta forma, será utilizado uma biblioteca desenvolvida para linguagem Python de algoritmos evolucionários com o objetivo de utilizar os ingredientes básicos de um algoritmo genético como os operadores de seleção, *crossover* e mutação. A escolha da linguagem Python foi devido a facilidade de integração com o software Abaqus®, visto que o mesmo possibilita execução de *scripts* em Python. Para garantir que o algoritmo genético implementado gere soluções esperadas, foram realizados alguns testes preliminares em problemas de otimização sem restrição, obtendo os resultados esperados.

Os algoritmos genéticos foram desenvolvidos para otimização de problemas sem restrições, tornando necessário a utilização de técnicas de penalização para indivíduos que não satisfazem as restrições. Assim, foi implementado e integrado ao AG um método de penalização adaptativa conhecido como APM para o tratamento de problemas com restrições. Por ser um método adaptativo, o algoritmo não fica dependente de parâmetros definidor pelo usuário. Foram realizados alguns testes preliminares em problemas de otimização com restrição, obtendo os resultados esperados.

Existem diversos problemas na engenharia que necessitam otimizar mais de uma função objetivo simultaneamente, o que traz a necessidade do tratamento dos mesmos. Muitos métodos convertem o problema original com múltiplas funções objetivos em um problema com uma única função objetivo, o que torna possível a utilização do AG. No entanto, não pretende-se neste texto apresentar todos os métodos para o tratamento de problemas multiobjetivos. No Capítulo 4 será utilizado um dos métodos para a solução dos experimentos multiobjetivos.

Como os problemas que serão abordados neste trabalho vão fazer uso de simuladores para obter informações necessárias para sua otimização, foi desenvolvido um script em Python para interligar o algoritmo genético e o simulador Abaqus®. Sendo assim, é possível construir o problema que se deseja otimizar no software Abaqus® e resolvê-lo para cada indivíduo gerado pelo algoritmo genético, obtendo as informações necessárias para informar se o mesmo é viável ou não.

O desenvolvimento de qualquer modelo utilizando o software Abaqus® pode ser feito através de duas maneiras, sendo uma delas utilizando uma interface gráfica e a outra por meio de *script* utilizando suas bibliotecas, implementadas em Linguagem Python. Tudo que é desenvolvido na interface gráfica pode ser feito no *script* e o inverso também é válido. Apesar dessas bibliotecas possuírem métodos que possibilitam a implementação de um modelo, seu uso direto é desmotivador, dado a complexidade e o grande trabalho de desenvolvimento requerido.

Uma forma prática adotada neste trabalho é a utilização da interface gráfica do software para a construção de um modelo base do projeto a ser otimizado, podendo ser salvo como um *script* Python. Nele estará contido toda a sequência de instruções necessárias para a criação do modelo utilizando os métodos encontrados nas bibliotecas.

Através deste *script* é possível identificar o que cada método faz na sequência de instruções, tornando assim mais simples a automatização do *script* de um modelo base para um modelo geral por meio de passagem de parâmetros. Um exemplo simples é mostrado no algoritmo da Figura 3.1, onde a primeira função representa o *script* vindo do software e a segunda a modificação realizada para torná-lo geral.

```
#função que desenha um retângulo no modelo base
desenha_retangulo():
    mdb.models['Model-1'].sketches['__profile__'].rectangle(
        point1=(0.0, 0.0), point2=(350.0, 80.0))

#função que desenha um retângulo com passagem de parâmetros
desenha_retangulo_geral(x1,y1,x2,y2):
    mdb.models['Model-1'].sketches['__profile__'].rectangle(
        point1=(x1, y1), point2=(x2, y2))
```

Figura 3.1: Exemplo de Criação de um Retângulo por *script*.

Desta maneira, pode-se criar um *script* de um determinado problema de otimiza-

ção onde as soluções candidatas do AG são passadas por parâmetro, possibilitando assim a análise de diferentes soluções.

Após a execução do *script* para uma determinada solução candidata, o Abaqus® escreve um arquivo no disco com a extensão ".odb" (*output database*) contendo os dados de saída da simulação, conforme apresentado na Figura 3.2.

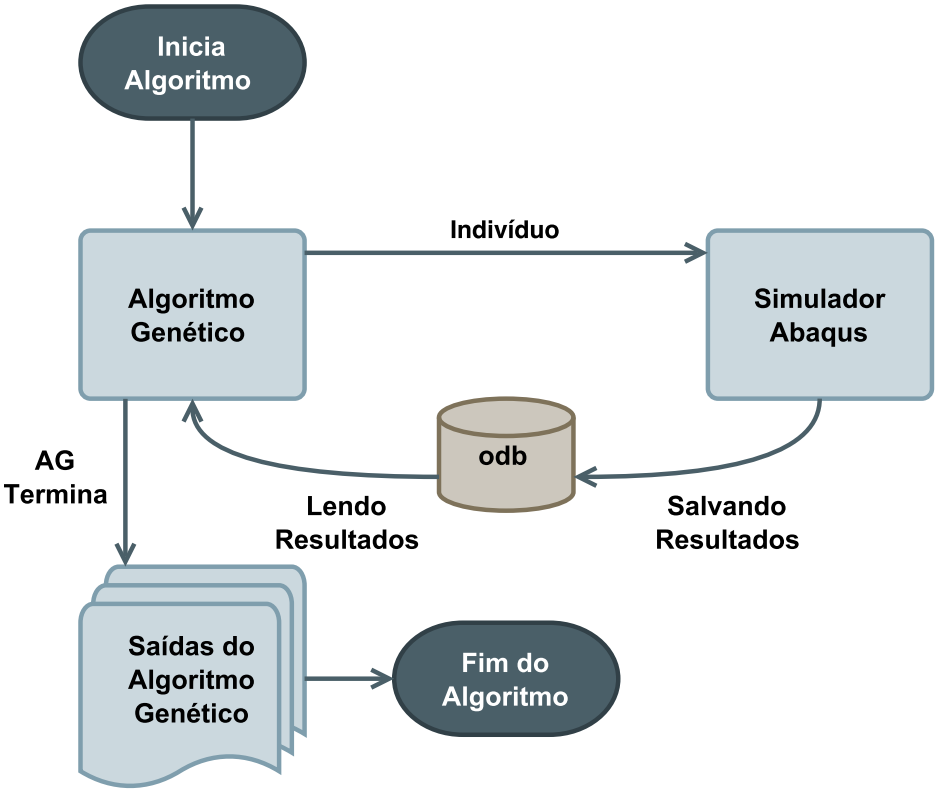


Figura 3.2: Fluxograma da Integração Computacional.

Os arquivos com extensão ".odb" são arquivos de banco de dados associados ao Abaqus®. Faz-se necessário a criação de uma interface capaz de realizar a leitura do mesmo, para que se possa extrair os resultados da simulação. A biblioteca disponibilizada pelo Abaqus® possui métodos que possibilitam o acesso das informações encontradas no arquivo, visto que estes arquivos não pertencem a um formato padrão de banco de dados.

Depois que os valores da simulação são lidos do arquivo ODB, estes são utilizados para a avaliação da solução candidata. Isso se repete para todas as outras soluções candidatas do problema, até que o AG termine todas as suas avaliações.

O Abaqus® é um software comercial para análise por elementos finitos, apresentando uma interface gráfica que é altamente recomendada na construção do problema, mas não recomendada durante o processo de otimização, visto que demandaria um alto

custo computacional desnecessário para renderização dos diferentes indivíduos da população. Desta forma, é possível realizar chamadas ao software Abaqus® sem a necessidade de utilizar a interface gráfica, reduzindo o tempo de cada simulação consideravelmente.

O software Abaqus® não foi desenvolvido para este tipo de abordagem, e por este motivo, durante todo o processo de otimização, é necessário realizar escritas e leituras em disco, o que torna o processo lento, visto que isto ocorre para cada indivíduo da população.

Uma possível estratégia para minimizar cada vez mais o tempo de escrita e leitura em disco é a utilização de um SSD (*Solid-State Drive*) ou até mesmo um disco virtual criado utilizando uma parcela da memória ram. Como a escrita e a leitura são mais rápidas nestes dispositivos, o tempo para avaliar um indivíduo será menor, diminuindo o custo computacional.

Desta forma, neste trabalho foi implementado um ferramental capaz de solucionar diversos problemas encontrados na engenharia, independente da presença de restrições, múltiplas funções objetivos e/ou alta complexidade.

4 EXPERIMENTOS NUMÉRICOS

Para validação da integração das ferramentas computacionais, foram realizados alguns experimentos numéricos envolvendo problemas encontrados na engenharia. Inicialmente foi tratado um problema tradicional da engenharia estrutural, que é a minimização do peso de uma treliça de 10 barras. Além da treliça, foram proposto dois problemas envolvendo uma viga engastada e livre. Os demais problemas tratados referem-se a otimização que envolvem Dinâmica dos Fluídos Computacional (CFD - Computational Fluid Dynamics), como por exemplo, a otimização em trocadores de calor.

4.1 Treliça de 10 barras

O problema de otimização dimensional da treliça de 10 barras (Figura 4.1) consiste em encontrar o conjunto de áreas $A = \{A_1, \ldots, A_{10}\}$ que minimiza o peso $W(A)$ da estrutura, ou seja:

$$min\ W(A) = \sum_{i=1}^{10} \rho A_i L_i$$

sujeito às restrições de deslocamentos dos nós e de tensões normais das barras, respectivamente,

$$\frac{|u_j|}{u_{max}} - 1 \leq 0, \quad \forall\ j = 1, 2, \ldots, m$$

$$\frac{\sigma_i}{\sigma_{max}} - 1 \leq 0, \quad \forall\ i = 1, 2, \ldots, n = 10$$

onde ρ é a densidade do material, L_i é o comprimento do i-ésimo membro da treliça, m é o número de graus de liberdade da treliça e n é o número de barras. A Tabela 4.1 mostra as propriedades do material e os limites admissíveis para os deslocamentos, tensões e limites inferiores e superiores para as áreas.

O problema corresponde a uma análise feita considerando-se as cargas concen-

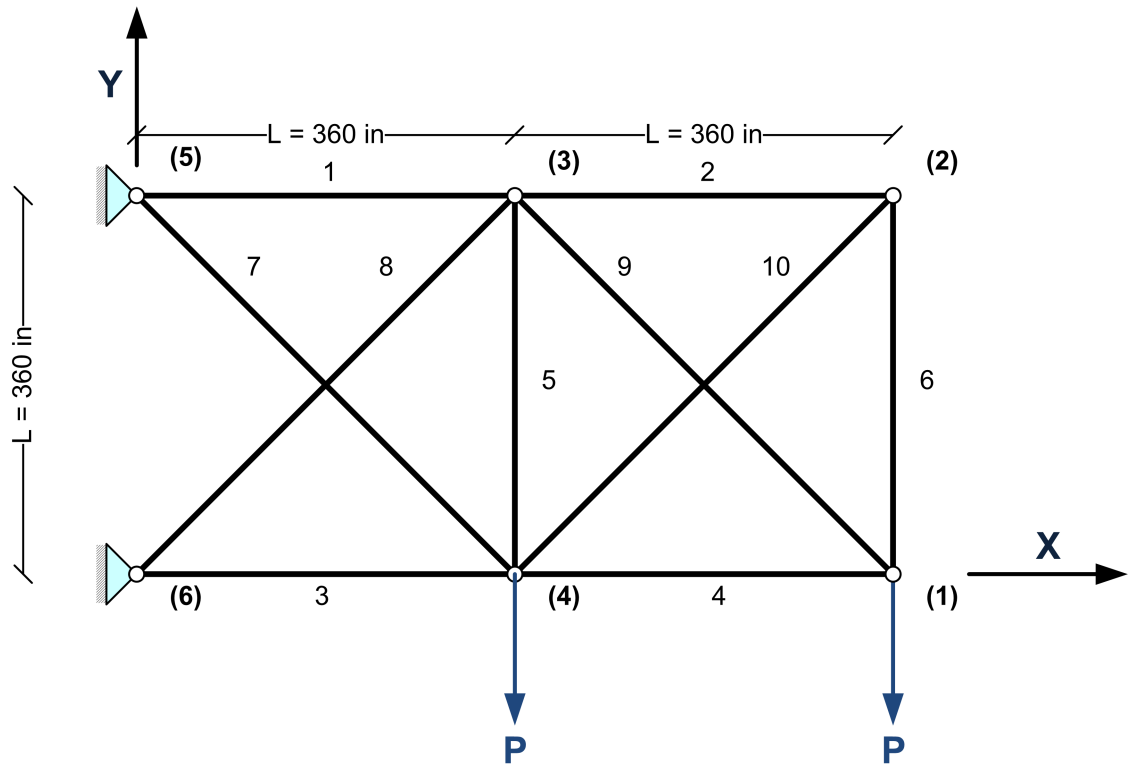


Figura 4.1: Treliça de 10 Barras.

Propriedade	Valor
Módulo de Elasticidade (E)	10.0^7 psi
Massa Específica (ρ)	0.1 lb/in^3
Deslocamentos Admissíveis (u_{max})	2.0 in
Tensões Admissíveis (σ_{max})	$\pm$25000.0 psi
Área: Limite Inferior (A_{min})	0.1 in^2
Área: Limite Superior (A_{max})	40.0 in^2

Tabela 4.1: Treliça de 10 barras - Propriedades do Material e Restrições.

tradas definidos com o valor de P = 100 Kips. Foi utilizado um AG geracional com codificação real, com os seguintes operadores de recombinação: *crossover* SBX (Deb and Agrawal, 1995) e mutação polinomial (Deb and Goyal, 1996).

O *crossover* SBX (Simulated Binary Crossover) foi desenvolvido para tratar variáveis de projetos representadas em codificação real. A ideia desse operador é criar um procedimento bastante similar ao cruzamento com um ponto de corte por variável, utilizado na codificação binária. A característica deste operador é gerar uma distribuição centrada nos pais, ao invés de uma distribuição centrada na média dos mesmos. Além disso, o SBX fornece um controle sobre a busca local através de um parâmetro ($\eta > 0$). Este parâmetro controla a distribuição das novas soluções, sendo que, quanto maior esse

valor, mais reunido em torno dos pais esta distribuição se torna.

Da mesma forma, a mutação polinomial foi desenvolvida para tratar variáveis de projetos representadas em codificação real. A ideia desse operador é criar uma pertubação do seu valor atual com o auxílio de uma distribuição polinomial, com a média no valor da variável e a variância em função de um parâmetro n. Quanto maior o valor do parâmetro n, menor será a variância da distribuição, portanto, menor será a mutação do valor original.

Dentro do operador de *crossover* SBX e o de mutação polinomial é necessário definir uma taxa responsável por selecionar aleatoriamente quais variáveis do indivíduo devem sofrer as devidas alterações. Os valores adotados para a probabilidade de recombinação e mutação são, respectivamente, 80% e 5%, e as taxas são ambas iguais a 50%. Além disso, foi adotado o elitismo na população, selecionando 5% dos melhores indivíduos para a geração seguinte e uma seleção de 80% dos melhores indivíduos para *crossover*.

Para completar a população da geração seguinte é realizado uma inserção de novos indivíduos gerados aleatoriamente dentro do espaço de busca. A população foi formada por 100 indivíduos e o processo evolutivo iterou por 200 gerações. O tempo médio de simulação para cada indivíduo da população foi de 4.14 segundos, demandando assim um tempo médio total de 23 horas.

A Figura 4.2 apresenta um gráfico da convergência da melhor solução em cada geração. A melhor solução encontrada possui peso final igual a 5159.9009 lbs. Suas variáveis de projeto são apresentadas na Tabela 4.2 onde "naval" representa o número de avaliações da função objetivo feitas pelo AG e esta solução é indicada por TCC.

Independentemente do orçamento computacional reduzido, a integração computacional obteve resultado satisfatório, convergindo para uma solução próxima da melhor solução conhecida, de peso igual à 5060.875 lb (Lemonge et al., 2015).

—	naval	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	Peso (lb)
TCC	20000	31.7064	0.1946	27.0424	16.0071	0.1060	0.3731	7.9972	18.6270	21.0473	0.3417	5159.9009
Ref.	80000	30.4629	0.1000	23.2828	15.2022	0.1000	0.5451	7.4467	21.0232	21.5545	0.1000	5060.8750

Tabela 4.2: Treliça de 10 barras - Comparação de Resultado: Variáveis de Projeto e Pesos Finais.

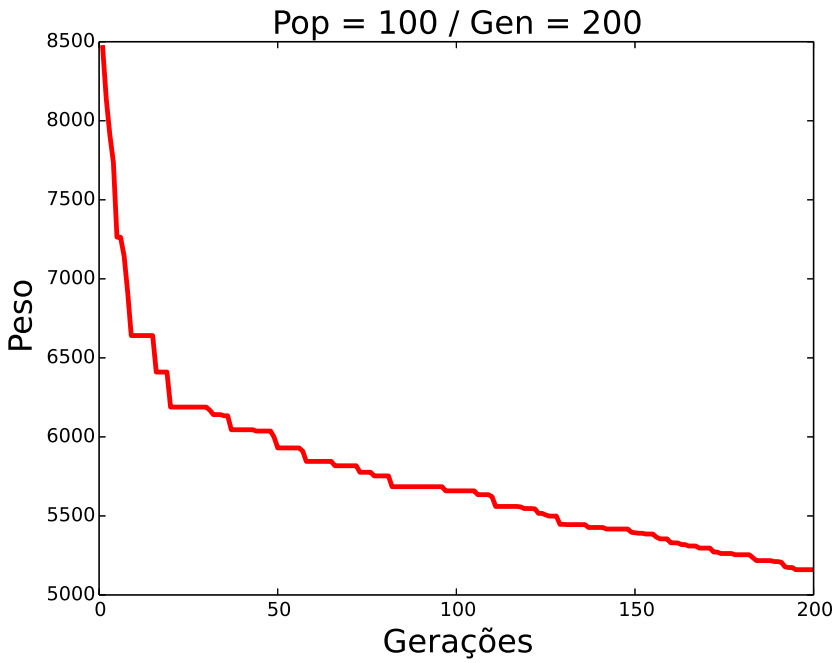


Figura 4.2: Treliça de 10 barras - População com 100 Indivíduos em 200 Gerações.

4.2 Viga engastada e livre

O problema de otimização bidimensional da viga engastada e livre (Figura 4.3) consiste em encontrar uma posição na mesma para realização de um furo cilíndrico de raio r, com o objetivo de minimizar o deslocamento vertical causado por uma carga concentrada P.

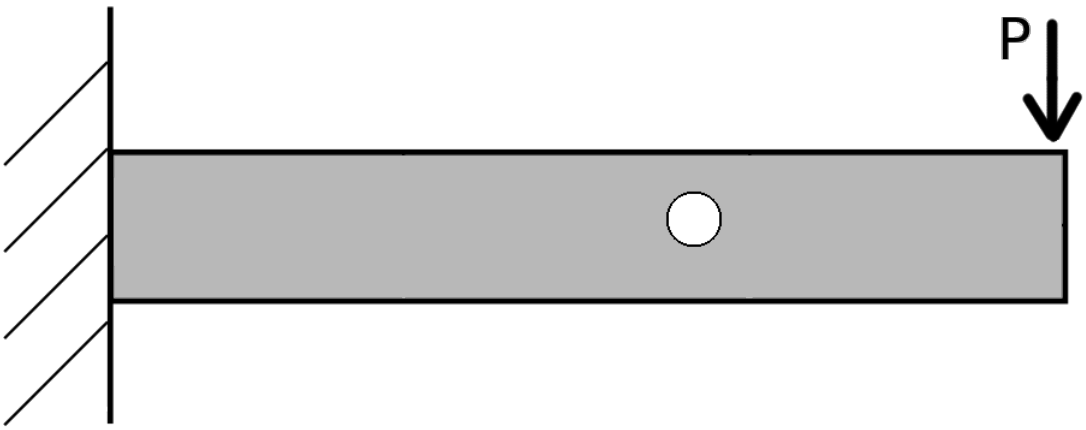


Figura 4.3: Viga Engastada e Livre.

As dimensões da viga referentes ao comprimento, largura e altura são de 150cm, 10cm e 20cm, respectivamente. Além disso, o valor atribuído para o raio do furo e para

a carga concentrada P são respectivamente 2.5 cm e 20 kN. O módulo de elasticidade longitudinal é igual a 80 GPa.

A malha utilizada neste problema possui entre 3000 e 4000 elementos quadráticos, construídas automaticamente pelo *software* Abaqus®. A variação da quantidade de elementos está diretamente relacionada com a posição do furo cilíndrico.

4.2.1 Minimização do deslocamento vertical

Aqui será tratado a busca pela posição que o furo cilíndrico deve ficar, com o objetivo de minimizar o deslocamento vertical δ_y.

O problema de otimização pode ser modelado da seguinte forma:

$$\begin{aligned} &\min \delta_y \\ &S.a: \\ &\quad 3.5 \leq x \leq 146.5 \text{ cm} \\ &\quad 3.5 \leq y \leq 16.5 \text{ cm} \end{aligned}$$

Foi utilizado um AG geracional com codificação real, com os seguintes operadores de recombinação: *crossover* SBX e mutação polinomial. Os valores adotados para a probabilidade de recombinação e mutação são, respectivamente, 80% e 100%, e as taxas são 100% e 5%. Além disso, foi realizado um processo de elitismo na população, selecionando 5% dos melhores indivíduos para a geração seguinte. A população foi formada por 40 indivíduos, o processo evolutivo iterou por 50 gerações. O tempo médio de simulação para cada indivíduo da população foi de 7.2 segundos, demandando assim um tempo médio total de 4 horas.

A Figura 4.4 apresenta um gráfico da convergência da melhor solução em cada geração. A melhor solução encontrada possui deslocamento vertical igual à -0.426939. Suas variáveis de projeto são apresentadas na Tabela 4.3.

naval	x	y	δ_y
2000	145.65807	16.48420	-0.426939

Tabela 4.3: Viga engastada e livre: Minimização do deslocamento vertical - Variáveis de Projeto.

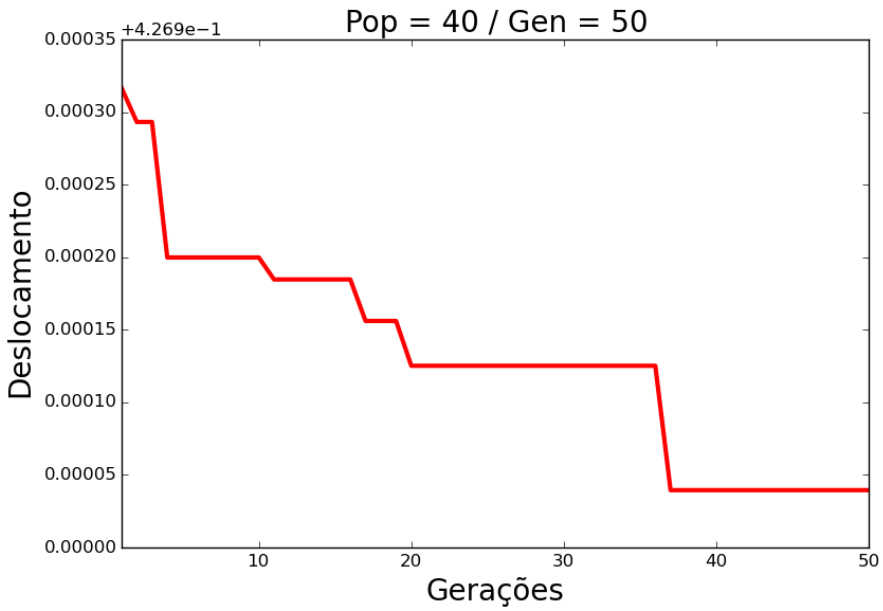


Figura 4.4: Viga engastada e livre: Minimização do deslocamento vertical - População com 40 Indivíduos em 50 Gerações.

Pode-se observar nas Figuras 4.5 e 4.6 a configuração encontrada pelo processo de otimização e o deslocamento vertical respectivamente.

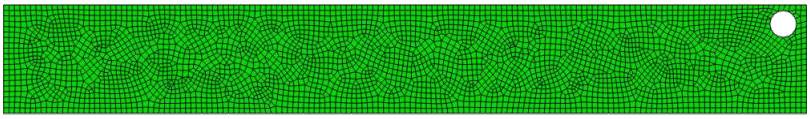


Figura 4.5: Viga engastada e livre: Minimização do deslocamento vertical - Configuração da Melhor Solução Encontrada.

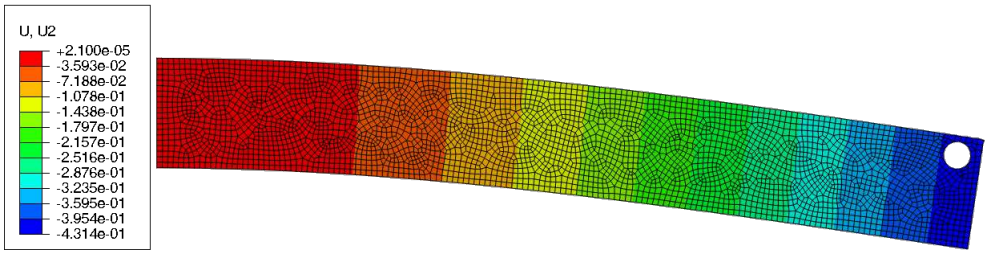


Figura 4.6: Viga engastada e livre: Minimização do deslocamento vertical - Deslocamento Vertical da Melhor Solução Encontrada.

4.2.2 Aproximação do deslocamento vertical

Aqui será tratado a busca pela posição que o furo cilíndrico deve ficar, com o

objetivo de aproximar o deslocamento vertical do problema entre um já conhecido, ou seja, trata-se de um problema inverso do proposto anteriormente.

O problema de otimização pode ser modelado da seguinte forma:

$$\min |\delta_y - \delta_{conhecido}|$$

$$S.a:$$

$$3.5 \leq x \leq 146.5 \text{ cm}$$

$$3.5 \leq y \leq 16.5 \text{ cm}$$

Para este problema, foi utilizado $\delta_{conhecido} = -0.426939$, referente a solução encontrada no problema anterior. Foi utilizado um AG geracional com codificação real, com os seguintes operadores de recombinação: *crossover* SBX e mutação polinomial. Os valores adotados para a probabilidade de recombinação e mutação são, respectivamente, 80% e 100%, e as taxas são 100% e 5%. Além disso, foi realizado um processo de elitismo na população, selecionando 5% dos melhores indivíduos para a geração seguinte. A população foi formada por 40 indivíduos, o processo evolutivo iterou por 50 gerações. O tempo médio de simulação para cada indivíduo da população foi de 7.2 segundos, demandando assim um tempo médio total de 4 horas.

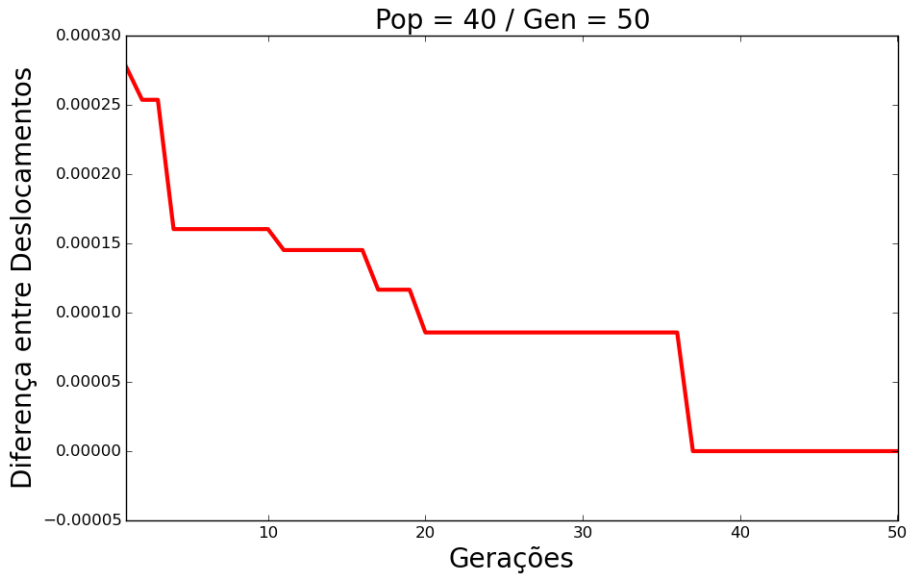


Figura 4.7: Viga engastada e livre: Aproximação do deslocamento vertical - População com 40 Indivíduos em 50 Gerações.

A Figura 4.7 apresenta um gráfico da convergência da melhor solução em cada geração. A melhor solução encontrada possui deslocamento vertical igual à -0.426939,

igual ao valor atribuído para $\delta_{conhecido}$. Suas variáveis de projeto são apresentadas na Tabela 4.4.

naval	x	y	δ_y
2000	145.65807	16.48420	-0.426939

Tabela 4.4: Viga engastada e livre: Aproximação do deslocamento vertical - Variáveis de Projeto.

Pode-se observar nas Figuras 4.8 e 4.9 a configuração encontrada pelo processo de otimização e o deslocamento vertical respectivamente. Percebe-se que a solução encontrada é igual a solução do problema proposto anteriormente, visto que foi atribuído o resultado do mesmo para o $\delta_{conhecido}$.

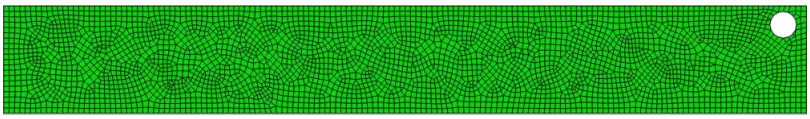


Figura 4.8: Viga engastada e livre: Aproximação do deslocamento vertical - Configuração da Melhor Solução Encontrada.

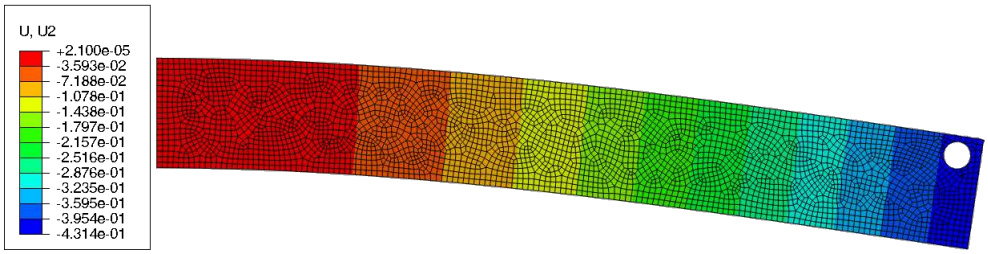


Figura 4.9: Viga engastada e livre: Aproximação do deslocamento vertical - Deslocamento Vertical da Melhor Solução Encontrada.

4.3 Trocador de calor

O problema de otimização de um modelo bidimensional de um Trocador de Calor (Figura 4.10) consiste em encontrar as posições mais favoráveis dos tubos de modo a maximizar a troca de calor e, ao mesmo tempo, minimizar a perda de pressão (Thévenin and Janiga, 2008). Uma configuração do Trocador de Calor é considerada utilizando a solução acoplada dos processos de transferência de fluxo e calor.

Um problema de CFD que foi corretamente formulado deve incluir um conjunto de condições de contorno adequados para velocidade, temperatura e outras variáveis. Contudo, algumas vezes é necessário utilizar condição de contorno artificial, visto que não é possível determinar o comportamento da região (Zikanov, 2010). Uma possibilidade para esta condição de contorno artificial é atribuir pressão igual a zero.

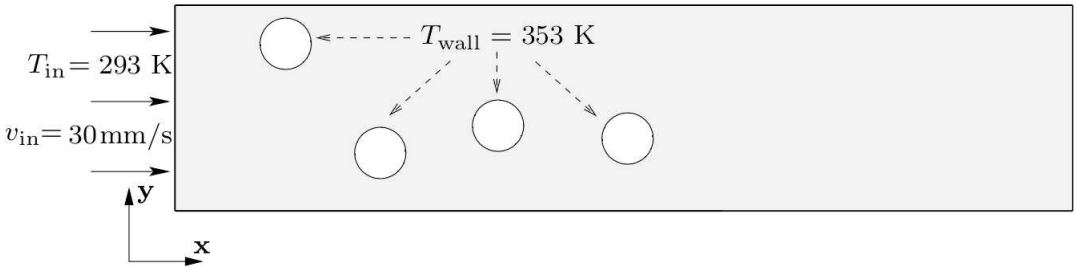


Figura 4.10: Modelo Bidimensional de um Trocador de Calor.

Na parte superior, inferior e em todos os tubos é aplicado uma condição de contorno de parede, onde a velocidade do fluido na superfície é igual a zero. Além disso, na saída, deve existir uma condição de contorno artificial, visto que ocorre um corte artificial no fluxo gerado dentro do domínio, impossibilitando determinar o comportamento real do mesmo. Com isso, na saída, é atribuir pressão igual a zero.

O domínio do trocador de calor é x = [0; 350] e y = [0; 80], em milímetros, sendo que, dentro dele, são posicionados os tubos de raio = 10 mm com temperatura constante T_{wall} = 353 K. O ar entra no domínio em x = 0 com temperatura T_{in} = 293 K a uma velocidade igual a 30 mm/s. O problema de otimização deve garantir algumas restrições do problema, onde os tubos não podem encostar na fronteira do domínio e nem entre si. A Tabela 4.5 apresenta as propriedades do fluido.

O comprimento do domínio foi escolhido para evitar qualquer influência do fluxo de entrada ou saída das condições de contorno. O número de Reynolds é igual a 41, definido utilizando o diâmetro do tubo ($D = 20$ mm) e a velocidade uniforme na entrada do domínio ($v_{in} = 30mm/s$). Desta forma, podemos afirmar que o escoamento dentro do domínio é laminar, visto que $Re = \frac{\rho v D}{\mu} < 2000$.

A malha utilizada neste problema possui entre 11000 e 12000 elementos triangulares, construídas automaticamente pelo *software* Abaqus®. A variação da quantidade de elementos está diretamente relacionada com as posições de todos os tubos cilíndricos.

Propriedade do Ar	Valor
Densidade (ρ)	$1.2047x10^{-9}$ kg/mm3
Viscosidade Dinâmica (μ)	$1.8205x10^{-8}$ kg/mm $\times$ s
Condutividade (k)	$0.025596x10^{-3}$ W/mm $\times$ k
Calor Específico à Pressão Constante (c_p)	1006.1 J/kg.K

Tabela 4.5: Trocador de calor - Propriedades do Ar.

4.3.1 Número de tubos fixo

Será tratado a otimização trocador de calor, onde o número de tubos é fixo e igual a 4. Como os tubos não podem encostar na fronteira do domínio e nem entrei si, foi adotado que eles devem estar no mínimo 5 mm distante da fronteira e 10 mm de qualquer outro tubo.

Sendo assim, o problema pode ser modelado da seguinte forma:

$$\max \Delta T$$

$$\max \Delta P$$

$$S.a:$$

$$15 \leq x_i \leq 35, \forall\ i = 1, 2, 3 \text{ e } 4$$

$$15 \leq y_i \leq 65, \forall\ i = 1, 2, 3 \text{ e } 4$$

e, as posições x_{ci} reais dos tubos é dada pela seguinte equação (Figura 4.11):

$$x_{c1} = x_1$$

$$x_{ci} = x_{c(i-1)} + x_i + 15, \forall i = 2, 3 \text{ e } 4.$$

Foi utilizado um AG geracional com codificação real, com os seguintes operadores de recombinação: *crossover* SBX e mutação polinomial. Os valores adotados para a probabilidade de recombinação e mutação são, respectivamente, 80% e 100%, e as taxas são 100% e 5%. Além disso, foi realizado um processo de elitismo na população, selecionando 5% dos melhores indivíduos para a geração seguinte.

O problema do trocador de calor com número de tubos fixos foi analisado e otimizado de 3 maneiras diferentes: Maximização da troca de calor, minimização da perda de pressão e, por fim, a otimização multiobjetivo. Os valores de ΔT e ΔP foram obtidos através da diferença entre a média dos valores da saída e de entrada do domínio.

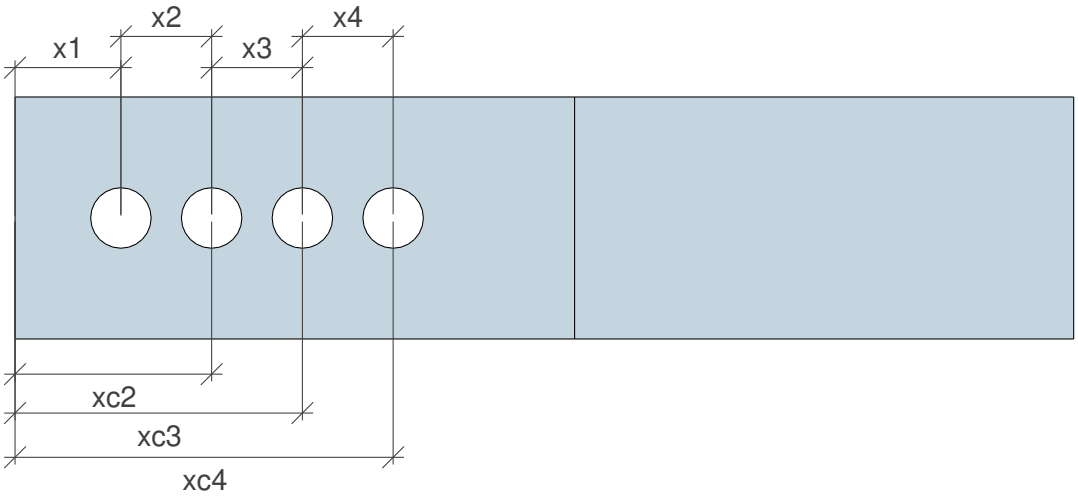


Figura 4.11: Número de tubos fixo - Imagem Representativa da Distribuição dos Tubos.

4.3.1.1 Maximização da troca de calor

Pretende-se aqui maximizar a troca de calor, sem levar em conta a perda de pressão. A população foi formada por 40 indivíduos, o processo evolutivo iterou por 20 gerações. O tempo médio de simulação para cada indivíduo da população foi de 36 segundos, demandando assim um tempo médio total de 8 horas.

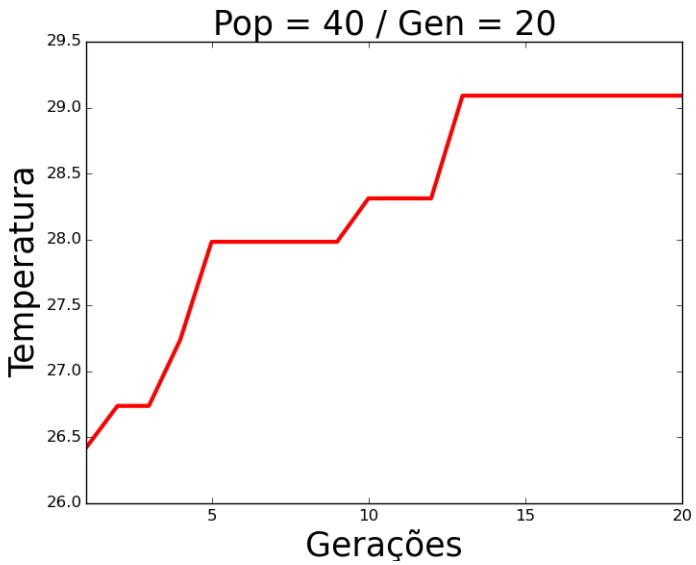


Figura 4.12: Número de tubos fixo: Maximização da troca de calor - População com 40 Indivíduos em 20 Gerações.

A Figura 4.12 apresenta um gráfico da convergência da melhor solução em cada geração. A melhor solução encontrada possui os seguintes valores: $\Delta T = 29.09083$ e $\Delta P = -0.00474$. Suas variáveis de projeto são apresentadas na Tabela 4.6.

naval	x_1	y_1	x_2	y_2	x_3	y_3	x_4	y_4
800	18.40394	14.59960	28.09929	36.75697	18.09952	63.04828	33.76259	42.93328

Tabela 4.6: Número de tubos fixo: Maximização da troca de calor - Variáveis de Projeto.

As Figuras 4.13 e 4.14 mostram respectivamente a temperatura e a pressão em todo o domínio do trocador de calor para a melhor solução.

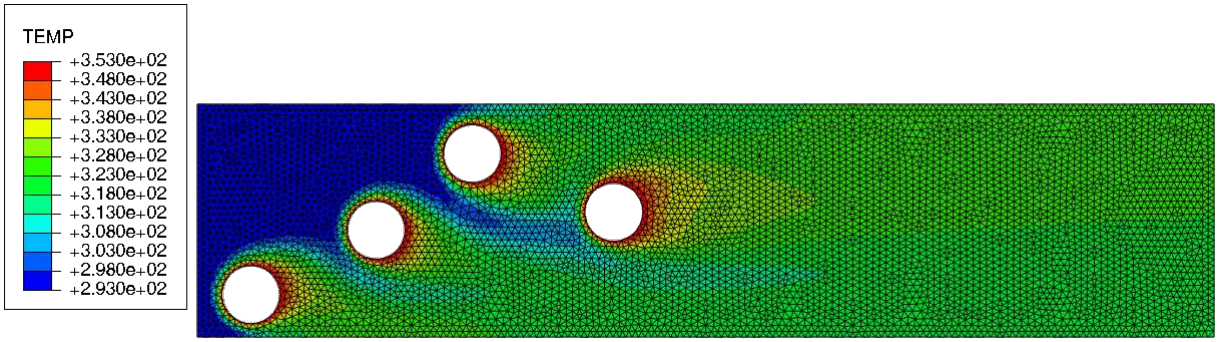


Figura 4.13: Número de tubos fixo: Maximização da troca de calor - Melhor Solução - Temperatura no Domínio.

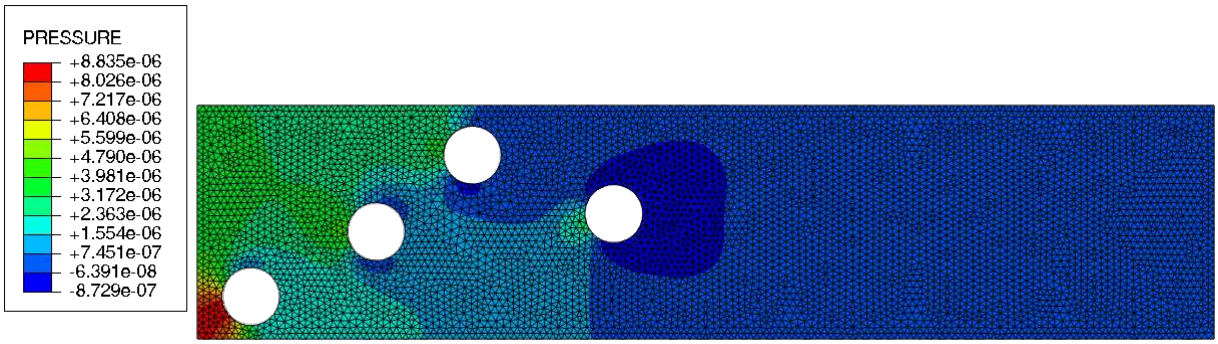


Figura 4.14: Número de tubos fixo: Maximização da troca de calor - Melhor Solução - Pressão no Domínio.

4.3.1.2 Minimização da perda de pressão

Pretende-se aqui minimizar a perda de pressão, sem levar em conta a troca de calor. A população foi formada por 40 indivíduos e o processo evolutivo iterou por 20

gerações. O tempo médio de simulação para cada indivíduo da população foi de 36 segundos, demandando assim um tempo médio total de 8 horas.

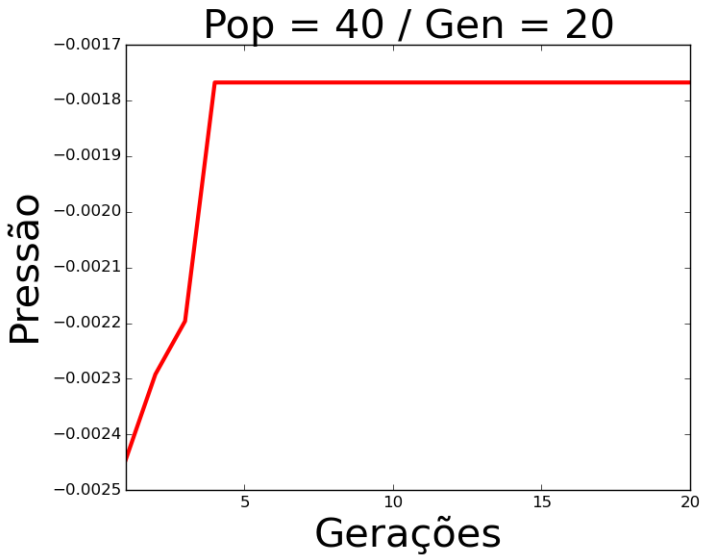


Figura 4.15: Número de tubos fixo: Minimização da perda de pressão - População com 40 Indivíduos em 20 Gerações.

A Figura 4.15 apresenta um gráfico da convergência da melhor solução em cada geração. A melhor solução encontrada possui os seguintes valores: $\Delta T = 9.56677$ e $\Delta P = -0.00176$. Suas variáveis de projeto são apresentadas na Tabela 4.7.

naval	x_1	y_1	x_2	y_2	x_3	y_3	x_4	y_4
800	28.04298	14.15481	28.07695	12.51660	22.98925	17.13080	32.26974	13.46213

Tabela 4.7: Número de tubos fixo: Minimização da perda de pressão - Variáveis de Projeto.

As Figuras 4.16 e 4.17 mostram respectivamente a temperatura e a pressão em todo o domínio do trocador de calor para a melhor solução.

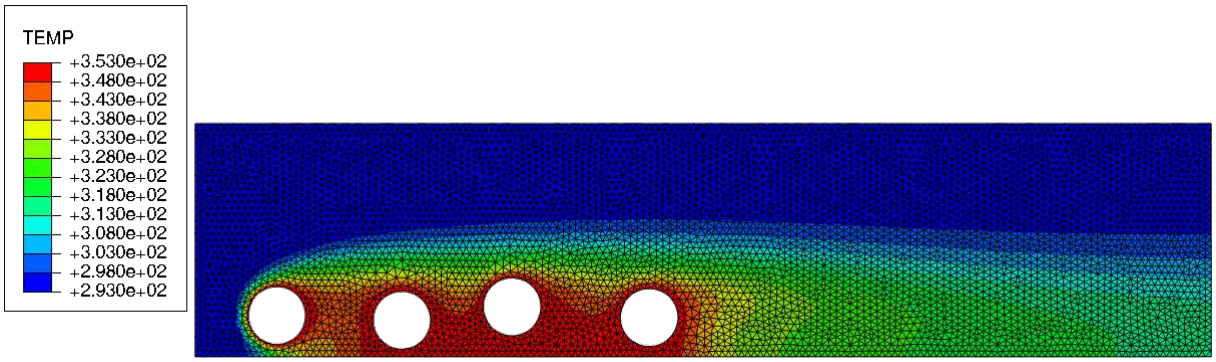


Figura 4.16: Número de tubos fixo: Minimização da perda de pressão - Melhor Solução - Temperatura no Domínio.

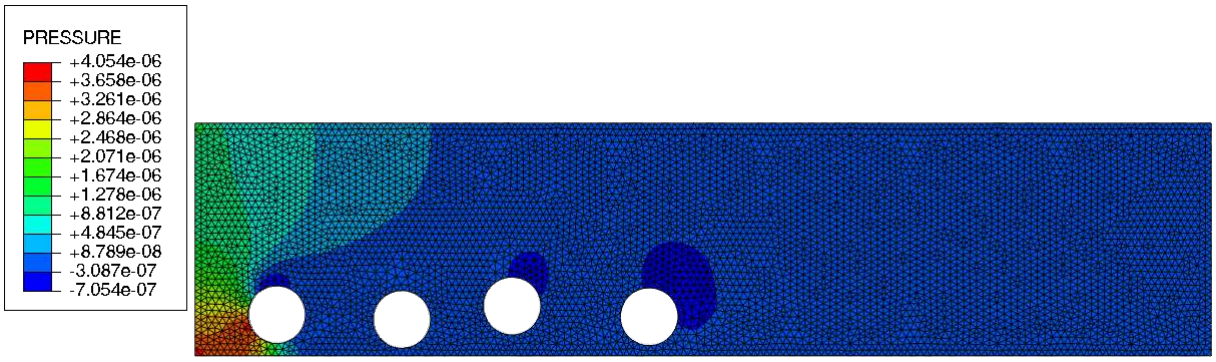


Figura 4.17: Número de tubos fixo: Minimização da perda de pressão - Melhor Solução - Pressão no Domínio.

4.3.1.3 Otimização multiobjetivo

Pretende-se aqui maximizar a troca de calor e minimizar a perda de pressão. Um dos métodos mais comuns para resolver problemas de otimização multiobjetivo é o método dos pesos (Zadeh, 1963). O método consiste em criar uma única função objetivo composta pela combinação de todas as funções objetivo utilizando um vetor de pesos.

A escolha dos pesos deve ser feita de forma a não possuir nenhuma preferência entre as funções objetivo, isso pode ser feito utilizando os valores obtidos através das otimizações mono-objetivo das mesmas. Os pesos devem ser iguais ao inverso da melhor solução encontrada na otimização mono-objetivo. Este critério, entre vários (Deb, 2001), foi o adotado aqui.

Pretende-se chegar a um conjunto de soluções não-dominadas em ambos os objetivos que formarão o que se denomina frente de Pareto (Pareto, 1896). A população foi formada por 40 indivíduos e o processo evolutivo iterou por 20 gerações. O tempo médio de simulação para cada indivíduo da população foi de 36 segundos, demandando assim um tempo médio total de 8 horas.

A Figura 4.18 apresenta o conjunto ótimo de Pareto contendo 36 indivíduos não dominados, obtidos através de um pós-processamento dos indivíduos gerados durante todo o processo de otimização. A Tabela 4.8 e 4.9 apresentam a solução de 6 indivíduos deste conjunto.

É possível observar que as soluções com menor perda pressão tendem a ficar enfileiradas, enquanto as soluções com maior troca de calor tendem a ficar mais distribuídas

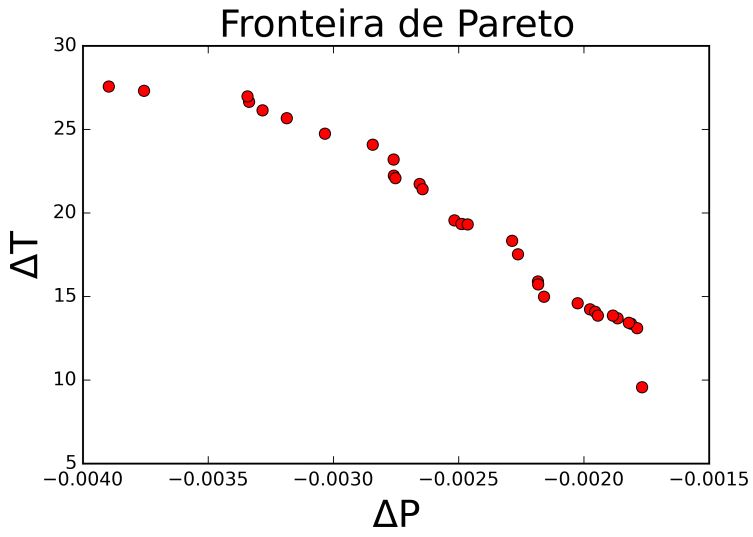


Figura 4.18: Número de tubos fixo: Otimização multiobjetivo - Conjunto Ótimo de Pareto.

ΔT	ΔP	x_1	y_1	x_2	y_2	x_3	y_3	x_4	y_4
28.77364	-0.00433	19.85937	62.67720	31.87051	43.12587	18.92783	14.43835	17.68338	36.82027
26.97394	-0.00334	23.31881	30.65541	23.53917	52.40709	15.81255	63.76963	28.22381	44.06357
22.22486	-0.00275	34.63457	31.26658	24.37283	36.55402	30.75956	25.70545	16.37889	16.76922
19.31215	-0.00246	29.71192	30.20792	24.49357	26.63028	20.19117	16.84378	24.25903	14.95929
14.23340	-0.00197	26.92801	43.30663	28.60848	14.81475	29.64305	14.96012	19.58239	12.11554
9.56677	-0.00176	28.04298	14.15481	28.07695	12.51660	22.98925	17.13080	32.26974	13.46213

Tabela 4.8: Número de tubos fixo: Otimização multiobjetivo - ΔT, ΔP e as Variáveis de Projeto.

no domínio. Não se pode afirmar qual é a melhor solução entre as demais, visto que elas são soluções não-dominadas. A escolha de uma configuração deve ser feita por um especialista, onde ele irá escolher a que satisfaça melhor o seu projeto.

4.3.2 Número de tubos variável

Será tratada a otimização no trocador de calor onde o número de tubos varia entre 1 e 4. Como os tubos não podem encostar na fronteira do domínio e nem entre si, foi adotado que eles devem estar no mínimo 5 mm distante da fronteira e 10 mm de qualquer outro tubo.

Sendo assim, o problema pode ser modelado da seguinte forma:

$$\max \Delta T$$

$$\max \Delta P$$

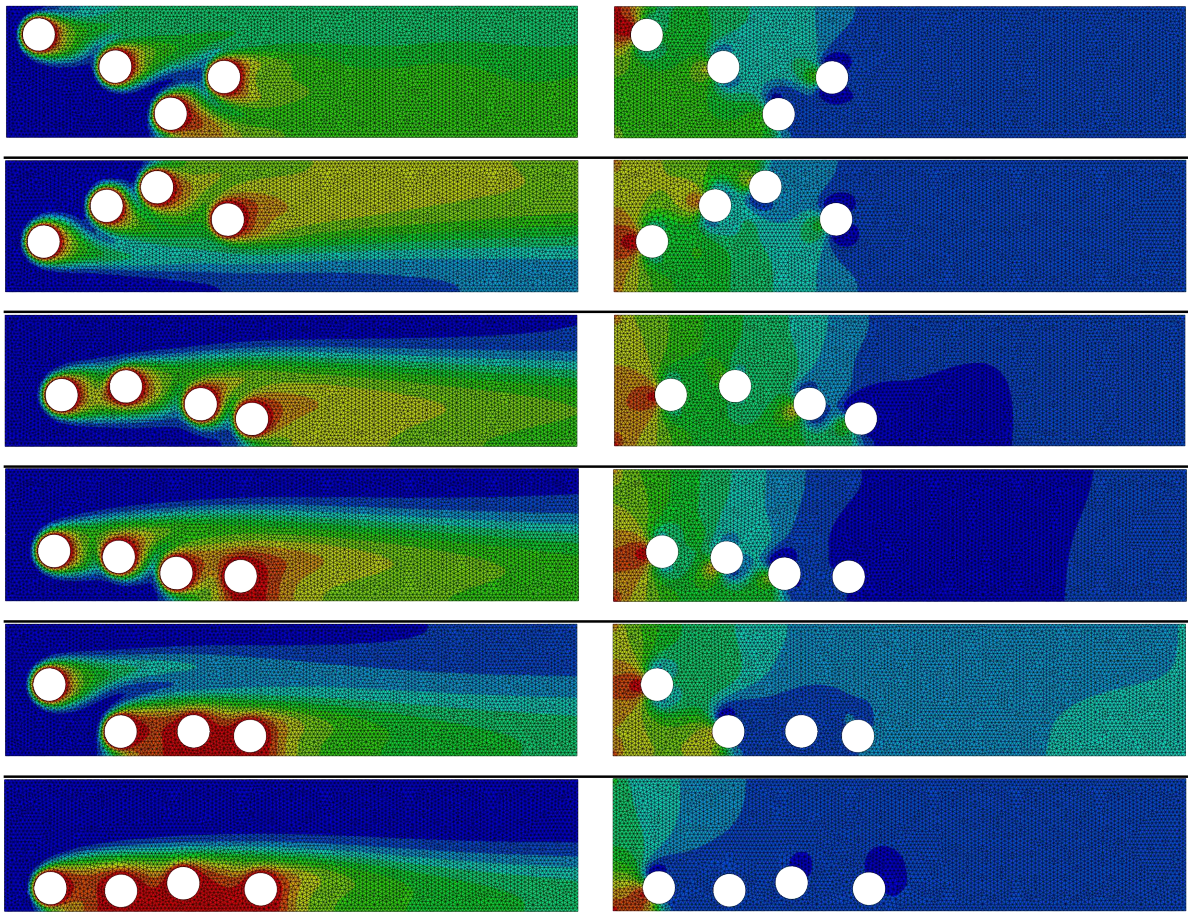
ΔT	ΔP
	

Tabela 4.9: Número de tubos fixo: Otimização multiobjetivo - Conjunto de Soluções.

$$
\begin{aligned}
&S.a: \\
&\quad 1 \leq n_{tubos} \leq 4 \\
&\quad 15 \leq x_i \leq 35, \forall\ i = 1, 2, 3 \text{ e } 4 \\
&\quad 15 \leq y_i \leq 65, \forall\ i = 1, 2, 3 \text{ e } 4
\end{aligned}
$$

e, as posições x_{ci} reais dos tubos é dada pela seguinte equação:

$$
\begin{aligned}
x_{c1} &= \tfrac{4}{n_{tubos}} x_1 + 15(4 - n_{tubos}) \\
x_{ci} &= x_{c(i-1)} + \tfrac{4}{n_{tubos}} x_i + 15, \forall i = 2, 3 \text{ e } 4.
\end{aligned}
$$

É importante notar que as posições reais dependem diretamente da quantidade de tubos presentes no sistema, o que possibilita manter o espaço de busca do problema independente do indivíduo analisado. Além disso, devemos considerar o número de tubos como uma variável de projeto, sendo necessário introduzi-lo no processo de busca.

Foi utilizado um AG geracional com codificação real, com os seguintes operadores de recombinação: *crossover* SBX e mutação polinomial. Os valores adotados para a pro-

babilidade de recombinação e mutação são, respectivamente, 80% e 100%, e as taxas são 100% e 5%. Além disso, foi realizado um processo de elitismo na população, selecionando 5% dos melhores indivíduos para a geração seguinte.

O problema do trocador de calor com número de tubos variável foi analisado e otimizado de 3 maneiras diferentes: Maximização da troca de calor, minimização da perda de pressão e, por fim, a otimização multiobjetivo. Os valores de ΔT e ΔP foram obtidos através da diferença entre a média dos valores da saída e de entrada do domínio.

4.3.2.1 Maximização da troca de calor

Pretende-se aqui maximizar a troca de calor, sem levar em conta a perda de pressão. A população foi formada por 40 indivíduos, o processo evolutivo iterou por 20 gerações. O tempo médio de simulação para cada indivíduo da população foi de 36 segundos, demandando assim um tempo médio total de 8 horas.

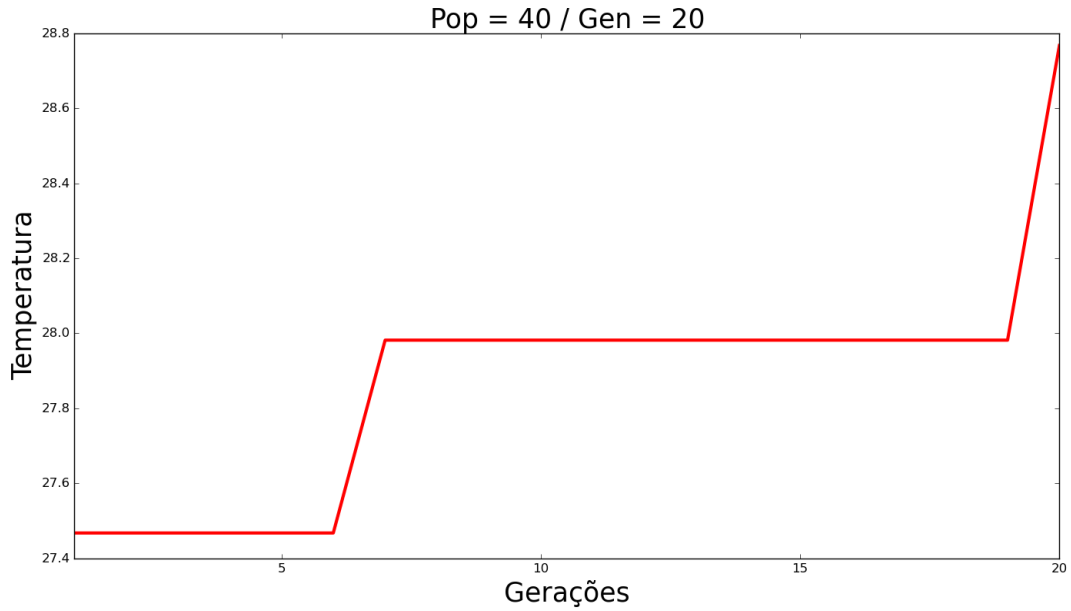


Figura 4.19: Número de tubos variável: Maximização da troca de calor - População com 40 Indivíduos em 20 Gerações.

A Figura 4.19 apresenta um gráfico da convergência da melhor solução em cada geração. A melhor solução encontrada possui os seguintes valores: $\Delta T = 28.76740$ e $\Delta P = -0.00433$. Suas variáveis de projeto são apresentadas na Tabela 4.10.

naval	x_1	y_1	x_2	y_2	x_3	y_3	x_4	y_4	n_{tubos}
800	31.20788	63.68768	19.29753	33.43302	25.87855	16.42637	21.80451	57.90464	4

Tabela 4.10: Número de tubos variável: Maximização da troca de calor - Variáveis de Projeto.

As Figuras 4.20 e 4.21 mostram respectivamente a temperatura e a pressão em todo o domínio do trocador de calor.

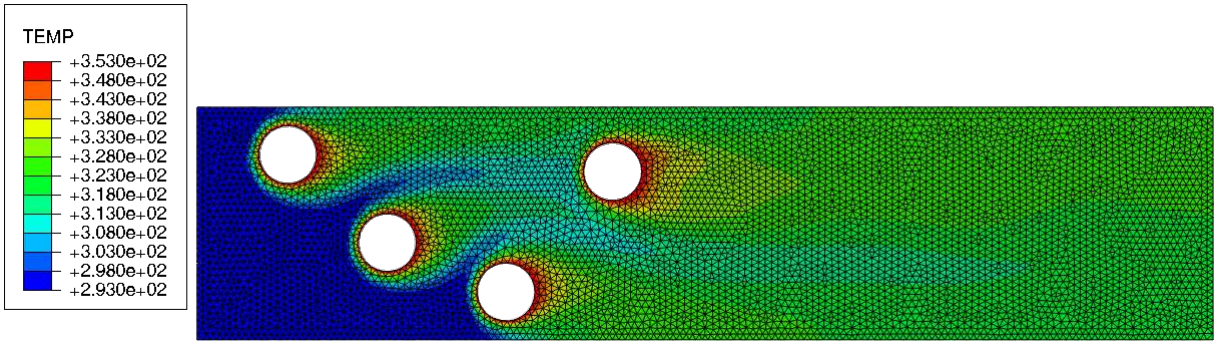


Figura 4.20: Número de tubos variável: Maximização da troca de calor - Melhor Solução - Temperatura no Domínio.

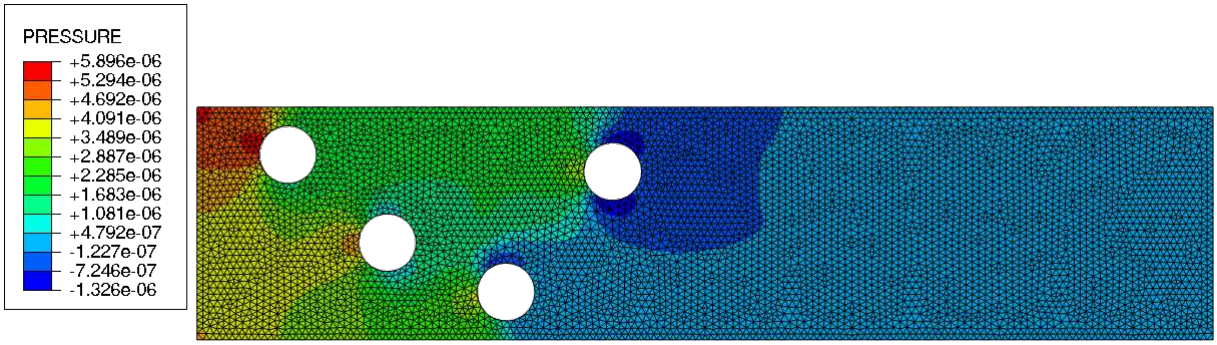


Figura 4.21: Número de tubos variável: Maximização da troca de calor - Melhor Solução - Pressão no Domínio.

4.3.2.2 Minimização da perda de pressão

Pretende-se aqui minimizar a perda de pressão, sem levar em conta a troca de calor. A população foi formada por 40 indivíduos e o processo evolutivo iterou por 20 gerações. O tempo médio de simulação para cada indivíduo da população foi de 36 segundos, demandando assim um tempo médio total de 8 horas.

A Figura 4.22 apresenta um gráfico da convergência da melhor solução em cada

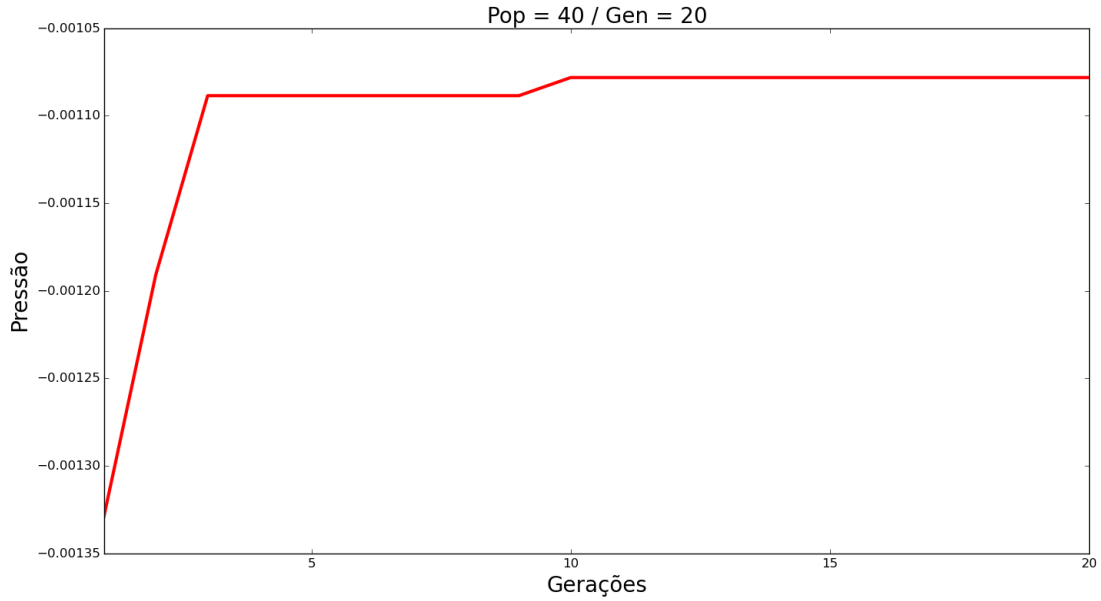


Figura 4.22: Número de tubos variável: Minimização da perda de pressão - População com 40 Indivíduos em 20 Gerações.

geração e a melhor solução encontrada possui os seguintes valores: $\Delta T = 3.98011$ e $\Delta P = -0.00107$. Suas variáveis de projeto são apresentadas na Tabela 4.11.

naval	x_1	y_1	x_2	y_2	x_3	y_3	x_4	y_4	n_{tubos}
800	18.88588	12.21036	32.80798	49.13921	24.54177	23.90547	27.19827	50.75584	1

Tabela 4.11: Número de tubos variável: Minimização da perda de pressão - Variáveis de Projeto.

As Figuras 4.23 e 4.24 mostram respectivamente a temperatura e a pressão em todo o domínio do trocador de calor.

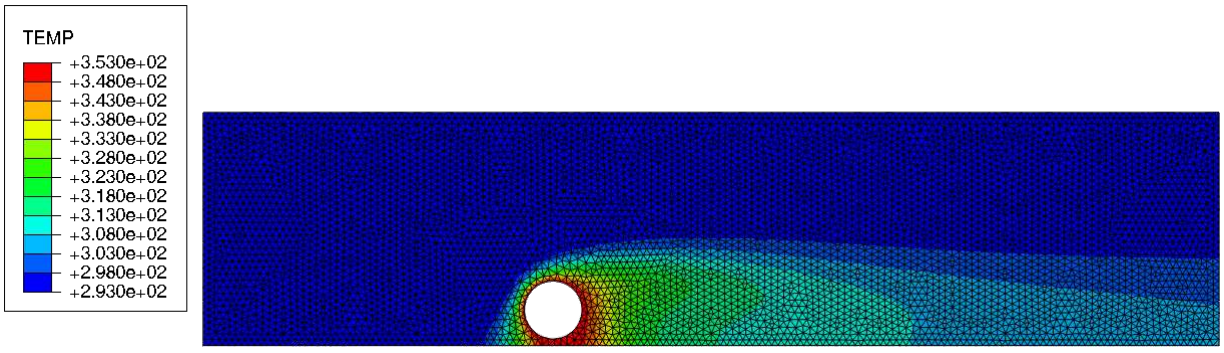


Figura 4.23: Número de tubos variável: Minimização da perda de pressão - Melhor Solução - Temperatura no Domínio.

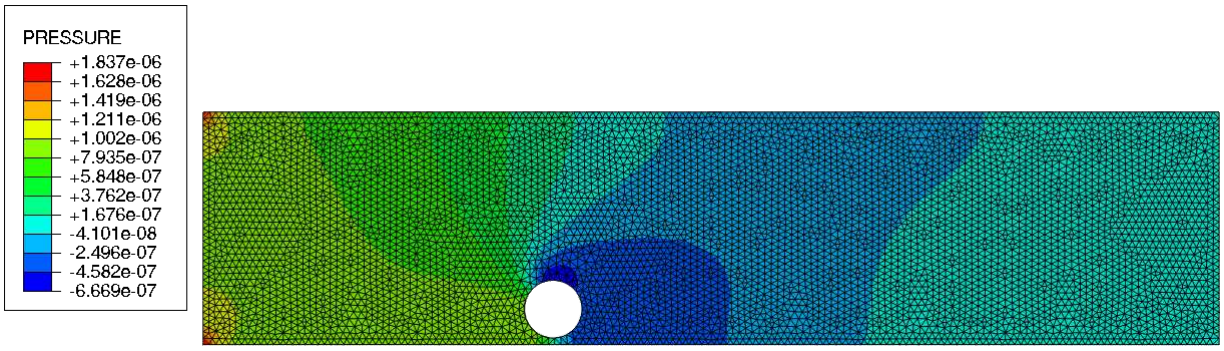


Figura 4.24: Número de tubos variável: Minimização da perda de pressão - Melhor Solução - Pressão no Domínio.

4.3.2.3 Otimização multiobjetivo

Pretende-se aqui maximizar a troca de calor e minimizar a perda de pressão e alcançar um conjunto de soluções não-dominadas em ambos os objetivos que formarão o que se denomina frente de Pareto. A população foi formada por 40 indivíduos e o processo evolutivo iterou por 20 gerações. O tempo médio de simulação para cada indivíduo da população foi de 36 segundos, demandando assim um tempo médio total de 8 horas.

A Figura 4.25 apresenta o conjunto ótimo de Pareto contendo 120 indivíduos não dominados, obtidos através de um pós-processamento dos indivíduos gerados durante todo o processo de otimização. A Tabela 4.12 e 4.13 apresentam a solução de 6 indivíduos deste conjunto.

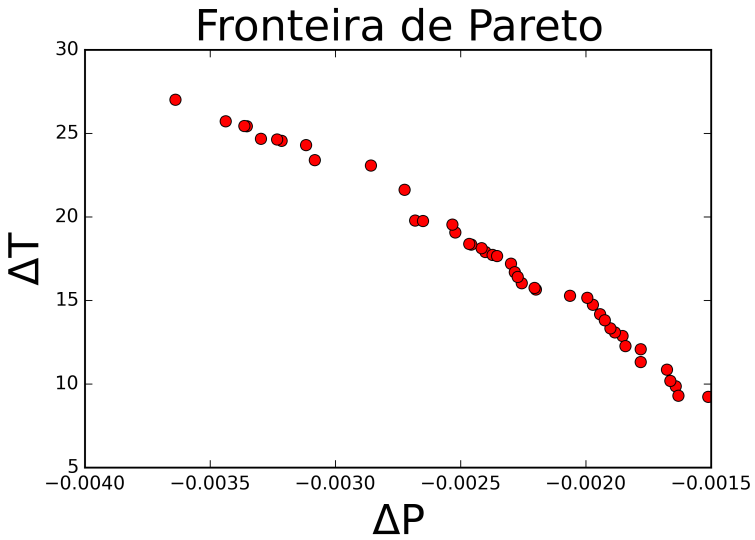


Figura 4.25: Número de tubos variável: Otimização multiobjetivo - Conjunto Ótimo de Pareto.

ΔT	ΔP	x_1	y_1	x_2	y_2	x_3	y_3	x_4	y_4	n_{tubos}
28.67225	-0.00560	15.08281	67.77885	17.72763	48.00625	24.79417	33.28836	25.74404	16.38387	4.0
27.89976	-0.00451	18.96238	16.84799	29.65144	64.27290	33.93466	27.31056	28.34550	53.28785	4.0
17.73156	-0.00237	32.70823	30.93637	28.70894	21.13983	26.14736	31.96590	23.76292	36.57833	3.0
12.08049	-0.00178	17.17528	67.70881	21.93457	51.40604	33.55444	24.35081	29.81056	33.70096	2.0
4.72579	-0.00117	34.61225	13.35669	16.91800	32.58906	19.52490	63.84345	26.17312	23.25886	1.0
3.93388	-0.00106	15.70226	67.91235	33.08907	28.87695	19.88467	16.85112	20.09971	38.89871	1.0

Tabela 4.12: Número de tubos variável: Otimização multiobjetivo - ΔT, ΔP e as Variáveis de Projeto.

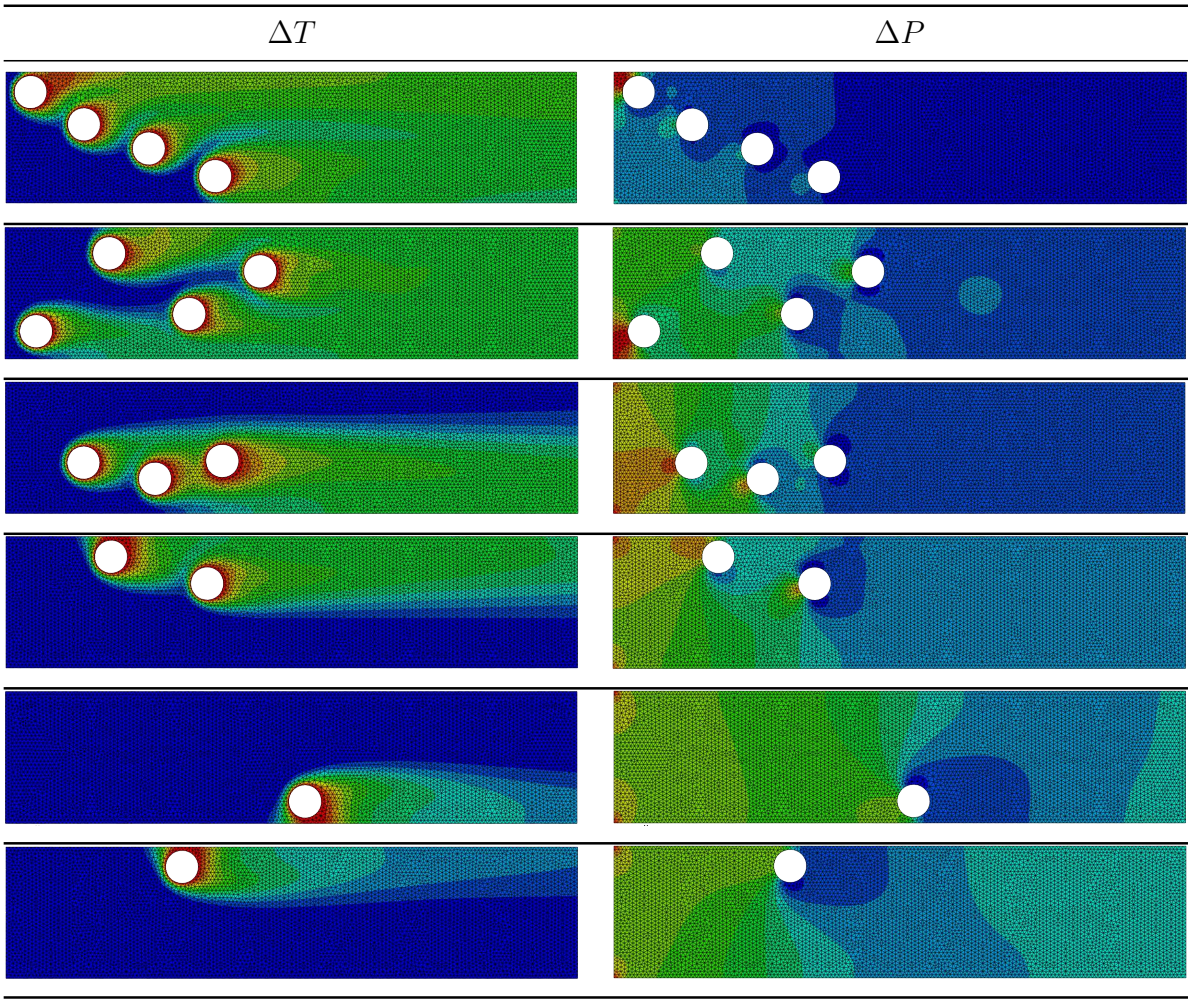


Tabela 4.13: Número de tubos variável: Otimização multiobjetivo - Conjunto de Soluções.

É possível observar que as soluções com menor perda de pressão tendem a ficar com o número minimo de tubos no domínio, enquanto as soluções com maior troca de calor tendem a ficar com número máximo. Quanto menor o número de tubos, mais fácil o fluido passará por todo domínio, encontrando poucas resistência, o que torna a perda de pressão baixa. Por outro lado, quanto maior o número de tubos, maior é a transferência de calor, visto que maior seria o contado do fluido frio com os tubos quentes.

Não se pode afirmar qual é a melhor solução entre as demais, visto que elas são

soluções não-dominadas. A escolha de uma configuração deve ser feita por um especialista, onde ele irá escolher a que satisfaça melhor o projeto.

5 CONCLUSÕES

Neste trabalho foi proposto a integração de ferramentas computacionais para a solução de problemas de otimização em engenharia. Para este fim, foi utilizado o *software* Abaqus® para realizar simulações enquanto um Algoritmo Genético implementado em Python ficou responsável pelo processo de busca da solução. Foi acoplado um método de penalização adaptativa (APM) ao Algoritmo Genético para o tratamento das restrições comumente encontradas em problemas de otimização.

Para avaliar a viabilidade da integração, foram feitos experimentos numéricos referente a problemas em engenharia. Estes experimentos são referentes a minimizar peso de uma treliça de 10 barras, minimizar o deslocamento vertical de uma viga engastada através de um furo cilindro na mesma e, por fim, maximizar a troca de calor e minimizar a perda de pressão em um trocador de calor posicionando tubos dentro de seu domínio.

Os resultados apresentados pelos experimentos numéricos validam a integração entre as ferramentas, mostrando o grande potencial do *software* Abaqus® para simulações de problemas de otimização em engenharia. É importante concluir que apesar de viabilizar a otimização em projetos complexos, a utilização do Abaqus® requer mais tempo de processamento, pois as trocas de dados entre processo de busca e simulador são realizados via escrita/leitura de arquivos em disco.

Como trabalhos futuros, sugere-se a implementação e a validação da integração proposta aqui na resolução de outros problemas da engenharia, bem como a obtenção de dados estatísticos referentes às otimizações presentes neste trabalho. Dentre os problemas futuros, destacam-se prioritariamente os problemas de otimização referentes a dinâmica dos fluidos, assim como: à redução da força de arrato em veículos automotivos (Figura 5.1) e a otimização aerodinâmica de aerofólios (Figura 5.2).

Os problemas relacionados à redução do arrasto nos veículos automotivos podem ser considerados um dos tópicos de maior interesse nesta área o que, evidentemente, impacta diretamente na redução do consumo de combustível e na consequente redução

das emissões de gás carbônico na atmosfera. Pretende-se investigar modelos simples de formas automotivas, ou parte delas, em duas dimensões por exemplo, buscando-se através dos problemas a serem estabelecidos achar formas otimizadas que minimizem o arrasto nas partes de interesse.

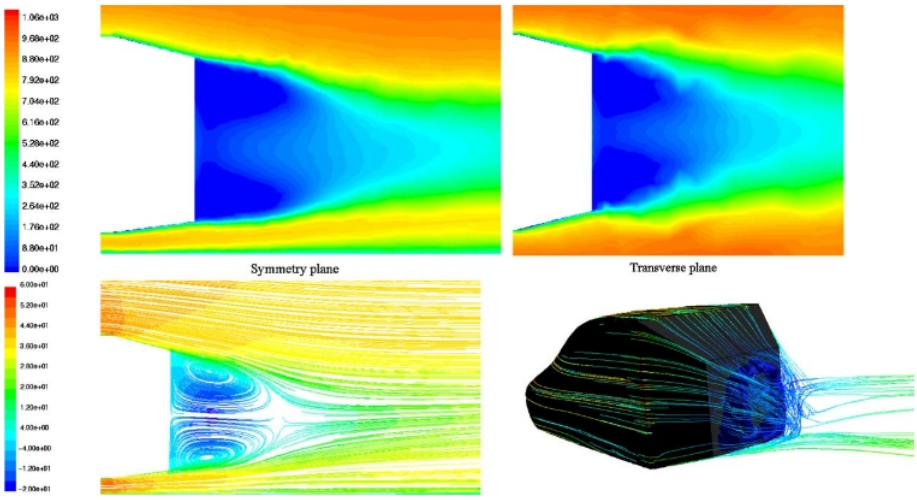


Figura 5.1: Ilustração da simulação as forças de arrasto em um veículo automotivo. Extraída de (Muyl et al., 2004).

Em razão das complexidades naturais referentes aos altos custos computacionais a otimização de forma de aerofólios recebe esforços contínuos em projetos contra intuitivos. Assim como o item anterior, este tema apresenta alta complexidade quando envolvidos também em problema de otimização. Pretende-se neste item iniciar estudos de modelagens envolvendo parametrizações usadas em problemas de otimização de forma de aerofólios. Experimentos simples como em duas dimensões são objeto desses estudos preliminares.

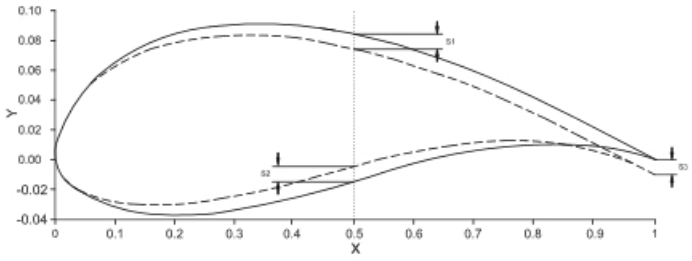


Figura 5.2: Otimização de forma de um aerofólio. Extraída de (Mohammad Zadeh et al., 2015).

Referências Bibliográficas

Barbosa, H. J. and Lemonge, A. C. (2002). An adaptive penalty scheme in genetic algorithms for constrained optimization problems. *Citeseer*, page 8.

Carvalho, d. C. R., Lemonge, A. C. d. C., Bernardino, H. S., and Hallack, P. H. (2014). Solução de problemas de otimização com restrições usando estratégias de penalização adaptativa e um algoritmo do tipo pso.

Darwin, C. (1859). *On the Origin of Species by Means of Natural Selection*. Murray, London. or the Preservation of Favored Races in the Struggle for Life.

Davis, L. (1996). *Handbook of Genetic Algorithms.* International Thomson Computer Press.

Deb, K. (2001). *Multi-Objective Optimization using Evolutionary algorithsm*. Joh Wiley & Sons, Ltd.

Deb, K. and Agrawal, R. B. (1995). Simulated binary crossover for continuous search space. *Complex Systems*, pages 115–148.

Deb, K. and Goyal, M. (1996). A combined genetic adaptive search (geneas) for engineering design. *Computer Science and Informatics*, 26:30–45.

Gandomi, A. H., Yang, X.-S., Talatahari, S., and Alavi, A. H. (2013). *Metaheuristic Applications in Structures and Infrastructures*. Elsevier Science.

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

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

Johnsen, S. (2013). Structural topology optimization : Basic theory, methods and applications. Master's thesis, Norwegian University of Science and Technology, Department of Engineering Design and Materials.

Lanes, R. M. and Greco, M. (2013). Aplicação de um método de otimização topológica evolucionária desenvolvido em script python. *Ciência Engenharia (Science Engineering Journal)*, page 11.

Lemonge, A. C., Barbosa, H. J., and Bernardino, H. S. (2015). Variants of an adaptive penalty scheme for steady-state genetic algorithms in engineering optimization. *Engineering Computations*, 32(8):2182–2215.

Mohammad Zadeh, P., Mehmani, A., and Messac, A. (2015). High fidelity multidisciplinary design optimization of a wing using the interaction of low and high fidelity models. *Optimization and Engineering*, pages 1–30.

Muyl, F., Dumas, L., and Herbert, V. (2004). Hybrid method for aerodynamic shape optimization in automotive industry. *Computers Fluids*, 33(5–6):849 – 858.

Pareto, V. (1896). *Cours déconomie politique: volume i and ii.* F. Rouge Lausanne.

Thévenin, D. and Janiga, G. (2008). *Optimization and Computational Fluid Dynamics.* Springer-Verlag Berlin Heidelberg, 1 edition.

Zadeh, L. A. (1963). Optimality and non-scalar-valued performance criteria. *Automatic Control, IEEE Transactions on*, 8(1):59–60.

Zikanov, O. (2010). *Essential Computational Fluid Dynamics.* Wiley, 1 edition.