Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas / Faculdade de Engenharia
Programa de Pós-graduação em Modelagem Computacional

**Tales Lima Fonseca**

# Algoritmo Genético com Regressão: Busca Direcionada Através de Aprendizado de Máquina.

Juiz de Fora
2017

**Tales Lima Fonseca**

# Algoritmo Genético com Regressão: Busca Direcionada Através de Aprendizado de Máquina.

Dissertação apresentada ao Programa de Pós-graduação em Modelagem Computacional da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de Mestre em Modelagem Computacional.

Orientador: Prof. D.Sc. Afonso Celso de Castro Lemonge

Coorientador: Prof. D.Sc. Heder Soares Bernardino

Coorientadora: Prof.ª D.Sc. Patricia Habib Hallak

Juiz de Fora

2017

Ficha catalográfica elaborada através do programa de geração
automática da Biblioteca Universitária da UFJF,
com os dados fornecidos pelo(a) autor(a)

Lima Fonseca, Tales.
Algoritmo Genético com Regressão: Busca Direcionada Através de Aprendizado de Máquina / Tales Lima Fonseca. -- 2017.
108 p. : il.

Orientador: Afonso Celso de Castro Lemonge
Coorientadores: Heder Soares Bernardino, Patricia Habib Hallak
Dissertação (mestrado acadêmico) - Universidade Federal de Juiz de Fora, ICE/Engenharia. Programa de Pós-Graduação em Modelagem Computacional, 2017.

1. Otimização. 2. Aprendizado de Máquina. 3. Metamodelo. I. Lemonge, Afonso Celso de Castro, orient. II. Bernardino, Heder Soares, coorient. III. Hallak, Patricia Habib, coorient. IV. Título.

**Tales Lima Fonseca**

**Algoritmo Genético com Regressão: Busca Direcionada Através de Aprendizado de Máquina.**

Dissertação apresentada ao Programa de Pós-graduação em Modelagem Computacional da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de Mestre em Modelagem Computacional.

Aprovada em:

BANCA EXAMINADORA

Prof. D.Sc. Afonso Celso de Castro Lemonge - Orientador
Universidade Federal de Juiz de Fora

Prof. D.Sc. Heder Soares Bernardino - Coorientador
Universidade Federal de Juiz de Fora

Prof.ª D.Sc. Patricia Habib Hallak - Coorientadora
Universidade Federal de Juiz de Fora

Prof. D.Sc. Leonardo Goliatt da Fonseca
Universidade Federal de Juiz de Fora

Prof.ª D.Sc. Solange Guimarães
Universidade Federal do Rio de Janeiro

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

## 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 noiva 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.

À UFJF pela sua estrutura. Aos professores e técnicos do Programa de Pós-Graduação em Modelagem Computacional pelos ensinamentos e ajuda.

A todos os meus amigos, novos ou antigos, que, de alguma forma, fizeram-se presentes nesta etapa da minha formação. Em especial, gostaria de agradecer ao Jonata Jefferson e ao Lucas Berg pelo tempo que passamos juntos, ajudando-nos.

"*Either the dead will defeat the living, in which case, all our troubles come to an end. Or life will win out. And what then? Don't fight in the North or the South. Fight every battle, everywhere, always, in your mind. Everyone is your enemy, everyone is your friend. Every possible series of events is all happening at once. Live that way and nothing will surprise you. Everything that happens will be something that you've seen before.*"

Petyr Baelish - Game of Thrones

## RESUMO

Problemas de otimização são comuns em diversas áreas. Nas engenharias, em muitas situações, os problemas de otimização eram modelados desconsiderando certas características do fenômeno estudado com a finalidade de simplificar as simulações durante o processo de busca. Contudo, com o passar do tempo, a evolução das máquinas possibilitou a modelagem de problemas de otimização com mais informações, aproximando os modelos da forma mais fidedigna possível. No entanto, uma parcela significativa desses problemas demanda um alto custo computacional para realizar as avaliações das soluções candidatas, tornando muitos deles de difícil análise e simulação. Dessa forma, o objetivo deste trabalho é a utilização de métodos de aprendizado de máquina acoplado a um algoritmo de otimização com intuito de direcionar o processo de busca de um algoritmo genético, inserindo possíveis soluções na população do algoritmo genético a cada geração com o intuito de reduzir o alto custo computacional de se encontrar as soluções ótimas. Além disso, é realizado um estudo comparativo para verificar quais métodos de aprendizado de máquina obtêm bons resultados na técnica proposta. Os experimentos são realizados em problemas de otimização com um alto custo computacional comumente encontrados na literatura.



Palavras-chave: Otimização. Algoritmos Genéticos. Aprendizado de Máquina. Metamodelo.

**ABSTRACT**

Optimization problems are common in many areas. In engineering, in many situations optimization problems were modeled disregarding certain characteristics of the studied phenomenon in order to simplify the simulations during the search process. However, over time, the evolution of the machines allowed the modeling of optimization problems with more information, approaching the models in the most reliable way possible. In this way, a significant portion of these problems requires a high computational cost to perform the evaluations of candidate solutions, making many of them difficult to analyze and simulate. Thus, the objective of this work is the use of machine learning methods coupled with an optimization algorithm with the purpose of directing the search process of a genetic algorithm, inserting new good quality solution into the population at each generation with the intention of reducing the high computational cost of finding the optimal solutions. In addition, a comparative study is carried out to verify which machine learning methods obtain good results in the proposed technique. The experiments are performed on optimization problems with a high computational cost commonly found in the literature.

Key-words: Optimization. Genetic Algorithm. Machine Learning. Surrogate Models.

## LISTA DE ILUSTRAÇÕES

## LISTA DE TABELAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| AG | Algoritmo Genético |
| ANN | Rede Neural Artificial |
| APM | Método de Penalização Adaptativa |
| CFD | Dinâmica dos Fluídos Computacional |
| DM | Mineração de Dados |
| ERM | Minimização do Risco Empírico |
| KDD | Descoberta de Conhecimento |
| MAE | Erro Absoluto Médio |
| MLP | Perceptron Multicamadas |
| RBF | Função de Base Radial |
| RBFN | Rede Neural de Função de Base Radial |
| SBX | Recombinação Binário Simulado |
| SRM | Minimização do Risco Estrutural |
| SVM | Máquinas de Vetores Suporte |
| SVR | Máquinas de Vetores Suporte para Regressão |

## SUMÁRIO

# 1 INTRODUÇÃO

No processo de criação de projetos de engenharia, o objetivo final de todas as decisões tomadas pelos profissionais (engenheiros) é de minimizar o esforço necessário ou maximizar o benefício desejado (Rao e Rao, 2009). Estes requisitos apresentados no projeto podem ser atingidos através da resolução de um problema de otimização previamente modelado.

Os processos de otimização, juntamente com toda a experiência do projetista ou do conjunto de projetistas, auxiliam de maneira considerável na busca das melhores soluções pretendidas para os projetos desenvolvidos. Desta maneira, são obtidas configurações para o projeto que exploram ao máximo o objetivo esperado pelo mesmo.

Os problemas de otimização são comuns em diversas áreas. Entretanto, com o foco para a engenharia estrutural e dinâmica dos fluidos computacional, particularmente abordados nesta dissertação, o processo de otimização pode requerer a a simulação e resolução de modelos muito complexos repetidas vezes. Devido ao alto nível de detalhamento e seu alto custo computacional, estes problemas apresentam uma grande complexidade para alcançar as suas soluções ótimas ou até mesmo as soluções próximas das ótimas em um curto período.

Nas engenharias em geral, os problemas de otimização eram modelados de forma simplificada, desconsiderando certas características do fenômeno estudado com a finalidade de reduzir a complexidade dos problemas e o custo computacional durante o processo de busca. Um exemplo destas simplificações pode ser observado no trabalho de Ahmed et al. (1984), em que é realizado um estudo sobre coeficiente de arrasto em um modelo simplificado de um dado veículo.

Entretanto, com o passar do tempo, a evolução dos recursos computacionais possibilitou que problemas de otimização através de modelos mais fidedignos fossem resolvidos, como, por exemplo, a otimização da forma aerodinâmica de configurações complexas de aeronaves (Reuther et al., 1996). Por outro lado, apesar de ser possível trabalhar atualmente em modelos mais realísticos, ainda existe um alto custo computacional na obtenção das melhores soluções do problema.

## 1.1 Motivação

Em um processo de busca pelas soluções ótimas de um problema de otimização, além das diversas dificuldades caracterizadas pelo problema, faz-se necessária a escolha de um método de busca ideal.

Os métodos clássicos de otimização, normalmente determinísticos, apresentam uma série de dificuldades e restrições para determinados tipos de problema de otimização. Entre

elas, de modo geral, pode-se citar (Lemonge, 1999; Albrecht, 2005):

- dificuldade de identificar soluções ótimas globais;
- exigência de que as funções objetivo sejam contínuas e diferenciáveis;
- dificuldade em problemas que envolvem variáveis contínuas e discretas;
- ineficácia quando o problema é multiobjetivo;
- exigência de ponto de partida factível;
- inadequados para programação paralela;
- domínio de aplicação restrito;

As meta-heurísticas populacionais, massivamente usadas atualmente, levam vantagem sobre os métodos clássicos por não apresentarem estas dificuldades e restrições. Apesar disso, elas se deparam com a dificuldade de avaliar muitas soluções candidatas com um alto custo computacional, processo necessário de convergência para uma solução desejada.

Dessa forma, faz-se necessário estudo de técnicas para viabilizar a busca por soluções ótimas desses problemas de otimização com um menor custo computacional possível.

## 1.2 Justificativa

É de fundamental importância que qualquer projeto consiga garantir um bom desempenho durante a sua vida útil, independentemente do grau de complexidade do mesmo. Além de tudo, o processo de busca de projetos otimizados é altamente desejável, visto que possibilita a minimização dos custos dos materiais utilizados, sem perda de segurança e qualidade no projeto.

A busca pela otimização em projetos deve estar presente a todo momento, pois os recursos naturais que nós humanos utilizamos para a nossa sobrevivência são limitados no planeta em que vivemos e o desperdício pode trazer consequências graves no futuro e um custo maior na obra final. Devido a isso, as questões ambientais são de suma importância neste processo, nas condições referentes à diminuição do uso exagerado de matéria-prima cada vez mais escassa.

Na engenharia, ainda existem problemas de otimização com um alto grau de complexidade que, até hoje, foram resolvidos por meio de consideráveis simplificações no modelo real ou simplesmente não foram resolvidos. O principal motivo é o alto custo computacional destes modelos e a falta de técnicas capazes de realizar a otimização em um baixo intervalo de tempo.

## 1.3 Objetivos

Uma alternativa disponível na literatura são os modelos de substituição ou metamodelos (*Surrogate Models*), em que a função objetivo é substituída por uma aproximação que demanda menos tempo de processamento para ser calculada (Qian et al., 2006). Esses modelos buscam evitar a utilização de avaliações que sejam computacionalmente custosas a fim de obter: (i) melhores resultados com o mesmo tempo de processamento, ou (ii) resultados similares mais rapidamente.

Pretende-se, durante este trabalho, desenvolver um método híbrido de otimização formado por uma técnica de aprendizado de máquina que é combinada com um algoritmo de otimização com o intuito de direcionar o processo de busca de um algoritmo genético. Para isso, o algoritmo de otimização local refinará as melhores soluções encontradas pelo algoritmo genético a cada iteração utilizando modelos de substituição.

Esta abordagem difere da comumente encontrada na literatura (El-Beltagy et al., 1999; Booker et al., 1999; Jin et al., 2002; Zhou et al., 2007; Wortmann et al., 2015), pois não realiza a substituição do cálculo da aptidão usada no algoritmo genético. O modelo aproximado é usado apenas na busca local, com o intuito de direcionar a busca do algoritmo genético. Além disso, pretende-se analisar os métodos de aprendizado de máquina a fim de determinar qual deles fornece melhor desempenho ao algoritmo proposto.

O objetivo deste trabalho é agilizar o processo de busca usando a técnica proposta em em problemas com alto custo computacional como, por exemplo, problemas de otimização estrutural e em dinâmica dos fluidos computacional. Os resultados numéricos apresentados mostraram que os objetivos pretendidos foram alcançados.

## 1.4 Organização do texto

Esta dissertação está organizada em sete capítulos, sendo esta introdução o primeiro deles. O Capítulo 2 é destinado à otimização, apresentando os conceitos teóricos do mesmo. No Capítulo 3, são apresentados todos os métodos de otimização utilizados para o desenvolvimento do trabalho, juntamente com um método de penalização para tratamento de restrições.

O Capítulo 4 aborda o conceito de metamodelos, uma revisão bibliográfica sobre o assunto e os métodos de regressão que são utilizados na técnica proposta. Já o Capitulo 5 expõe a ideia do método hibrido de otimização proposto, com detalhes da construção de sua estrutura. No Capítulo 6, são apontados os experimentos numéricos realizados durante o desenvolvimento deste trabalho e, por fim, as conclusões são apresentadas no Capítulo 7.

# 2 OTIMIZAÇÃO

## 2.1 Introdução

A otimização é um ramo da matemática e vem sendo aplicada em diversas áreas, tendo este trabalho como foco alguns dos problemas comumente encontrados na engenharia estrutural. A otimização pode ser definida como um processo de busca pelos valores máximos ou mínimos de uma função (ou várias funções), cujas variáveis satisfaçam determinadas restrições na forma de igualdades ou desigualdades (Arora, 1989; Zörnig, 2014). O conjunto de pontos que maximiza ou minimiza uma função (ou várias funções) é chamado de soluções ótimas.

A maximização de uma função $f(\mathbf{x})$ pode ser convertida em uma minimização de $g(\mathbf{x})$, onde $g(\mathbf{x}) = -f(\mathbf{x})$. Com isso, maximizar uma função $f(\mathbf{x})$ é o mesmo que minimizar $-f(\mathbf{x})$. Estas funções são conhecidas como funções objetivo e representam a quantidade que se deseja minimizar ou maximizar no problema. A Figura 1, extraída e adaptada de (Rao e Rao, 2009), mostra a situação em que o ponto $\mathbf{x}^*$ gera o valor mínimo de uma função $f(\mathbf{x})$ e o valor máximo do negativo da função, $-f(\mathbf{x})$.

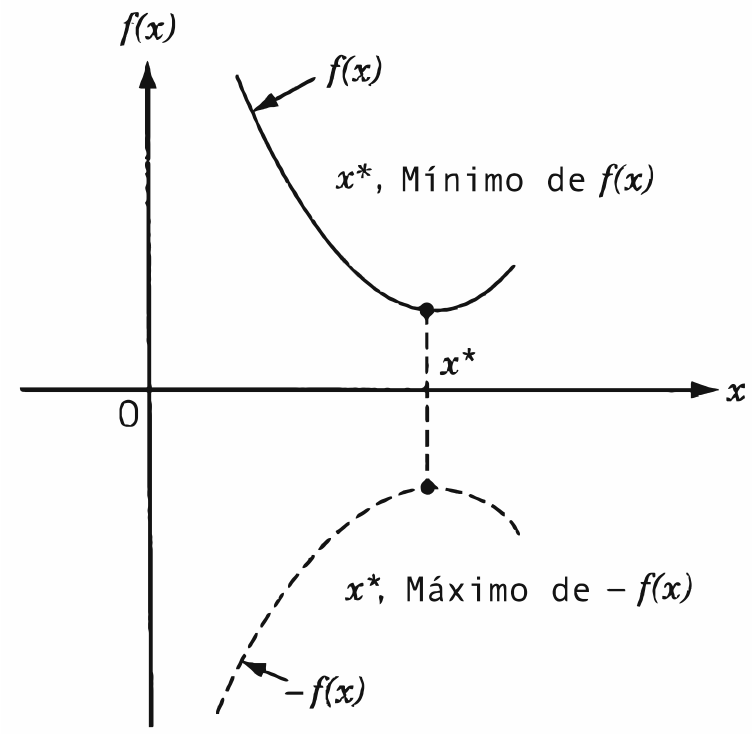



Figura 1 – Minimizar $f(\mathbf{x})$ é o mesmo que maximizar $-f(\mathbf{x})$.

## 2.2 Formulação do problema de otimização

Um problema de otimização, de modo geral, pode ser elaborado formalmente do seguinte modo:

$$\begin{aligned} &\min_{\mathbf{x}}/\max \quad f_i(\mathbf{x}) \qquad i = 1, 2, \ldots, n_f \\ &\text{sujeito a} \\ &\qquad g_j(\mathbf{x}) \leq 0, \qquad j = 1, 2, \ldots, n_g \\ &\qquad h_k(\mathbf{x}) = 0, \qquad k = 1, 2, \ldots, n_h \\ &\qquad x_l^L \leq x_i \leq x_l^U, \qquad l = 1, 2 \ldots, n \end{aligned}$$

onde o vetor $\mathbf{x}$ é designado como o vetor de incógnitas ou vetor de variáveis de projeto, $f_i(\mathbf{x})$ são as funções objetivo e $g_j(\mathbf{x})$ e $h_k(\mathbf{x})$ são as restrições de desigualdade e igualdade, respectivamente. As funções $f_i(\mathbf{x})$, $g_i(\mathbf{x})$ e $h_i(\mathbf{x})$ podem ser funções lineares ou não lineares do vetor de variáveis de projeto. Os limites das variáveis são determinados através de $x_i^L$ e $x_i^U$, representando respectivamente o valor inferior e superior da variável $x_i$. Nas expressões, $n$, $n_f$, $n_g$ e $n_h$ são o número de variáveis de projeto, número de funções objetivo, número de restrições de desigualdade e igualdade, respectivamente.

Comumente ocorre a transformação das restrições de igualdade em restrições de desigualdade na seguinte forma:

$$|h_k(x)| - \varepsilon \leq 0, \ k = 1, 2, \ldots, n_h$$

onde a tolerância $\varepsilon$ é suficientemente pequena, de modo que esta transformação não afeta significativamente a qualidade da solução do problema.

A otimização de um problema com restrições é mais complexa do que a de um problema sem restrições e pode requerer estratégias específicas do resolvedor para que essas sejam satisfeitas. Um vetor $\mathbf{x}$ 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 Figura 2 apresenta um espaço de busca bidimensional, contendo duas regiões, uma factível e a outra infactível. O papel das restrições de igualdade e desigualdade em um processo de otimização é de determinar os limites de viabilidade do projeto, separando as regiões factíveis das infactíveis (Rao e Rao, 2009).

A solução do problema de minimização/maximização é dita solução ótima. Caso exista mais de uma solução ótima, estas são ditas soluções ótimas alternativas. No caso da minimização, o conjunto de variáveis de projeto que proporcionam o menor valor da função objetivo entre todas as combinações possíveis dos valores das variáveis é chamado de mínimo global.

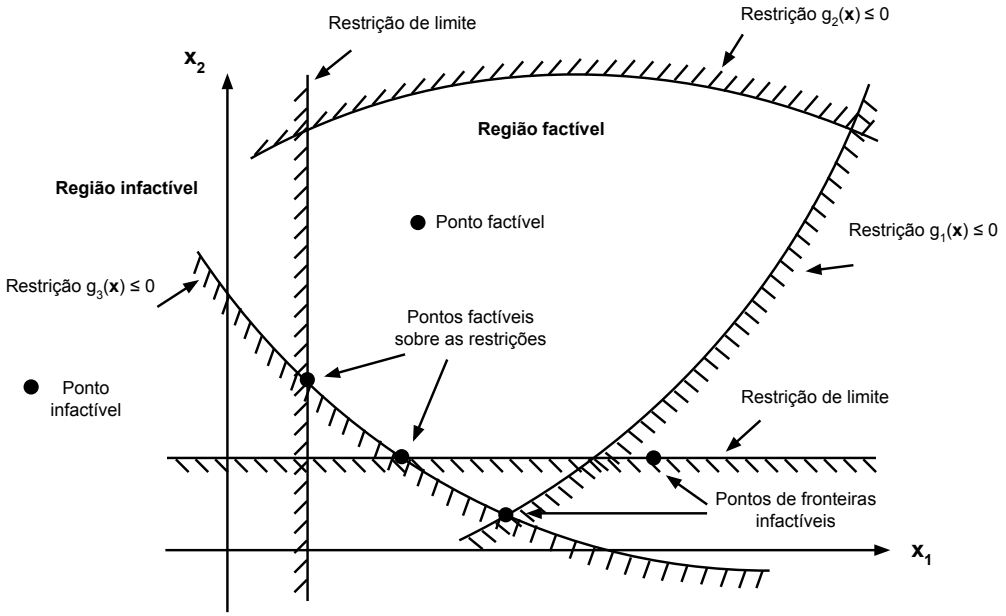



Figura 2 – Regiões do espaço de busca com restrições, extraída e adaptada de (Rao e Rao, 2009).

Em geral, é difícil afirmar que tal valor é global devido à possibilidade de existência de outros mínimos, locais ou globais, no espaço de busca. Esta possibilidade de existência dificulta a verificação da característica global no valor analisado. O que se pode afirmar é que o valor encontrado é mínimo numa vizinhança do espaço de busca. A Figura 3 apresenta um exemplo de dois pontos, um mínimo local e o outro mínimo global, para a função $f(x) = \cos(3\pi x)/x$, onde $0.1 \leq x \leq 1.1$.

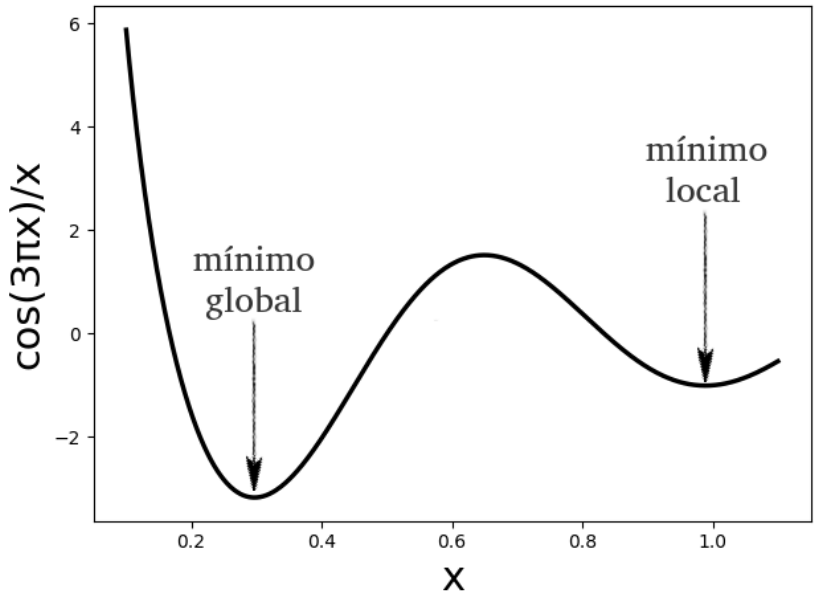



Figura 3 – $f(x) = \cos(3\pi x)/x, 1 \leq x \leq 1$

Pode-se dizer que vários tipos de problemas de otimização podem ser construídos de acordo com determinadas escolhas, assim como: as funções objetivo, restrições e as variáveis de projeto utilizadas. Estes problemas de otimização podem ser classificados de acordo com estas características. A Tabela 1 apresenta, de maneira simplificada, algumas das classificações existentes em um problema de otimização.

Tabela 1 – Classificação dos problemas de otimização, extraída e adaptada de (Gandomi et al., 2013).

| Base da Classificação | Categoria |
|---|---|
| Número de Variáveis de Projeto | Única Variável<br>Multivariável |
| Número de Funções Objetivo | Único Objetivo<br>Multiobjetivo<br>Nenhum |
| Presença de Restrições | Sem Restrições<br>Com Restrições |
| Características de Restrições e Funções Objetivo | Linear<br>Não Linear |
| Tipo de Variáveis de Projeto | Discreta<br>Contínua<br>Mista. |
| Natureza das Variáveis e Dados de Entrada | Determinística<br>Probabilística |

## 2.3 Uma breve classificação dos métodos de otimização

Não se pretende aqui nenhuma tentativa de esgotar a apresentação e discussão de todos os métodos de otimização disponíveis para as suas aplicações. Na Figura 4, extraída e adaptada de (Gandomi et al., 2013), mostra-se uma classificação bastante informativa acerca dos métodos mais usuais e consolidados na literatura. A técnica de otimização principal utilizada nesta dissertação são os Algoritmos Genéticos.

## 2.4 Otimização em engenharia

Na engenharia, de modo geral, existem diversos problemas que necessitam da realização de algum processo de busca para a obtenção de informações que, até então, não eram conhecidas pelo especialista. Isso pode ocorrer com frequência de acordo com a complexidade do problema tratado. Deste modo, pode-se afirmar que a engenharia, de uma forma abrangente, está fortemente presente na otimização.

A otimização na engenharia pode ser vista basicamente como o processo de encontrar projetos com o melhor desempenho possível e/ou menor custo, garantindo que todas as condições do projeto sejam satisfeitas (Arora, 1989; Ramteen Sioshansi, 2017).

Conforme apresentado em Rao e Rao (2009), indicando o vasto alcance do tema, algumas aplicações típicas em engenharia são, por exemplo:

- Projeto de aeronaves e estruturas aeroespaciais com peso mínimo (Zhu et al., 2016);
- Busca das melhores trajetórias de veículos espaciais (Walmsley et al., 2016);
- Projeto de estruturas de engenharia civil, como pórticos, fundações, pontes, torres, chaminés e barragens com um custo mínimo (Estes e Frangopol, 2001);
- Projeto de estruturas de menor peso para terremotos, vento e outros tipos de carregamento aleatório (Salajegheh e Heidari, 2005);
- Projeto de sistemas de recursos hídricos para máximo benefício (Savic e Walters, 1997);
- Projeto de estruturas com comportamento plástico ideal (Prager e Shield, 1967);
- Projeto ótimo de acoplamentos, cames, engrenagens, máquinas-ferramenta e outros componentes mecânicos (Fox e Willmert, 1967);
- Seleção de condições de usinagem em processos de corte de metais para custo mínimo de produção (Mukherjee e Ray, 2006);
- Projeto de equipamentos de manuseio de materiais, como transportadores, caminhões e guindastes, para custo mínimo (Cardarelli e Pelagagge, 1995);
- Projeto de bombas, turbinas e equipamentos de transferência de calor para máxima eficiência (Fesanghary et al., 2009);
- Projeto ótimo de máquinas elétricas, como motores, geradores e transformadores(Andersen, 1967);
- Projeto ótimo das redes elétricas (Knight, 1960);
- A rota mais curta realizada por um vendedor que visita várias cidades durante uma turnê (Caixeiro Viajante) (Dorigo e Gambardella, 1997);
- Planejamento, controle e agendamento ótimo de produção (Shi et al., 2011);
- Análise de dados estatísticos e construção de modelos empíricos a partir de resultados experimentais para obter a representação mais precisa do fenômeno físico (Whelan e Goldman, 2001);
- Projeto ótimo de equipamentos e plantas de processamento químico (Grossmann e Sargent, 1978);

- Projeto ótimo de redes de tubulação para indústrias de processo (Soliman e Murtagh, 1982);
- Seleção de um local ótimo para a instalação de uma indústria (Kuo, 2011);
- Planejamento de manutenção e substituição de equipamentos para reduzir custos operacionais (Sachdeva et al., 2008);
- Controle de inventário (Adida e Perakis, 2006);
- Alocação de recursos ou serviços entre várias atividades para maximizar algum tipo de benefício (Hegazy, 1999);
- Controle dos tempos de espera, ociosos e de filas em linhas de produção para reduzir os custos (Patrick e Puterman, 2008);
- Planejamento da melhor estratégia para obter o máximo lucro na presença de um concorrente (Shiau e Michalek, 2009);
- Projeto ótimo dos sistemas de controle (Balakrishnan e Boyd, 1993).

Em problemas de otimização na engenharia, é comum que os problemas possuam mais do que uma função objetivo, representando assim um problema de otimização multiobjetivo. Um exemplo deste cenário é apresentado em um Trocador de Calor (Thévenin e Janiga, 2008), em que se deseja maximizar a troca de calor e, ao mesmo tempo, minimizar a perda de pressão.

Também é comum que os problemas de otimização na engenharia possuam restrições oriundas das especificações de projeto, como, por exemplo, fatores de segurança. Em otimização estrutural, por exemplo, as restrições que estão presentes são os limites para as tensões internas nos elementos, deslocamentos máximos permitidos e outros (Gelle et al., 2000).

Para alguns problemas, as funções objetivos e as restrições podem se encontrar de forma implícita e, para obter o seu valor, faz-se uso de simuladores. Estes acabam demandando um alto custo computacional na maioria destes problemas (Carson e Maria, 1997).

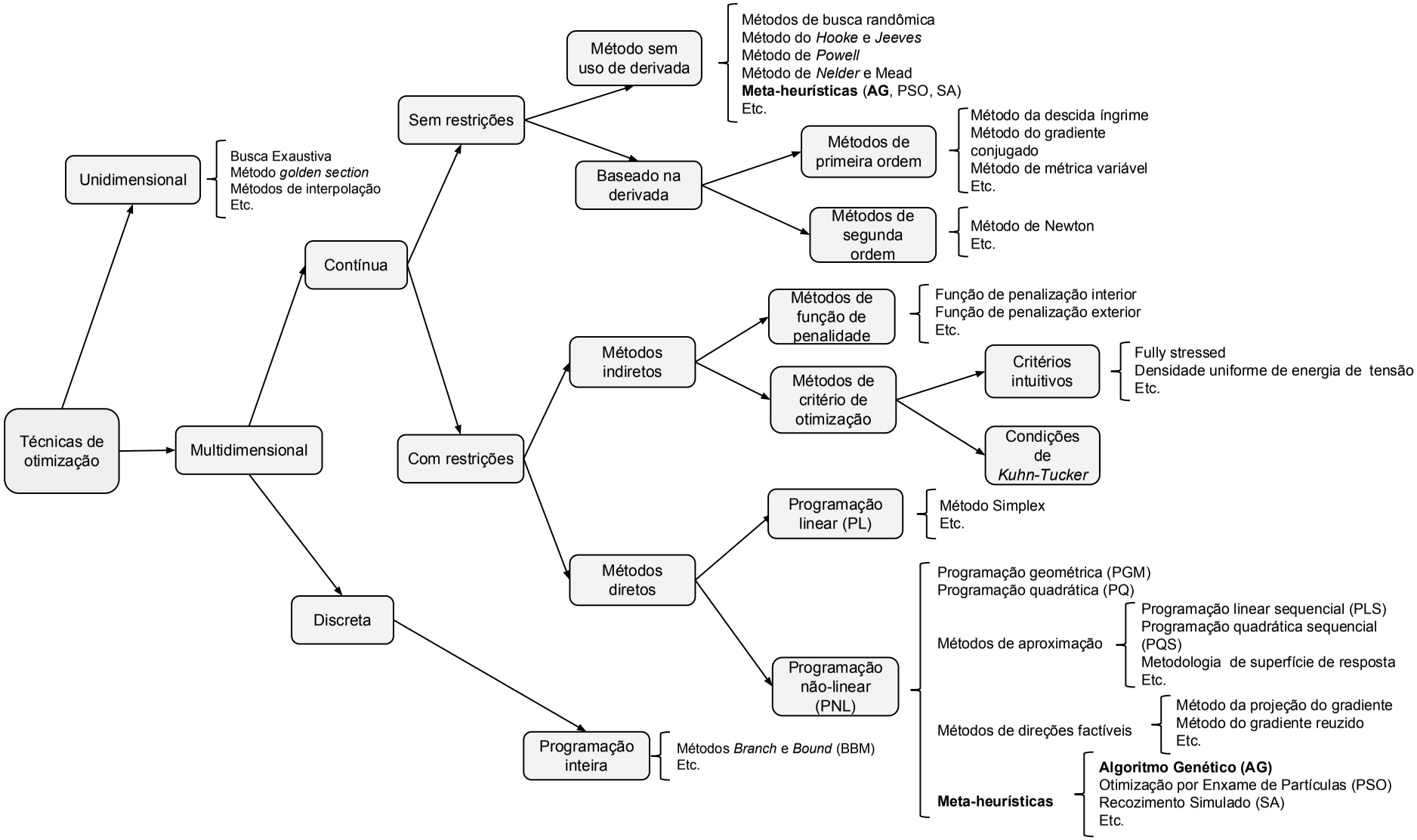



Figura 4 – Classificação das técnicas de otimização numérica, extraída e adaptada de (Gandomi et al., 2013).

# 3 ALGORITMOS DE BUSCA

## 3.1 Introdução

Neste capitulo, são apresentados dois algoritmos de busca, um Algoritmo Genético (AG) do tipo geracional e o L-BFGS-B (*Limited-memory Broyden–Fletcher–Goldfarb–Shanno Bound-constrained*), que podem ser utilizados para procura da solução ótima de problemas de otimização. Além disso, é apontado um método de penalização adaptativa para o tratamento das restrições dos problemas de otimização.

## 3.2 Algoritmo Genético

Conforme foi apresentado por Darwin na Teoria da Evolução (Darwin, 1859), a luta pela sobrevivência dos seres vivos é um fator de grande importância para a seleção natural. Deste modo, a maioria das transformações apresentadas em um ser vivo tem como único propósito tentar lhe fornecer vantagens na sua luta pela sobrevivência. Algumas destas transformações podem ser benéficas e proporcionar 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 onde se apresentam recursos limitados, existe uma competição entre todos os seres vivos presentes, acarretando assim a necessidade de um equilíbrio sobre a quantidade de indivíduos que convivem em um mesmo ambiente. Isso leva a pensar que, as condições ambientais restringem o tamanho máximo da população no ambiente, de modo que uma população não pode crescer indefinidamente se houver escassez de recursos. Caso os recursos fossem ilimitados, não existiria uma seleção natural, contradizendo que os seres lutam para poder sobreviver e se reproduzir, mantendo as suas heranças genéticas. Os meios de controle populacional e de seleção natural em uma região são as enfermidades, os predadores e a quantidade de recursos existentes, e somente os seres mais aptos conseguem sobreviver.

A seleção natural leva a diversas extinções, pois a luta constante pela sobrevivência tem como objetivo selecionar apenas os indivíduos mais aptos para a sua manutenção na população e, como consequência, ocorre a redução das 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 lhes dar vantagens acaba acarretando maiores chances de se reproduzirem, e essa seleção deixa a espécie cada vez mais apta, já que a natureza opta apenas por beneficiar os bons desempenhos.

Toda mudança apresentada em uma espécie é inicialmente local, oriunda de um indivíduo. Se a mudança apresentada pelo indivíduo estimular uma melhoria no processo de sobrevivência, em um pequeno intervalo de tempo, surgiria um pequeno grupo de

reprodução que espalharia estas novas características por uma grande região. Isso leva à conclusão de que pequenas modificações se acumulam caso sejam vantajosas para o indivíduo e acabam se realçando de geração em geração para toda a espécie, levando esses indivíduos a se adaptarem ao *habitat* caso possuam características que levem à sua sobrevivência. Os descendentes de uma espécie que sofreram modificações vantajosas podem se reproduzir e habitar diferentes regiões, podendo ocorrer luta por sobrevivência com outros seres que já habitam a região.

Com o passar do tempo, a Teoria da Evolução foi sendo construída nas ideias apresentadas por Darwin sobre a seleção natural, juntamente com a genética, criando assim o princípio básico da Genética Populacional, de que 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 realizadas 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 possível 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 tenham mais chance de 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 para soluções ótimas locais ou globais.

Os Algoritmos Genéticos (AGs) foram imaginados e desenvolvidos por John Holland nos anos 70, na Universidade de Michigan (Holland, 1975, 1995). Ele acreditava na possibilidade de criação de um algoritmo de otimização capaz de resolver problemas complexos, inspirando-se na Teoria da Evolução. Em sua pesquisa, ele se concentrou em dois pontos fundamentais: abstrair e explicar rigorosamente os processos adaptativos dos sistemas naturais e projetar *software* para simular sistemas artificiais, mantendo os mecanismos importantes dos sistemas naturais (Goldberg, 1989).

Os AGs podem ser definidos como procedimentos de busca inspirados na genética e seleção natural das espécies (Koza, 1992). Eles adotam o conceito de sobrevivência por meio de testes de classificação em um sistema artificial. Os testes de classificação realizados através da informação das aptidões dos indivíduos na população têm como objetivo identificar os melhores e os piores indivíduos pertencentes a uma população. Os operadores genéticos dos AG's são inspirados na natureza, formando um mecanismo eficiente de busca evolutiva.

Holland (1975) adotou a semântica da biologia em seu algoritmo e assim elencou três informações importantes na construção de um AG: o cromossomo, o indivíduo e a população. Ele buscava uma maneira de codificar cada possível solução de um problema de otimização em um cromossomo que representasse um indivíduo. Logo, uma população

pode ser vista como um conjunto de cromossomos. O alfabeto binário foi a sua primeira representação e é considerada a mais clássica. O alfabeto binário é composto de 0's e 1's, representando artificialmente um "código genético".

O conjunto formado por indivíduos define uma população e estes indivíduos são codificados por seus cromossomos que serão submetidos a avaliações submetidos a avaliações que medem seus desempenhos. Os indivíduos serão sempre classificados por algum critério baseado nos seus desempenhos e irão se reproduzir formando uma nova geração, parte do mecanismo de evolução das soluções no algoritmo.

Após realizar a classificação, três operações básicas são efetuadas sobre a população: a seleção, a recombinação e a mutação. Os genitores (indivíduos pais) são escolhidos pelo operador de seleção e sobre eles atuará o operador de recombinação e, posteriormente, o operador de mutação agirá sobre os indivíduos filhos. Imaginando-se que os indivíduos 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 cada vez melhor.

Um AG pode ser identificado por quatro elementos fundamentais (Mitchell, 1998):

1. Populações de cromossomos;
2. Seleção de acordo com a aptidão;
3. Recombinação para produzir novos cromossomos;
4. Mutação aleatória nos novos cromossomos;

Um pseudocódigo que representa um AG é apresentado no Algoritmo 1.

**Algoritmo 1:** Pseudocódigo de um Algoritmo Genético.

```
1  g ← 0;
2  Gera uma população inicial P_g com N_POP indivíduos;
3  Avalia cada indivíduo da população P_g;
4  enquanto g < N_G faça
5      Seleciona N_POP indivíduos em P_g, formando a população G_g;
6      Aplica operador de recombinação (probabilidade p_c) em G_g;
7      Aplica operador de mutação(taxa p_m) em G_g;
8      Avalia cada indivíduo da população G_g;
9      Seleciona as k melhores soluções de P_g formando K_g;
10     Combina K_g com G_g, formando uma população mista Q_g;
11     Ordena a população Q_g;
12     Seleciona os N_POP melhores indivíduos de Q_g e copia para P_{g+1};
13     g ← g + 1;
```

A utilização de AGs para solução de problemas da engenharia se torna muito atrativa, visto que, diferente dos métodos clássicos que têm aplicações limitadas, os mesmos conseguem identificar soluções globais, em geral, podendo convergir para a solução ótima do problema e não para uma solução local.

Além disso, eles não possuem domínio de aplicação restrito e podem resolver problemas que envolvem variáveis contínuas, discretas e mistas. As funções objetivos não precisam ser contínuas ou diferenciáveis e não é preciso informar um ponto factível para iniciar a busca. A utilização da programação paralela em algoritmos genéticos é recomendada. Com isso, pode-se concluir que a técnica de otimização é viável para ser utilizada em problemas encontrados não somente na engenharia estrutural, mas também em diversas áreas.

### 3.2.1 Codificação

Para representar as possíveis soluções do problema que se deseja otimizar, é preciso definir como os indivíduos serão codificados. A codificação depende das características do problema. Entre os vários *"alfabetos"* utilizados neste procedimento, os que se destacam são os referentes à codificação binária e à codificação real.

A utilização da codificação binária é comum e recomendada em problemas com variáveis discretas, enquanto a codificação real é comum e recomendada em problemas com variáveis contínuas. A Tabela 2 apresenta um exemplo da codificação real e da codificação binária. Cada conjunto de variáveis ($x_1$, $x_2$, $x_3$, $x_4$ e $x_5$) de uma codificação representa uma possível solução do problema.

Tabela 2 – Codificação real e binária.

| **Codificação Real** | | | | |
|---|---|---|---|---|
| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
| 11.5 | 25.7 | 15.0 | 67.57 | 2.0 |

| **Codificação Binária** | | | | |
|---|---|---|---|---|
| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
| 1101 | 111 | 1011 | 101 | 100101 |

Para a codificação binária, é necessário um procedimento de decodificação para recuperar os valores em números reais ou inteiros das variáveis na base 10. Basicamente, é feito um mapeamento da codificação binária para um valor inteiro e/ou real. A Tabela 3 apresenta um exemplo de mapeamento da codificação binária para o caso inteiro e real.

É importante notar que a escolha do número de bits para cada uma das variáveis e a sua respectiva decodificação depende de cada problema.

Tabela 3 – Mapeamento da codificação binária.

| **Mapeamento Binário para Inteiro** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Binário** | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| **Inteiro** | -3 | -2 | -1 | 0 | 1 | 2 | 3 | 4 |

| **Mapeamento Binário para Real** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Binário** | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| **Real** | -2.5 | -2.0 | -1.5 | -1.0 | -0.5 | 0.0 | 0.5 | 1.0 |

Michalewicz (1992) realizou uma análise detalhada sobre a utilização da codificação binária ou a codificação real. Um caso de teste é estudado utilizando duas implementações. Os experimentos realizados indicaram que a codificação real é mais rápida, mais consistente de execução para execução e fornece uma maior precisão, principalmente em problemas com grandes domínios em que a codificação binária exigiria uma representação proibitivamente longa.

A representação em codificação real torna mais fácil a criação de outros operadores que possam incorporar conhecimento específico do problema, visto que está mais próxima do espaço do problema. Isto é essencial no tratamento de restrições não triviais específicas do problema.

### 3.2.2 População inicial

O primeiro procedimento do AG, depois de definida a codificação ideal para o problema, é a criação de uma população inicial que represente as possíveis soluções do problema a ser avaliado.

O processo de criação da população inicial é importante, visto que isso pode interferir no processo de busca e influenciar a solução final. Se a população inicial de AG é boa, então o algoritmo tem possibilidades melhores de encontrar uma boa solução para o problema (Zitzler et al., 2000; Burke et al., 2004).

O processo mais comum e mais geral de se criar uma população inicial é de forma aleatória, através de funções pseudorrandômicas. A criação da população inicial de maneira aleatória pode trazer vantagens ou desvantagens, dependendo do problema a ser resolvido (Back, 1996). A vantagem da aleatoriedade é espalhar bem os indivíduos pelo espaço de busca, favorecendo a exploração do algoritmo. Como desvantagem, em problemas com restrições, toda as soluções da população podem ser infactíveis e o processo de busca pode demorar a encontrar uma solução factível.

O tamanho da população é um parâmetro do AG e não do problema a ser otimizado. Este parâmetro deve ser escolhido de forma adequada para cada problema (Goldberg et al., 1991, 2014).

### 3.2.3 Operador de seleção

No AG, os indivíduos são selecionados de uma população para serem pais no processo de recombinação. A questão é como devem ser selecionados estes indivíduos. Logo, o objetivo do operador de seleção é escolher os indivíduos que servirão de pais no processo de reprodução.

De acordo com a Teoria da Evolução de Darwin, os melhores indivíduos da população devem ser escolhidos para sobreviver e gerar filhos. Existem diversas formas de se realizar a seleção, sendo alguns dos operadores de seleção mais utilizados:

- Seleção por Roleta:

  Na seleção por roleta, cada indivíduo da população é representado na roleta proporcionalmente ao seu índice de aptidão. Assim, para indivíduos com alta aptidão, é dada uma porção maior da roleta, enquanto aos indivíduos de aptidão mais baixa é dada uma porção relativamente menor. Por exemplo, pretende-se maximizar f(x) e existem três indivíduos na sua população com os seguintes valores para f(x): 150, 20 e 10. A Figura 5 apresenta este exemplo da distribuição dos indivíduos na roleta.


Figura 5 – Seleção por Roleta.

- Seleção por Ranking:

  Na seleção por ranking, primeiramente, ordena-se a população do melhor para o pior e então se atribui a cada indivíduo um valor de adequação determinado pela sua posição. O pior indivíduo terá adequação igual a 1, o segundo pior 2 e assim por diante, de forma que o melhor terá adequação igual a N, onde N é o tamanho da população.

Por exemplo, pretende-se maximizar f(x) e existem três indivíduos na sua população com os seguintes valores para f(x): 150, 20 e 10. A Figura 6 apresenta um exemplo da distribuição dos indivíduos na roleta com ranking.


Figura 6 – Seleção por Ranking.

- Seleção por Torneio:

  Na seleção por Torneio, um número $n$ de indivíduos da população é escolhido aleatoriamente para formar uma subpopulação temporária. Deste grupo, o indivíduo de maior adequação será selecionado.

- Outros.

Ao se realizar o processo de seleção, deseja-se encontrar o conjunto de pais da geração atual que irá gerar os novos indivíduos. Quando a nova geração é formada somente pelos novos indivíduos, corre-se o risco de se perder um bom indivíduo, mesmo que alguns dos novos indivíduos sejam exatamente iguais a alguns dos pais selecionados (quando a recombinação e mutação não são aplicadas ao individuo).

Para que os bons indivíduos não sejam perdidos a cada geração, pode-se selecionar uma porcentagem dela que será mantida na próxima geração. Esta prática é conhecida como elitismo (Davis, 1996). Isso garante que, durante o passar das gerações, a melhor solução conhecida não seja perdida no processo de busca.

### 3.2.4 Operador de recombinação

Este operador é considerado o operador genético predominante. Por meio de recombinação, são criados novos indivíduos, misturando características dos indivíduos

“pais". Trechos das características de um indivíduo são trocados pelo trecho equivalente do outro. O resultado desta operação é um indivíduo que poderia (ou não) combinar as melhores características dos indivíduos usando como base, havendo uma chance de melhorar a avaliação do individuo em relação aos pais. Algumas das recombinações mais utilizadas são (Goldberg, 1989):

- Recombinação de um Ponto:

  Este operador é o mais clássico dentro dos algoritmos genéticos para codificação binária. Sua função é selecionar aleatoriamente um ponto localizado entre os limites dos cromossomos selecionados como pais e promover a recombinação de material entre eles, gerando dois filhos. Os exemplos a seguir esclarecem esta operação:

$$\begin{array}{llll} Pai-1 & 11111|11 & Filho-1 & 11111|00 \\ Pai-2 & 00000|00 & Filho-2 & 00000|11 \end{array}$$

- Recombinação de dois Pontos:

  Analogamente à recombinação de um ponto, a recombinação de dois pontos tem a função de selecionar aleatoriamente dois pontos localizado entre os limites do cromossomos selecionados como pais e promover a recombinação de material entre eles, gerando dois filhos. Os exemplos a seguir esclarecem esta operação:

$$\begin{array}{llll} Pai-1 & 10|110|01101 & Filho-1 & 10|011|01101 \\ Pai-2 & 00|011|10111 & Filho-2 & 00|110|10111 \end{array}$$

- Recombinação de $n$ Pontos:

  Este é uma generalização natural da recombinação de um ou dois pontos. Aqui, escolhem-se aleatoriamente $n$ pontos pertencentes ao intervalo dos cromossomos selecionados como pais e troca-se o material genético.

- Recombinação Uniforme:

  Neste caso, são gerados números aleatórios uniformes que indicarão a troca de material genético. Se a “máscara” tiver o *bit* 1, haverá troca do *bit* correspondente à sua posição. Se o *bit* for 0, nada acontecerá.

$$\begin{array}{llll} Pai-1 & 11111111 & Filho-2 & 10010110 \\ & \text{“}Máscara\text{”} & 01101001 & \\ Pai-2 & 01001100 & Filho-2 & 00100101 \end{array}$$

- Recombinação Binária Simulada:

  A Recombinação Binária Simulada (*Simulated Binary Crossover* - SBX) (Deb e Agrawal, 1995) é uma recombinação para codificação real baseada nas propriedades apresentadas pela recombinação de um ponto em uma codificação binária. Suas propriedades são:

  - **Média:** A média dos pais é igual à média dos filhos após a operação de recombinação;
  - **Fator de Propagação $\beta$ :** A maioria dos eventos de recombinação resultam em um fator de propagação de $\beta \approx 1$, isto é, os filhos tendem a ser parecidos com os seus pais.

  Para manter a propriedade média, os valores dos filhos são calculados como:

$$ch_1 = \bar{q} - \frac{1}{2}\beta(p_2 - p_1)$$

$$ch_2 = \bar{q} + \frac{1}{2}\beta(p_2 - p_1)$$

  onde $\bar{q} = \frac{1}{2}(p_1 + p_2)$, $p_2 > p_1$, $p_1$ e $p_2$ são os pais e, $ch_1$ e $ch_2$ são os filhos. Pode-se observar que $\bar{ch} = \bar{q}$.

  A distribuição probabilística de $\beta$ no SBX deveria ser similar a distribuição probabilística de $\beta$ em uma recombinação de codificação binaria. Deb e Agrawal (1995) propõe uma função de distribuição probabilística:

$$f(\beta) = \begin{cases} 0.5(\eta+1)\beta^{\eta}, & \text{se } \beta \leq 1, \\ 0.5(\eta+1)\frac{1}{\beta^{\eta+2}} & \text{caso contrário} \end{cases}$$

  onde valores pequenos para $\eta$ geram descendentes que são mais parecidos com seus pais e, por outro lado, grandes valores geram descendentes muito diferentes dos pais.

  O Fator de Propagação $\bar{\beta}$ pode ser um número aleatório que segue a função de distribuição probabilística proposta. Para que seu valor seja calculado, faz-se:

  - Gera um número aleatório $r$ no intervalo de [0,1];
  - Encontra $\bar{\beta}$ que faz a área sob a curva ser igual a $r$.

  Deste modo, os valores dos filhos são calculados como:

$$ch_1 = \bar{q} - \frac{1}{2}\bar{\beta}(p_2 - p_1)$$

$$ch_2 = \bar{q} + \frac{1}{2}\bar{\beta}(p_2 - p_1)$$

- Outros.

### 3.2.5 Operador de mutação

Este operador é utilizado após a aplicação dos operadores de recombinação. Seu objetivo principal é introduzir diversidade entre os novos indivíduos da população.

- Mutação de Inversão de Bits

  Em uma codificação binária, o operador de mutação efetua uma troca no *bit*. Se o *bit* for 1, ele passa para 0 e vice-versa. Cada *bit* é mudado com taxa $p_m$. Os exemplos a seguir esclarecem esta operação:

$$\begin{aligned} |0|110|0|110 \quad &\rightarrow \quad |1|110|1|110 \\ 10|0|1|1|0|1|1 \quad &\rightarrow \quad 10|1|1|0|0|0|1 \end{aligned}$$

- Mutação Aleatória

  Para codificação binária e real, o operador de mutação escolhe aleatoriamente um elemento no cromossomo e substitui o seu valor por um novo valor obtido aleatoriamente dentro do intervalo admissível pela variável.

$$\begin{aligned} 0110|0|110 \quad &\rightarrow \quad 0110|1|110 \;\; intervalo\ [0,\ 1] \\ 22.5\ |1.34|\ 47.89 \quad &\rightarrow \quad 22.5\ |4.34|\ 47.89 \;\; intervalo\ [0.5,\ 5.5] \end{aligned}$$

- Mutação Polinomial

  A mutação Polinomial (Deb e Goyal, 1996) é um operador que utiliza distribuição de probabilidade polinomial para perturbar uma solução na sua proximidade. Dado um indivíduo $p \in [a, b]$, onde $a$ e $b$ são os limites inferiores e superiores da variável $p$, o indivíduo da mutação polinomial $p'$ é criado usando um número aleatório $r \in [0, 1]$, para uma determinada variável, da seguinte forma:

$$p_i' = \begin{cases} p_i + \bar{\delta}_L(p_i - x_i^{(L)}), & \text{para } r \leq 0.5, \\ p_i + \bar{\delta}_R(x_i^{(R)} - p_i) & \text{para } r > 0.5 \end{cases}$$

  onde $\bar{\delta}_L$ e $\bar{\delta}_L$ podem ser calculados da seguinte forma:

$$\begin{aligned} \bar{\delta}_L &= (2r)^{(1/1+\eta_m)} - 1, for\ r \leq 0.5 \\ \bar{\delta}_R &= 1 - (2(1-r))^{(1/1+\eta_m)}, for\ r > 0.5 \end{aligned}$$

  onde valores pequenos para $\eta_m$ geram indivíduos próximos do original e, por outro lado, grandes valores geram indivíduos distantes do original. É importante notar que nenhum valor fora o intervalo $[a, b]$ é criado pela mutação Polinomial.

- Outros.

## 3.3 Algoritmo L-BFGS-B

O Algoritmo L-BFGS-B (Byrd et al., 1995; Zhu et al., 1997) é uma extensão do algoritmo L-BFGS (Liu e Nocedal, 1989) para o tratamento dos limites inferiores e superiores das variáveis de projeto de um problema de otimização. Ambos os algoritmos foram originados através do algoritmo BFGS (Fletcher, 1987), considerado um método iterativo quase-Newton de busca local.

O Algoritmo 2 apresenta um pseudocódigo do algoritmo BFGS. O valor $\mathbf{H}_k$ representa a aproximação da matriz Hessiana da função $f(\mathbf{x}_k)$ no passo $k$. $\nabla f(\mathbf{x}_k)$ representa o gradiente da função $f(\mathbf{x}_k)$ no passo $k$, onde o mesmo pode ser o valor real ou o valor aproximado. $||\mathbf{v}||$ representa a norma do vetor $\mathbf{v}$. A matriz Hessiana é atualizada a cada iteração do algoritmo BFGS, conforme mostrado na linha 12 do pseudocódigo.

**Algoritmo 2:** Pseudocódigo BFGS.

1 **Entrada:** ponto inicial $\mathbf{x}_0$ e matriz Hessiana inicial $\mathbf{H}_0$;
2 **Saída:** ponto $\mathbf{x}_k$;
3 **para** $k = 0, 1, 2, \ldots$ **faça**
4 $\quad \mathbf{g}_k = -\nabla f(\mathbf{x}_k)$;
5 $\quad$ **se** $||\mathbf{g}_k|| < \varepsilon$ **então**
6 $\quad\quad$ Termina;
7 $\quad$ Obtém a direção $\mathbf{d}_k$ resolvendo o sistema $\mathbf{H}_k\mathbf{d}_k = \mathbf{g}_k$;
8 $\quad$ Encontre $\alpha_k$ que minimiza $||f(\mathbf{x}_k + \alpha_k\mathbf{d}_k)||$;
9 $\quad \mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{d}_k$;
10 $\quad \mathbf{s}_k = \alpha_k\mathbf{d}_k$;
11 $\quad \mathbf{y}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$;
12 $\quad \mathbf{H}_{k+1} = \mathbf{H}_k + \frac{1}{\mathbf{y}_k^\mathbf{T}\mathbf{s}_k}\mathbf{y}_k\mathbf{y}_k^\mathbf{T} + \frac{1}{\mathbf{s}_k^\mathbf{T}\mathbf{H}_k\mathbf{s}_k}\mathbf{H}_k\mathbf{s}_k\mathbf{s}_k^\mathbf{T}\mathbf{H}_k$;

É importante notar que a aproximação da Hessiana fica cada vez melhor conforme as iterações do BFGS ocorrem. Diferente do método de Newton, o algoritmo BFGS não necessita que a matriz Hessiana seja exata. Apesar disso, o algoritmo requer uma grande quantidade de memória para realizar a atualização de todos os elementos da matriz Hessiana. Com o objetivo de reduzir a quantidade de memória utilizada o algoritmo L-BFGS foi desenvolvido.

O Algoritmo 3 apresenta um pseudocódigo do algoritmo L-BFGS. Nesta nova versão do algoritmo, a atualização da matriz Hessiana $\mathbf{H}_k$ não ocorre (apesar disso, ela não necessita ser idêntica à matriz Hessiana inicial $\mathbf{H}_0$ em cada passo de tempo $k$).

Para contornar o problema referente à quantidade de memória, apenas $m$ pares de vetores $(s_i, y_i)$ são armazenados, onde estes pares contêm sempre informação referente à curvatura da função dos últimos $m$ passos, onde $m$ denota o número de iterações anteriores a serem levadas em consideração. Devido a isso, se a matriz Hessiana inicial $\mathbf{H}_0$ é esparsa,

o algoritmo L-BFGS não demandará grandes quantidade de memória para armazenar a matriz Hessiana, nos casos em que $m$ seja adequadamente pequeno.

**Algoritmo 3:** Pseudocódigo L-BFGS.

**Entrada:** ponto inicial $\mathbf{x}_0$ e matriz Hessiana inicial $\mathbf{H}_0$;
**Saída:** ponto $\mathbf{x}_k$;
**para** $k = 0, 1, 2, \ldots$ **faça**
    $\mathbf{g}_k = -\nabla f(\mathbf{x}_k)$;
    **se** $||\mathbf{g}_k|| < \varepsilon$ **então**
        Termina;
    $\mathbf{q}_k = \mathbf{g}_k$;
    **para** $i = k-1, k-2, \ldots, k-m$ **faça**
        $\rho_i = \frac{1}{\mathbf{y}_k^{\mathbf{T}} \mathbf{s}_k}$ ;
        $a_i = \rho_i \mathbf{s}_i^{\mathbf{T}} \mathbf{q}_k$;
        $\mathbf{q}_k = \mathbf{q}_k - a_i \mathbf{y}_i$
    Obtém a direção $\mathbf{d}_k$ resolvendo o sistema $\mathbf{H}_k \mathbf{d}_k = \mathbf{q}_k$;
    **para** $i = k-m, k-m+1, \ldots, k-1$ **faça**
        $b = \rho_i \mathbf{y}_i^{\mathbf{T}} \mathbf{d}_k$;
        $\mathbf{d}_k = \mathbf{d}_k + (a_i - b)\mathbf{s}_i$;
    Encontre $\alpha_k$ que minimiza $||f(\mathbf{x}_k + \alpha_k \mathbf{d}_k)||$;
    $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$;
    $\mathbf{s}_k = \alpha_k \mathbf{d}_k$;
    $\mathbf{y}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$;

Os algoritmos BFGS e L-BFGS foram desenvolvidos para realizar otimização em problemas sem restrições, o que torna os mesmos inapropriados para qualquer problema com restrições. Até mesmo os problemas com apenas restrições de caixa não podem ser otimizados com estes algoritmos. Para tratar problemas com restrições de caixa, o algoritmo L-BFGS-B foi desenvolvido.

O L-BFGS-B é um algoritmo de memória limitada capaz de resolver grandes problemas de otimização não lineares sujeitos a restrições de caixa nas variáveis de projeto. O algoritmo é indicado para problemas onde a obtenção da matriz Hessiana original é inviável e onde as restrições de caixa devem ser satisfeitas.

A diferença entre o algoritmo L-BFGS-B para o L-BFGS é o tratamento das restrições de caixa. Para tratar estas restrições, o L-BFGS-B realiza uma busca unidimensional ao longo da direção $\mathbf{d}_k$, sujeita às restrições de caixa do problema, para calcular o tamanho do passo.

Uma característica importante do algoritmo L-BFGS-B, assim como do L-BFGS, é o fato de o mesmo evitar o armazenamento das aproximações sequenciais da matriz Hessiana, o que permite que ele generalize bem as altas dimensionalidades (Coppola e Stewart, 2014). O algoritmo armazena informações sobre a curvatura da função objetivo

nas últimas $k$ iterações e utiliza esta mesma informação para encontrar uma nova direção de busca.

## 3.4 Tratamento de restrições

Os problemas de otimização com restrições, comumente encontrados em engenharia, podem demandar dificuldade na busca por soluções viáveis. Existem várias técnicas para o tratamento de restrições e, entre elas, uma bastante utilizada é a que faz uso de funções de penalização. As meta-heurísticas bioinspiradas utilizam largamente as técnicas de penalização, pois foram desenvolvidas para resolver problemas de otimização sem restrições.

As técnicas para o tratamento de restrições que fazem uso de funções de penalização transformam o problema com restrições em problemas sem restrições através da introdução da função de penalização em sua função aptidão.

Mezura-Montes e Coello (2011) apresentaram uma revisão bibliográfica extremamente detalhada acerca de várias técnicas para o tratamento de restrições. Uma das formas mais simples de avaliar um problema de otimização com restrições é:

$$F(\mathbf{x}) = f(\mathbf{x}) + penal(\mathbf{x})$$

onde $f(\mathbf{x})$ representa a função aptidão real, $penal(\mathbf{x})$ representa a função de penalização (representação da violação das restrições) e $F(\mathbf{x})$ representa a função aptidão.

Na maioria dos métodos, a parcela $penal(\mathbf{x})$ 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 aplicadas às soluções infactíveis.

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

Basicamente, o que é feito é penalizar um indivíduo que não satisfaz alguma restrição do problema de modo que o indivíduo que violar mais receberá uma penalização maior e o que violar menos receberá uma penalização menor. Estas técnicas podem ser estáticas, dinâmicas ou adaptativas.

Nesta dissertação, será utilizado um método de penalização adaptativa, o qual utiliza informações do processo evolutivo do algoritmo genético para definir automaticamente um parâmetro de penalidade para cada restrição. Este processo não necessita de parâmetros a serem definidos e pode ser aplicado a qualquer problema de otimização com restrição.

### 3.4.1 Método de penalização adaptativa

O Método de Penalização Adaptativa (*Adaptive Penalty Method* - APM) foi desenvolvido por Barbosa e Lemonge (2002) para aplicação em problemas com restrições em um Algoritmo Genético. Este método de penalização não requer nenhum parâmetro definido diretamente pelo usuário e não demanda de conhecimento do problema.

As informações necessárias para a adaptação do método são extraídas diretamente da população, tais como a média da função objetivo e o nível de violação de cada uma das restrições. 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. Se a solução avaliada for factível, o método retornará exatamente o valor da função objetivo, mas, caso contrário, se a solução for infactível, devem-se realizar algumas verificações.

As informações obtidas através da população em cada geração são utilizadas para definir os coeficientes de penalização. Estes coeficientes são empregados para penalizar os indivíduos infactíveis da população corrente de acordo com os seus níveis de violação. A ideia é que os coeficientes de penalização sejam distribuídos de forma que aqueles relacionados a restrições mais difíceis de ser atendidas tenham valores relativamente mais altos.

Para avaliar uma solução infactível, deve-se verificar se a sua função objetivo é maior do que a média da função objetivo da população. Se ela for maior, deve-se indicar que a solução infactível tem um valor igual a sua própria função objetivo acrescido 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.

A função aptidão para um problema de minimização pode ser escrita como:

$$F(\mathbf{x}) = \begin{cases} f(\mathbf{x}), & \text{se } \mathbf{x} \text{ é factível,} \\ \overline{f}(\mathbf{x}) + \sum_{j=1}^{m} k_j v_j(\mathbf{x}) & \text{caso contrário} \end{cases} \tag{3.1}$$

onde

$$\overline{f}(\mathbf{x}) = \begin{cases} f(\mathbf{x}), & \text{se } f(\mathbf{x}) > \langle f(\mathbf{x}) \rangle, \\ \langle f(\mathbf{x}) \rangle & \text{caso contrário} \end{cases} \tag{3.2}$$

a $\langle f(\mathbf{x}) \rangle$ e a média da função objetivo na população atual, $f(\mathbf{x})$ é a função objetivo, $v_j(\mathbf{x})$ é o nível de violação da $j$-ésima restrição e $k_j$ é o $j$-ésimo coeficiente de penalização.

O coeficiente de penalização de cada restrição é definido em cada geração por:

$$k_j = |\langle f(\mathbf{x}) \rangle| \frac{\langle v_j(\mathbf{x}) \rangle}{\sum_{l=1}^{m} [\langle v_l(\mathbf{x}) \rangle]^2} \tag{3.3}$$

onde $\langle v_j(\mathbf{x})\rangle$ $(j = 1, 2, \ldots, m)$ e $\langle v_l(\mathbf{x})\rangle$ são respectivamente as médias das violações às restrições $j$ e $l$ na população atual. Denotando por $N_{POP}$ o tamanho da população, pode-se escrever:

$$k_j = \frac{|\sum_{i=1}^{pop} f(\mathbf{x}^i)|}{\sum_{l=1}^{m}[\sum_{i=1}^{N_{POP}} v_l(\mathbf{x}^i)]^2} \sum_{i=1}^{pop} v_j(\mathbf{x}^i) \tag{3.4}$$

A Figura 7, extraída e adaptada de Barbosa e Lemonge (2008), apresenta um conjunto de soluções factíveis e infactíveis. Os seis pontos numerados representam o conjunto de soluções infactíveis. Entre este conjunto, os quatro primeiros (representados por círculos abertos) possuem seus valores de função objetivo menores do que a média da função objetivo da população e, devido a isso, o valor $\overline{f}(\mathbf{x})$ é alterado para a média da função objetivo $\langle f(\mathbf{x})\rangle$. As outras duas soluções infactíveis possuem seus valores de função objetivo maiores do que a média da função objetivo da população e, assim, $\overline{f}(\mathbf{x}) = f(\mathbf{x})$

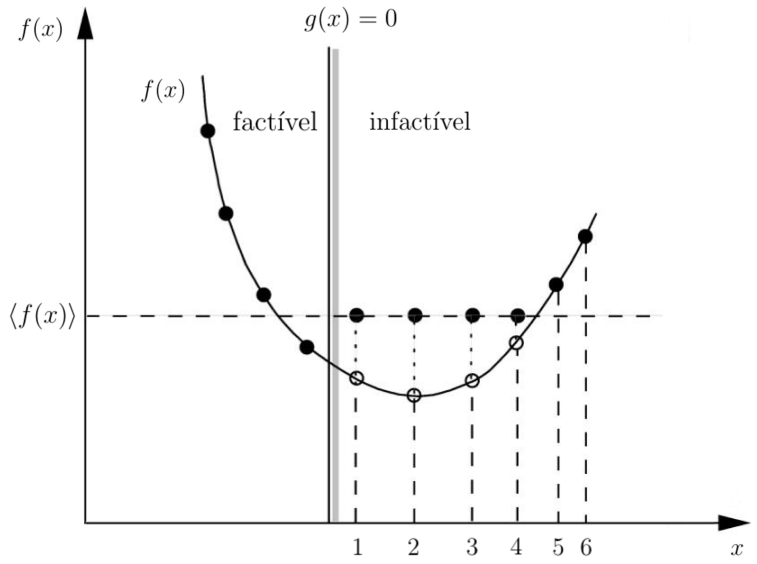



Figura 7 – Descrição da função $\overline{f}$, extraída e adaptada de (Barbosa e Lemonge, 2008).

# 4 METAMODELOS

## 4.1 Introdução

Os problemas de otimização que apresentam um alto custo computacional acabam impossibilitando uma busca rápida das soluções boas ou ótimas. Uma das alternativas encontradas para tratar este problema é a utilização de modelos de substituição ou metamodelos (*Surrogate Models*) (Forrester et al., 2008). Pode-se dizer que os metamodelos são modelos simplificados do modelo original do problema, de modo que o seu papel é substituir ou aproximar funções que requerem muito esforço computacional para encontrar o resultado.

Um modelo nada mais é do que uma idealização simplificada de fenômenos no mundo real, mas que, apesar disso, reproduz de maneira significativa o comportamento observado. Já o metamodelo é mais uma abstração, destacando propriedades do próprio modelo.

De acordo com Fonseca (2009), o termo metamodelo surgiu por volta dos anos 1970 (Blanning, 1974). O objetivo dos primeiros trabalhos era de auxiliar no cálculo das sensibilidades dos modelos de simulação, no qual existia a necessidade de executar o metamodelo várias vezes (Blanning, 1974, 1975; Michel e Permut, 1975; Kleijnen, 1975). Um exemplo de metamodelo sendo aplicado em métodos clássicos de otimização pode ser observado no método BFGS (Fletcher, 1987), em que a inversa da matriz Hessiana é construída através de aproximação, reduzindo o custo computacional no processo de inversão de matriz.

Podem-se encontrar na literatura outros trabalhos relacionados a metamodelo. Alguns dos metamodelos mais populares de acordo com Fonseca (2009) são:

- As Redes Neurais Artificiais são modelos computacionais inspirados na estrutura neural de um ser vivo e adquirem conhecimento através das experiências (Cochocki e Unbehauen, 1993; Haykin, 1994; Hastie, 1998);
- Os Processos Gaussianos ou Kriging se baseiam no princípio que os pontos próximos no espaço tendem a valores similares do que pontos mais distantes, ou seja, os dados se encontram correlacionados no espaço (Cressie, 1990; Quiñonero-Candela e Rasmussen, 2005; Stein, 2012).
- As Máquinas de Vetores Suporte, em sua definição original, buscam encontrar o hiperplano separador entre dados de duas classes (Cortes e Vapnik, 1995; Hearst et al., 1998; Gunn et al., 1998).
- Os Modelos Polinomiais Lineares e Não Lineares são técnicas que permitem construir métodos de regressão lineares e não lineares. A sua principal ideia é a transformação

de uma regressão não linear em uma regressão linear, utilizando um outro espaço, normalmente de dimensões maiores (Lesh, 1959; Blanning, 1974; Saunders et al., 1998).

A primeira aplicação de metamodelo em computação evolucionista foi na década de 1980, quando foi realizada a substituição das avaliações (Grefenstette e Fitzpatrick, 1985). Desde então, a ideia de metamodelo vem sendo utilizado em várias aplicações, tendo como principal objetivo reduzir o custo computacional.

## 4.2 Revisão bibliográfica

Metamodelos vêm sendo utilizados cada vez mais, tornando possível o tratamento e a otimização de problemas que anteriormente não eram possíveis. Diversas abordagens relacionadas a metamodelo vêm sendo utilizadas em diferentes tipos de problemas reais de otimização.

O trabalho desenvolvido por Alexandrov et al. (1998) apresenta uma abordagem para o gerenciamento do uso de metamodelos na otimização baseada na abordagem da região de confiança da programação não linear. De acordo com a proposta, é provável que a abordagem seja convergente para uma solução do problema original de alta fidelidade. Além disso, o método para o gerenciamento das aproximações na otimização em engenharia sugere formas de se decidir quando a fidelidade, assim como o custo das aproximações, podem ser aumentadas ou diminuídas no decorrer das iterações da otimização.

É apresentada por El-Beltagy et al. (1999) uma estrutura geral para o acoplamento de técnicas de metamodelos com algoritmos evolutivos para reduzir o custo computacional de problemas da ciência e da engenharia que são computacionalmente caros. Estes problemas de alto custo representam uma séria restrição para a aplicação bem-sucedida de técnicas de otimização evolutivas e as técnicas de metamodelos apresentam a possibilidade de redução destes custos.

No artigo apresentado por Booker et al. (1999) apresenta e analisa uma ferramenta para gerar uma sequência de aproximações da função objetivo e gerenciar o uso dessas aproximações como substitutos para otimização. O objetivo foi desenvolver algoritmos de otimização que levam a soluções rigorosas do problema para a aplicação em projetos de engenharia, onde a otimização tradicional não é satisfatória. Além disso, Serafini (1999), em sua Tese, desenvolveu uma metodologia para gerenciamento de modelo aproximado, de modo que, no processo de busca, algumas pontos eram avaliados na função objetivo real para adaptar o modelo em uma região de interesse.

Jin et al. (2002) realizaram extensos estudos empíricos para poder investigar as propriedades de convergência de uma estratégia evolutiva que utiliza uma função aproximada para avaliação das possíveis soluções candidatas ao problema de otimização. Neste

estudo, dois problemas de *benchmark* foram avaliados. Foi observado que a convergência do problema será incorreta se o modelo aproximado tiver um valor ótimo falso, ou seja, o ótimo do modelo aproximado não representa o ótimo no problema real. É proposta uma estrutura para o gerenciamento dos modelos aproximados e ela é capaz de garantir a convergência correta dos algoritmos evolutivos, assim como reduzir o custo de computação, tanto quanto possível.

Giannakoglou (2002) estudou um conjunto de métodos relevantes capazes de construir metamodelos eficientes de modo a reduzir o alto custo computacional, assim como tornar otimizações estocásticas eficientes e efetivas. Algoritmos evolutivos estão sendo utilizados na aeronáutica devido ao fato de oferecerem uma ferramenta útil para ajudar a resolver problemas tecnológicos atuais e futuros. Apesar disso, todos os algoritmos de busca baseados em população requerem alto custo computacional devido à quantidade de soluções candidatas que participam do processo evolutivo.

Karakasis et al. (2003) propõem um método que é utilizado para resolver problemas inversos e de otimização na aeronáutica e em turbomáquinas. O projeto e a otimização de formas aerodinâmicas usando um AG, apesar de serem robustos, sofrem com o alto custo computacional devido a chamadas excessivas para as simulações envolvidas na Dinâmica dos Fluidos Computacional no processo de avaliação das soluções candidatas do problema. O objetivo do trabalho é utilizar um metamodelo no contexto de um AG distribuído em ambiente de computação e com isso demonstrar que a combinação de ambos resulta na economia do custo da CPU.

Ong et al. (2003) apresentam um algoritmo evolutivo que tira proveito dos modelos de substituição para resolver problemas de otimização, com restrições gerais, computacionalmente caros em um orçamento computacional limitado. Destaca-se o algoritmo evolutivo acoplado a um modelo de programação quadrática sequencial inspirado na aprendizagem Lamarckiana. Foi empregada uma abordagem de região de confiança, de modo a intercalar o uso de modelos exatos com modelos de substituição durante a busca local.

O trabalho apresentado por Büche et al. (2003, 2005) realiza um procedimento de otimização utilizando modelos empíricos como aproximação de funções com alto custo. O modelo é treinado com o conjunto atual de soluções avaliadas e a busca pela solução ótima é realizada sobre o modelo treinado. Esta solução ótima é então avaliada na função real e as suas informações são inseridas no conjunto de soluções avaliadas. Na próxima iteração, o processo se repete e o modelo aproximado fica mais robusto. Esta proposta de metamodelo avalia apenas o ótimo previsto na função real, caso contrário, ele utiliza o modelo aproximado.

Zhou et al. (2007) apresentam uma nova estrutura de otimização evolucionista assistida por um modelo de substituição para resolver problemas com alto custo computa-

cional. A estrutura proposta utiliza modelos de substituição computacionalmente baratos, que são construídos através de aprendizado *on-line* de modo a realizar a substituição da função objetivo altamente custosa durante o processo de busca evolutiva. Um modelo de substituição global, baseado em um processo Gaussiano, é usado primeiramente para filtrar indivíduos promissores da população do algoritmo evolucionista. Depois disso, esses indivíduos promissores passam por uma busca memética na forma de aprendizado Lamarckiano. Os resultados apresentados nesta dissertação sugerem que o método proposto converge para boas soluções em um orçamento computacional limitado.

Wang e Shan (2007) realizaram uma revisão bibliográfica sobre os metamodelos. No trabalho, foram apresentadas as várias técnicas classificadas por tipo de amostragem, modelagem e ajuste de modelos. Mostraram-se aplicações e pontos que merecem ser aprofundados nas análises.

Fonseca (2009) desenvolveu em sua tese uma metodologia para o uso combinado de AG e metamodelos para otimização mono-objetivo e multiobjetivo de alto custo computacional. No trabalho, metamodelos baseados em similaridade foram incorporados nos AGs com o objetivo de melhorar o seu desempenho.

Tenne e Goh (2010) editaram um livro onde são apresentadas várias técnicas para a construção de modelos de aproximação.

Shi et al. (2012) apresenta uma métrica Bayesiana para complementar o Erro Médio Quadrático de modo a selecionar o melhor metamodelo entre vários candidatos em uma biblioteca que possui incerteza nos dados. Foi proposta uma estratégia para selecionar automaticamente o melhor metamodelo e determinar um tamanho de amostra razoável para a otimização de problemas complexos de grande escala.

Boukouvala e Ierapetritou (2013) realizaram um trabalho com objetivo principal de desenvolver uma estratégia eficiente de projeto e otimização baseado em simulação utilizando metamodelos para o processo de fabricação de comprimidos farmacêuticos. Na abordagem proposta, foi utilizado a técnica Kriging combinada com análise de viabilidade de caixa para resolver problemas de otimização restrita e com ruído em um menor tempo computacional.

Wortmann et al. (2015) propõem a otimização baseada em metamodelo como um método que promove a compreensão de um problema de projeto arquitetônico. O metamodelo interpola um modelo matemático a partir de dados que relacionam parâmetros de projeto com critérios de desempenho. Os projetistas podem interagir diretamente com este modelo de modo a explorar o impacto aproximado da mudança de variáveis de projeto.

Tsirikoglou et al. (2016) investigaram o desempenho de vários esquemas de otimização assistida por metamodelos aplicados a um problema de transferência de calor de uma superfície com nervuras. No estudo, foi utilizado Kriging, Co-Kriging e Máquina de

Vetor Suporte para Regressão como metamodelos em diferentes esquemas de otimização evolutiva.

Tradicionalmente, na adaptação do metamodelo, apenas um único ponto de amostra é escolhido para atualizar o modelo substituto durante cada ciclo de atualização, após o modelo de substituição inicial ser construído. Devido a isso, Liu et al. (2017) propõe uma estratégia alternativa de preenchimento paralelo de amostras para otimização restrita baseada em metamodelos. A estratégia é demonstrada através da otimização de formas aerodinâmicas de asas transônicas.

## 4.3 Modelos de Regressão

Nesta seção, são apresentadas duas técnicas de aprendizado, Máquinas de Vetores Suporte e Redes Neurais com Função de Base Radial, que podem ser utilizadas para obtenção de uma função aproximada através de dados de treinamento.

### 4.3.1 Máquinas de vetores suporte

As Máquinas de Vetores Suporte (*Support Vector Machines* - SVM) são uma técnica fundamentada na Teoria de Aprendizado Estatístico (Cortes e Vapnik, 1995). A sua formulação se baseia no princípio de Minimização do Risco Estrutural (*Structural Risk Minimization* - SRM)(Vapnik e Vapnik, 1998). Essa formulação possui um desempenho de generalização superior ao princípio de Minimização do Risco Empírico (*Empirical Risk Minimization* - ERM)(Gunn et al., 1998), o qual é aplicado em redes neurais convencionais.

Enquanto o ERM realiza a minimização do erro sobre os dados de treinamento informados, espera-se que o SRM faça a minimização de um limite superior para o erro de generalização, o que ocasionaria um desempenho de generalização superior ao ERM.

O SVM é uma técnica que toma como entrada um conjunto de dados de treinamento e constrói um modelo que prediz, para cada nova entrada fornecida, qual das possíveis classes a entrada faz parte. Deste modo, o SVM é considerado um algoritmo de aprendizado de máquina parar gerar classificadores. A versão do SVM para regressão pode ser chamada de Máquinas de Vetores Suporte para Regressão (*Support Vector Regression* - SVR) (Drucker et al., 1997).

Seja a base de dados de treinamento $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\} \subset \mathbb{R}^d \text{ x } \mathbb{R}^s$, onde $s = 1$, $\mathbf{x}_i$ representa um padrão de entrada e $y_i$ o valor de saída a ser predito. De acordo com Smola e Schölkopf (2004), o objetivo do método SVR é encontrar uma função $\hat{f}(\mathbf{x})$ que tenha no máximo um desvio $\varepsilon$ do valor real $y_i$ para todos os dados de treinamento e ao mesmo tempo que seja o mais nivelado possível. Em outras palavras, o método não se importa com o erro, desde que o mesmo seja menor do que $\varepsilon$. A Figura 8 exemplifica a

situação em que a função $\hat{f}(\mathbf{x})$ tem no máximo um desvio $\varepsilon$ do valor real $y_i$ para todos os dados de treinamento (marcações em x).

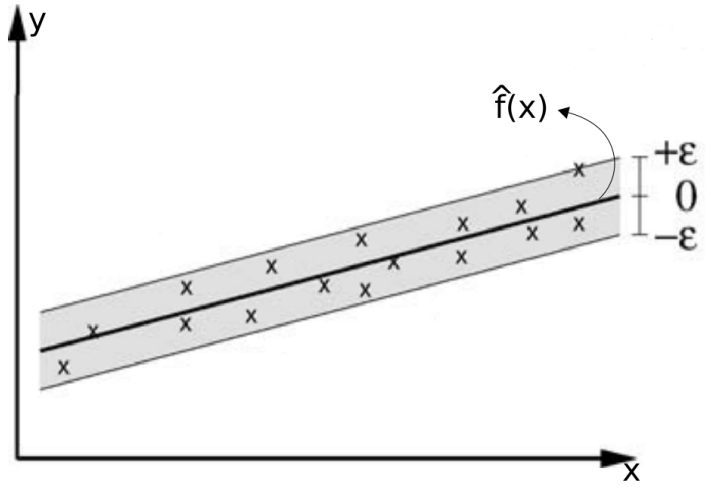



Figura 8 – SVR Linear com desvio máximo $\varepsilon$, modificado de (Smola e Schölkopf, 2004).

Considerando inicialmente o caso das funções lineares, tem-se a seguinte aproximação:

$$\hat{f}(\mathbf{x}) = \mathbf{w}\mathbf{x}^{\mathbf{T}} + b \tag{4.1}$$

em que é realizado o mapeamento linear entre as variáveis de entrada $\mathbf{x} \in \mathbb{R}^d$ na variável de saída $y \in \mathbb{R}$.

O nivelamento no caso da Equação 4.1 significa procurar os menores valores do vetor $\mathbf{w}$ em $\hat{f}(\mathbf{x})$ (Smola e Schölkopf, 2004). Isso pode ser alcançado através da minimização da norma do vetor $\mathbf{w}$. Este problema pode ser modelado como um problema de otimização convexa:

$$\begin{aligned} &\textit{minimizar} \quad \frac{1}{2}||\mathbf{w}||^2 \\ &\text{sujeito a} \\ &\qquad\qquad y_i - \mathbf{w}\mathbf{x}_i^{\mathbf{T}} - b \leq \varepsilon, \ \forall\ i = 1, \ldots, \ell \\ &\qquad\qquad \mathbf{w}\mathbf{x}_i^{\mathbf{T}} + b - y_i \leq \varepsilon, \ \forall\ i = 1, \ldots, \ell \end{aligned} \tag{4.2}$$

O problema de otimização convexa apresentado em 4.2 assume que existe uma função $\hat{f}(\mathbf{x})$ que aproxima os dados de treinamento $\{(\mathbf{x}_1, y_1), \ \ldots, \ (\mathbf{x}_\ell, y_\ell)\}$ com uma determinada precisão $\varepsilon$, ou seja, assume-se que o problema é factível. Entretanto, em alguns casos, tal problema de otimização pode não conter soluções factíveis, ou até mesmo, podem-se permitir alguns erros superiores à margem $\varepsilon$ definida.

De maneira análoga à função de perda de "margem suave" (Bennett e Mangasarian, 1992), podem-se introduzir variáveis de folga $\xi_i$ e $\xi_i^*$ de modo a flexibilizar o modelo da Equação 4.2 a aceitar erros que são superiores ao desvio $\varepsilon$. Assim, a formulação indicada por Vapnik (2013) é dada por:

$$\begin{aligned} &minimizar \quad \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{\ell}(\xi_i + \xi_i^*) \\ &\text{sujeito a} \\ &\qquad y_i - \mathbf{w}\mathbf{x}_i^{\mathbf{T}} - b \leq \varepsilon + \xi_i, \ \forall\ i = 1, \ldots, \ell \\ &\qquad \mathbf{w}\mathbf{x}_i^{\mathbf{T}} + b - y_i \leq \varepsilon + \xi_i^*, \ \forall\ i = 1, \ldots, \ell \\ &\qquad \xi_i, \xi_i^* \geq 0 \end{aligned} \tag{4.3}$$

A constante $C > 0$, chamada de constante de regularização, realiza o equilíbrio entre a complexidade da função $\hat{f}(\mathbf{x})$ e a tolerância de desvios maiores do que $\varepsilon$. Isso corresponde a lidar com uma função de perda $\varepsilon$-insensível descrita por:

$$|\xi|_\varepsilon := \begin{cases} 0 & \text{se } |\xi| \leq \varepsilon \\ |\xi| - \varepsilon & \text{caso contrário} \end{cases} \tag{4.4}$$

Desta maneira, pode-se escrever a Equação 4.3 da seguinte forma:

$$minimizar \quad \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{\ell}|\xi|_\varepsilon \tag{4.5}$$

em que o termo $\frac{1}{2}||\mathbf{w}||^2$ controla a complexidade do modelo e o termo $|\xi|_\varepsilon$ penaliza os valores que se encontram fora do tubo, ou seja, com erros maiores que o desvio $\varepsilon$.

A Figura 9 exemplifica uma função de perda $\varepsilon$-insensível com SVR linear. Apenas os pontos que estão fora da região factível que contribuem para o acréscimo do custo(penalização) na função objetivo através de uma penalização linear.

A formulação da Equação 4.3 é denominada primal. Ocorre que, na maioria dos casos, o problema de otimização pode ser resolvido de maneira mais fácil através de sua formulação dual. Isso acontece quando a dimensionalidade de $\mathbf{w}$ for muito maior do que o número de observações ou quando métodos especializados podem oferecer economia computacional considerável (Lee e Mangasarian, 2001). A formulação do problema de

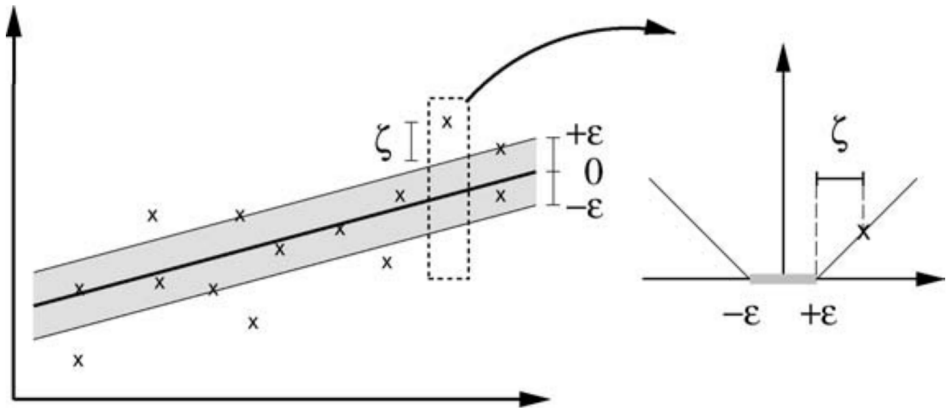



Figura 9 – A função de perda com margem flexível com SVR linear, extraído de (Smola e Schölkopf, 2004).

otimização na sua forma dual pode ser escrita como:

$$
\begin{aligned}
\textit{maximizar} \quad & -\frac{1}{2}[\sum_{i,j=1}^{\ell}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)K(\mathbf{x}_i, \mathbf{x}_j)] \\
& -\varepsilon[\sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*)] + [\sum_{i=1}^{\ell} y_i(\alpha_i - \alpha_i^*)] \\
\text{sujeito a} \quad & \\
& \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*) = 0 \\
& \alpha_i, \alpha_i^* \in [0, C]
\end{aligned}
\tag{4.6}
$$

em que o espaço original é mapeado para um novo espaço, denominado espaço de características, por meio da função $\phi$ e do produto interno $K(\mathbf{x_i}, \mathbf{x_j}) = \phi(\mathbf{x_i})\phi(\mathbf{x_j})^{\mathbf{T}}$, onde $K$ é uma função *kernel*.

*Kernels* fornecem uma maneira de calcular o produto interno em algum espaço de características sem a necessidade de se conhecer o espaço ou a função $\phi$. A Figura 10 ilustra o mapeamento do espaço original no espaço de características em um problema de classificação.

As variáveis $\alpha_i$ e $\alpha_i^*$ da Equação 4.6 representam os multiplicadores de Lagrange que são responsáveis por satisfazer as desigualdades apresentadas no problema primal. O cálculo de $\mathbf{w}$ e da função $\hat{f}(x)$ para o modelo dual pode ser formulado do seguinte modo:

$$
\mathbf{w} = \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*)\mathbf{x_i} \tag{4.7}
$$

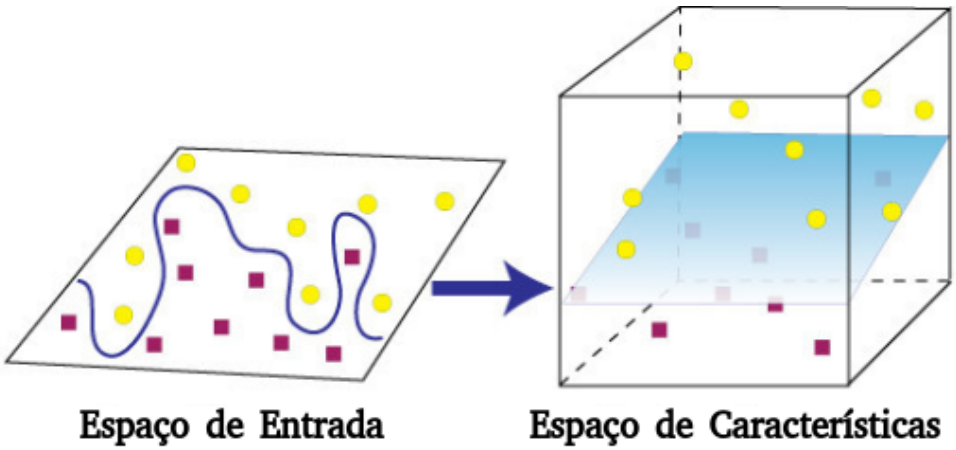



Figura 10 – Mapeamento do espaço original no espaço de características em um problema de classificação, adaptado de (Yu et al., 2010).

$$\hat{f}(x) = \sum_{i=1}^{\ell} (\alpha_i - \alpha_i^*) K(\mathbf{x}_i, \mathbf{x}_j) + b \tag{4.8}$$

Isso é chamado de expansão de vetores suporte, ou seja, $\mathbf{w}$ pode ser determinado por uma combinação linear dos padrões de treinamento $(\mathbf{x}_1, \ldots, \mathbf{x}_\ell)$. De certo modo, a complexidade de uma função construída por vetores suporte é independente da dimensionalidade do espaço de entrada $\mathbb{R}^d$ e depende somente do número de vetores suporte (Smola e Schölkopf, 2004).

Uma maneira de tratar o SVR no caso não linear seria realizando um pré-processamento nos dados de treinamento através de um mapeamento $\phi : \mathbb{R}^d \to \mathfrak{F}$ que leva ao espaço de características $\mathfrak{F}$ (Aizerman et al., 1964; Nilsson, 1965) e, após isso, aplicar o SVR linear nos dados transformados. Deste modo, a linearidade da regressão é obtida no espaço de características e não no espaço original.

Apesar de essa técnica funcionar, o SVR torna-se computacionalmente inviável em problema com uma grande quantidade de dados de treinamento. Para evitar o cálculo da transformação do espaço de entrada no espaço de características, são escolhidas funções *kernel* que podem ser escritas em função dos dados de treinamento. Deste modo, este processo de transformação é feito implicitamente pela função *kernel*, não sendo necessário realizar nenhum tratamento nos dados de treinamento. A Figura 11 apresenta um exemplo da utilização do SVR com diferentes funções *kernel* para dados sintéticos de treinamento, sem realizar tratamento nos mesmos.

Nesta dissertação, foi utilizado a função de base radial (*Radial Base Function* - RBF) como função *kernel*. Este *kernel* é apresentado na Equação 4.9 e o mesmo apresenta

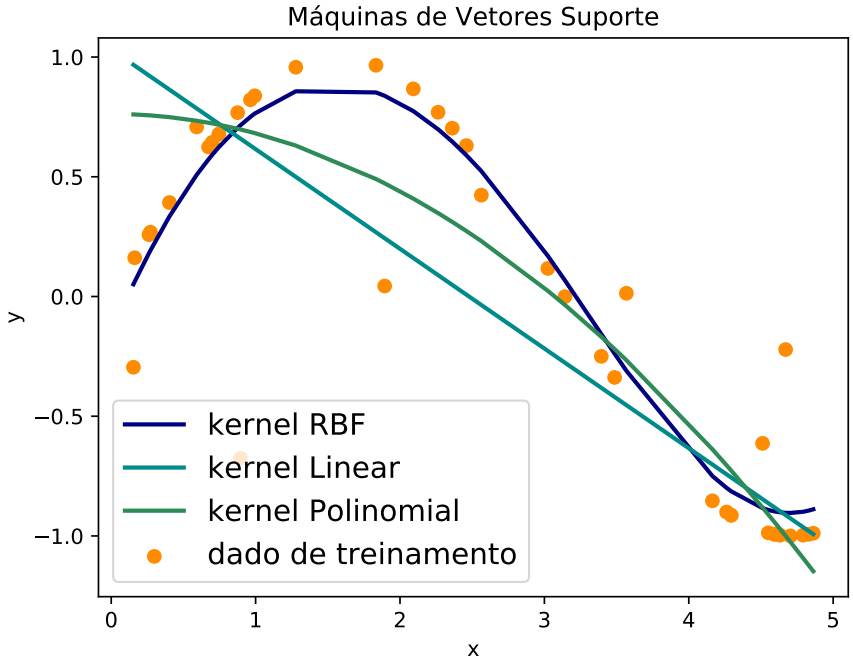



Figura 11 – SVR com diferentes funções *kernel*.

um parâmetro $\gamma$ que deve ser escolhido de forma adequada.

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(\gamma||\mathbf{x}_i - \mathbf{x}_j||^2) \tag{4.9}$$

Um *kernel* expressa uma medida de semelhança entre dois vetores. No caso do *kernel* RBF, esta representação de semelhança é feita através de uma função que decai de acordo com a distância entre os vetores. Deste modo, os vetores próximos são mais representativos na função *kernel* RBF comparados a vetores afastados. Esta função tem uma forma de Curva de Gauss ou Curva do Sino (*Bell Curve*).

O parâmetro $\gamma$ da função *kernel* RBF define o decaimento (a largura da Curva de Sino). Quando o valor de $\gamma$ é pequeno, curvas largas são produzidas. A Figura 12 apresenta uma análise do parâmetro $\gamma$ na função *kernel* RBF.

### 4.3.2 Rede neural com função de base radial

As Redes Neurais Artificiais (*Artificial Neural Network* - ANN) são modelos matemáticos bioinspirados em redes neurais biológicas, capazes de aproximar funções não lineares através de aprendizado. A Figura 13 apresenta um modelo de um neurônio biológico com a sequência de propagação dos sinais pela célula.

O primeiro trabalho a descrever um modelo artificial para um neurônio biológico foi desenvolvido por McCulloch e Pitts (1943). Foi proposto um modelo de neurônio como

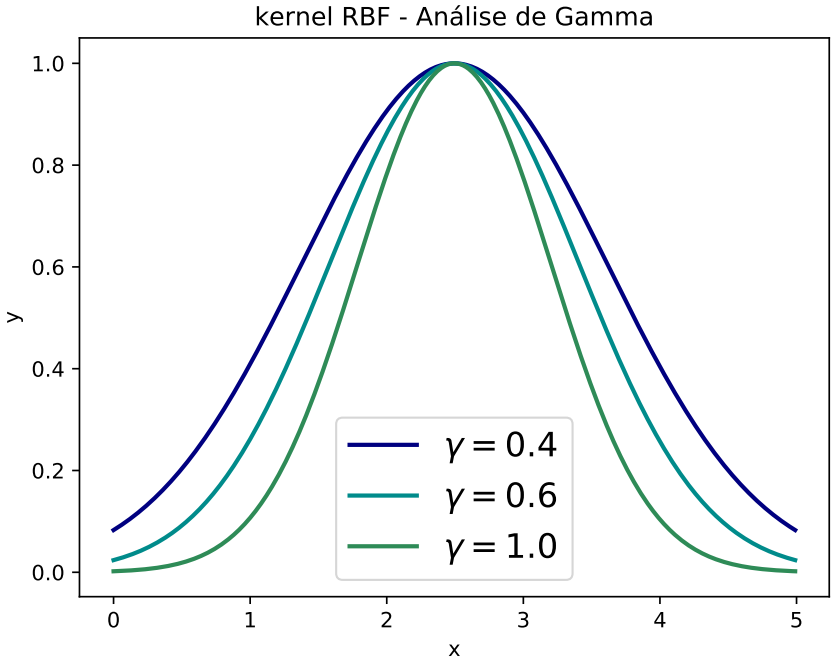



Figura 12 – Análise do parâmetro $\gamma$ na função *kernel* RBF.

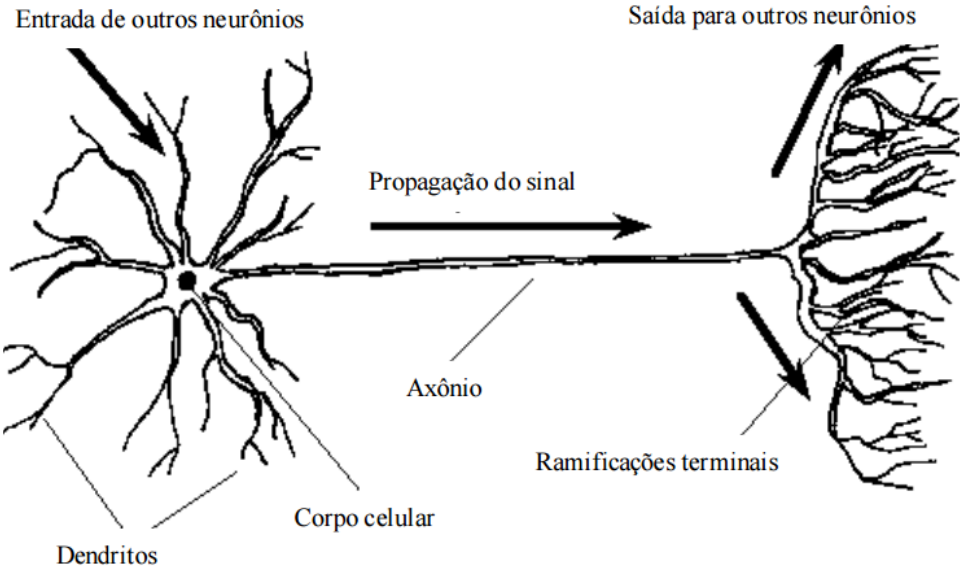



Figura 13 – Célula neural biológica com a sequência de propagação do sinal, extraída de (Kovács, 2002).

uma unidade de processamento binária, em que o modelo era formado por um vetor de entradas e as sinapses eram representadas por pesos numéricos. A soma ponderada das entradas pelos pesos era sujeita à uma função de ativação que determinava se a soma era maior ou menor que um determinado valor. Caso o valor fosse maior, o neurônio era "ativado" (retornando o valor 1), caso contrario, era "desativado" (retornando o valor 0).

O modelo então foi aprimorado em uma nova abordagem para solução do problema de reconhecimento de padrões introduzida em 1958 (Rosenblatt, 1958), denominada *Perceptron*, e sua maior contribuição diz respeito ao desenvolvimento de uma regra de aprendizado. Esta era baseada na minimização do erro quadrático médio de uma aproximação em relação a uma saída desejada. A Figura 14 apresenta a representação do *Perceptron*.

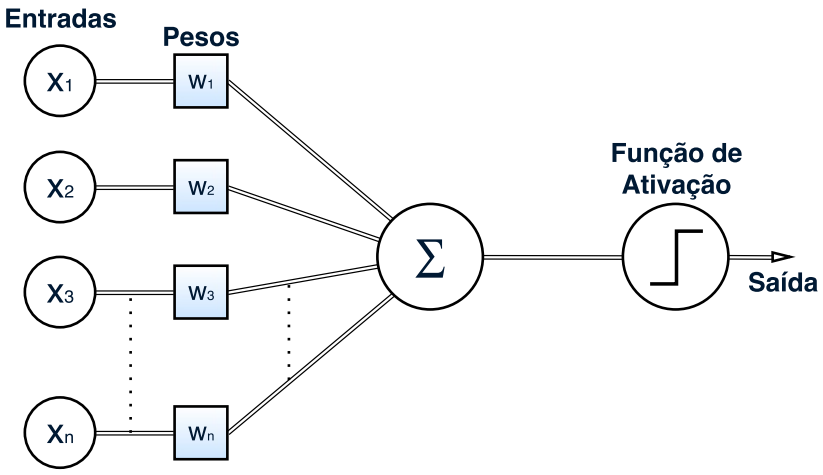



Figura 14 – *Perceptron*.

Minsky e Papert (1969) demonstraram a capacidade limitada do *Perceptron*, por meio de uma análise matemática do mesmo e demostraram que redes com apenas uma camada interna não são capazes de solucionar problemas que não sejam linearmente separáveis. Isso levou a uma grande perda de interesse pelo estudo de redes neurais artificiais.

Rumelhart et al. (1988) desenvolveram um algoritmo de aprendizado de retropropagação (*Backpropagation*), mostrando que é possível treinar eficientemente redes utilizando camadas intermediárias. Isso deu origem a um modelo de redes neurais artificiais mais conhecido, as redes Perceptron Multicamadas (*Multi-Layer Perceptron* - MLP). A Figura 15 apresenta uma arquitetura do MLP com duas camadas intermediárias.

Devido a este trabalho, o interesse pelo estudo de redes neurais artificiais cresceu novamente. Isso se deu devido à capacidade de aproximação universal apresentada pelo MLP. Cybenko (1989) demonstrou ser possível realizar uma aproximação de qualquer mapeamento contínuo definido em uma região compacta do espaço de aproximação, caso seja utilizado um número adequado de neurônios na camada intermediária da rede neural.

Esta capacidade de aproximação universal ocorre porque as camadas intermediárias

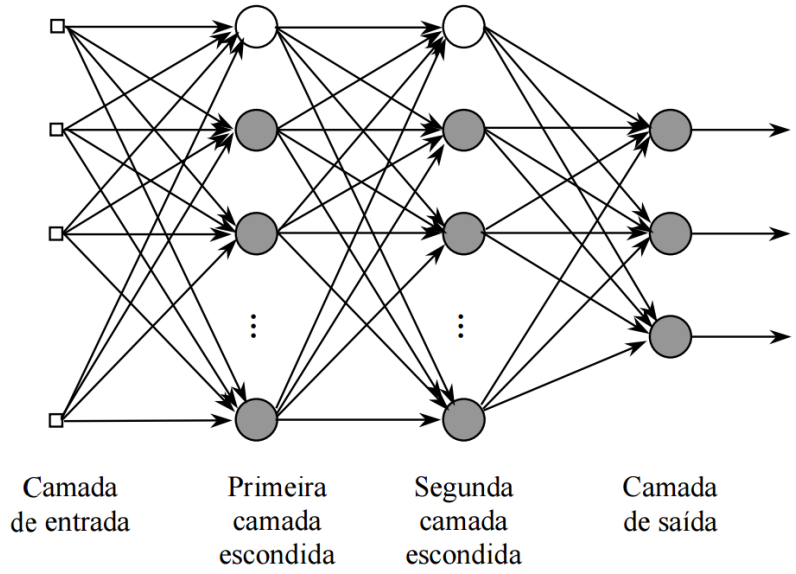



Figura 15 – A arquitetura do MLP com duas camadas intermediárias, extraída de (Kovács, 2002).

funcionam como extratoras de características. Os pesos das camadas intermediárias são uma codificação de características apresentadas nos padrões de entrada, permitindo assim que a rede crie a sua própria representação do problema.

Posteriormente, diferentes arquiteturas de redes neurais artificiais foram desenvolvidas. Nesta dissertação, foi utilizada a Rede Neural de Função de Base Radial (*Radial Basis Function Networks* - RBFN) (Broomhead e Lowe, 1988b). A RBFN é um tipo particular de rede neural artificial com apenas três camadas (Figura 16):

1. a camada de entrada, em que os dados são introduzidos à rede;
2. uma única camada intermediária, que realiza uma transformação não linear do espaço de entrada para um espaço escondido de alta dimensionalidade linearmente separáveis. Esta camada tem como finalidade agrupar os dados de entrada em *clusters*;
3. a camada de saída que produz a resposta da rede neural ao vetor de entrada informado.

A função de ativação de cada neurônio da camada intermediária de uma RBFN é uma função de base radial, ou seja, representa a distância entre o vetor de entrada e centro do neurônio. Já em sua camada de saída, os neurônios artificiais são sempre lineares.

O modelo gerado pela RBFN pode ser formulado do seguinte modo:

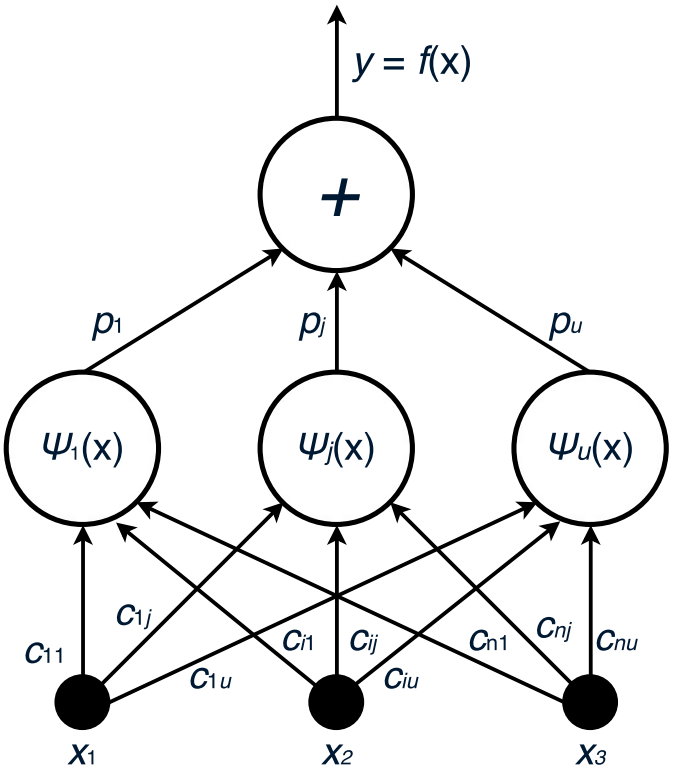



Figura 16 – Arquitetura da Rede Neural de Função de Base Radial (Broomhead e Lowe, 1988a).

$$y = \sum_{i=1}^{u} p_i \psi_i(||\mathbf{x} - \mathbf{c}_i||)$$

onde $\mathbf{c}_i$, $i = 1, ..., u$, são os centros da função de base radial dos neurônios, $p_i$, $i = 1, ..., u$ são os pesos que ponderam cada ativação da camada intermediária para a camada de saída, e $\psi_i(x)$,$i = 1, ..., u$ são as funções de base radial dos neurônios.

As funções de base radial $\psi(x)$, que são usualmente utilizadas, são as funções gaussianas (Schwenker et al., 2001). Apesar disso, outras funções de base radial podem também ser utilizadas como funções de ativação, sem interferir significativamente no desempenho da rede (Haykin, 1994). As funções de base radial são uma classe especial de funções e sua principal característica é que o seu valor diminui/aumenta monotonamente com a distância de um ponto central. A Tabela 4 apresenta alguns exemplos de funções de base radial.

Diferente das redes neurais tradicionais, a RBFN é treinada em dois estágios distintos (Schwenker et al., 2002):

- A definição dos centros da função de base radial dos neurônios, que normalmente são obtidos através de treinamento não supervisionado;

Tabela 4 – Exemplos de Funções de Base Radial.

| Nome | Função |
|---|---|
| Gaussiana | $\psi(r) = e^{-(\varepsilon r)^2}$ |
| Multiquadrática | $\psi(r) = \sqrt{1 + (\varepsilon r)^2}$ |
| Quadrática Inversa | $\psi(r) = \frac{1}{1+(\varepsilon r)^2}$ |
| Multiquadrática Inversa | $\psi(r) = \frac{1}{\sqrt{1+(\varepsilon r)^2}}$ |
| Spline Poli-harmônica | $\psi(r) = r^k,\ k = 1, 3, 5, \ldots$ |
| | $\psi(r) = r^k \ln(r),\ k = 2, 4, 6, \ldots$ |

- O aprendizado dos pesos da camada de saída, que são obtidos através de treinamentos supervisionados.

Para determinar os centros da RBFN, tipicamente, são utilizadas técnicas de clusterização (Moody e Darken, 1989). Apesar disso, outras técnicas podem ser utilizadas, assim como: Árvores de Decisão (Kubat, 1998), Algoritmos Genéticos (Billings e Zheng, 1995) ou os centros podem ser atribuídos como sendo um subconjunto aleatório do conjunto de treinamento. Após a definição dos centros, o aprendizado dos pesos é reduzido a um problema de otimização linear, que pode ser resolvido usando o método de mínimos quadrados ou um método de gradiente descendente (Wu et al., 2012). Nesta dissertação, os centros foram atribuídos como sendo um subconjunto aleatório do conjunto de treinamento e os pesos foram obtidos através do método de mínimos quadrados.

# 5 ALGORITMO DE BUSCA DIRECIONADO

O estudo desenvolvido nesta dissertação envolve uma nova abordagem para a solução de problemas de otimização através de um levantamento de informações teóricas básicas na literatura que serão suficientes para o desenvolvimento, implementação e solução dos problemas propostos.

Os AGs, na sua forma original, apresentam uma grande quantidade de vantagens, mas apresentam a desvantagem de necessitar de um grande número de avaliações da função objetivo do problema em casos em que a função objetivo é cara de se calcular, deste modo o tempo para obtenção da melhor solução cresce significativamente. Apesar de possuir esta grande quantidade de avaliações, os AGs utilizam as informações da aptidão apenas para o desenvolvimento, a cada geração, do processo evolutivo.

No âmbito da Mineração de Dados (*Data Mining* - DM) e da Descoberta de Conhecimento (*Knowledge Discovery in Databases* - KDD), os conjuntos de dados formados pelas avaliações podem conter informações desconhecidas tanto pelo especialista do problema quanto pelo próprio algoritmo genético.

Considerando estas informações, o enfoque principal desta dissertação é o desenvolvimento de um método híbrido de otimização que utiliza o processo evolutivo do algoritmo genético e as informações extraídas dos conjuntos de dados formados pelas avaliações da função objetivo para agilizar o processo de busca. Sem perda de generalidade, este método é composto de uma técnica de aprendizado de máquina combinada com um algoritmo de otimização com o intuito de direcionar o processo de busca de um algoritmo genético.

A Figura 17 apresenta, de modo geral, um fluxograma do método híbrido proposto. O algoritmo de otimização local refinará os indivíduos correntes encontrados pelo algoritmo genérico a cada geração, utilizando o modelo de substituição oriundo do aprendizado de máquina.

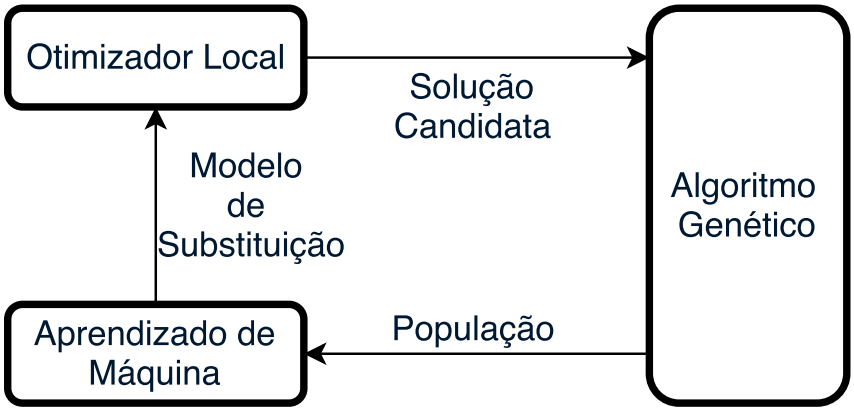



Figura 17 – Fluxograma do Método Híbrido.

Diferentemente dos metamodelos apresentados na revisão bibliográfica do Capítulo 4, este método não realiza a substituição do cálculo da aptidão usado no AG. O modelo aproximado é utilizado na busca local, com o intuito de direcionar a busca do AG através da inserção de possíveis soluções no processo geracional.

## 5.1 Desenvolvimento do metamodelo

O desenvolvimento do metamodelo proposto necessita que quatro etapas fundamentais sejam realizadas no AG a cada geração, sendo elas:

- Coleta e Normalização dos Dados de Treinamento;
- Criação de um Modelo de Substituição;
- Processo de Busca Local;
- Avaliação e Inserção de Possível Solução.

Estas etapas podem ser observadas no Algoritmo 4, nas linhas 9, 10, 11, 12 e 14. Percebe-se que o metamodelo proposto não altera os operadores de seleção, recombinação e mutação do algoritmo genético, mantendo seu processo evolutivo como proposto originalmente.

**Algoritmo 4:** Pseudocódigo de um Algoritmo Genético Direcionado.

```
1  g ← 0;
2  Gera uma população inicial P_g com N_POP indivíduos;
3  Avalia cada indivíduo da população P_g;
4  enquanto g < N_G faça
5      Seleciona N_POP indivíduos em P_g, formando a população G_g;
6      Aplica operador de recombinação (probabilidade p_c) em G_g;
7      Aplica operador de mutação(taxa p_m) em G_g;
8      Avalia cada indivíduo da população G_g;
9      Cria X_g e y_g através da população P_g;
10     Cria uma função f̂(x) com Aprendizado de Máquina sob os dados X_g e y_g;
11     Realiza uma otimização local em f̂(x), obtendo a solução x̂*_g;
12     Realiza avaliação real da solução x̂*_g;
13     Seleciona as k melhores soluções de P_g formando K_g;
14     Combina K_g, x̂*_g e G_g, formando uma população mista Q_g;
15     Ordena a população Q_g;
16     Seleciona os N_POP melhores indivíduos de Q_g e copia para P_{g+1};
17     g ← g + 1;
```

### 5.1.1 Coleta e normalização dos dados de treinamento

Seja $P_g$ uma população de tamanho $N_{POP}$ da geração $g$. No processo de otimização do AG, cada indivíduo da população $P_g$ deve ser avaliado e o valor obtido na função aptidão deve ser armazenado para ser utilizado posteriormente no processo de seleção e elitismo.

Cada cromossomo de um indivíduo do AG pode ser visto como um dado de entrada em uma técnica de aprendizado de máquina e a sua avaliação (função objetivo e valor de penalização) pode ser vista como um dado de saída. Deste modo, pode-se criar um conjunto de dados $X_g$ e $y_g$ que representam todos os cromossomos e todas as avaliações da população $P_g$ da geração $g$, respectivamente.

O conjunto $(X_g, y_g)$ representa os dados de treinamento na geração $g$ da técnica de aprendizado de máquina. Apesar de não ser um passo necessário para o treinamento da técnica de aprendizado de máquina, é extremamente recomendada a utilização de técnicas de normalização nos dados de entrada $X_g$ visto que a normalização geralmente conduz a melhores resultados.

O propósito da utilização de normalização nos dados é de minimizar os problemas provenientes do uso de unidades e dispersões distintas entre as variáveis. Nesta dissertação, foi utilizada a normalização linear no intervalo $[0, 1]$ e ela pode ser escrita como:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Como as variáveis de projeto no processo de busca do AG precisam ter um limite inferior e um limite superior, os seus valores podem ser utilizados no processo de normalização, em que $X_{min}$ assume o valor do limite inferior e $X_{max}$ assume o valor do limite superior.

### 5.1.2 Criação de um modelo de substituição

Através do conjunto $(X_g, y_g)$, é possível realizar um treinamento supervisionado em uma técnica de aprendizado de máquina, gerando um modelo de substituição. O objetivo da técnica é criar um modelo que seja capaz de aprender uma regra geral que realiza uma transformação nos dados de entrada para os dados de saída. Em outras palavras, pretende-se definir uma função $\hat{f}(x)$, tal que:

$$\hat{f}(X_g^i) \simeq y_g^i, \ \forall \ i = 1, \ \ldots, \ N_{POP}$$

Os parâmetros dos modelos de substituição que não são aprendidos diretamente pelo treinamento são definidos através de uma busca exaustiva (*Grid Search*) (Pedregosa

et al., 2011) em um espaço de busca previamente definido dos parâmetros para encontrar o conjunto que leva ao melhor desempenho.

No *Grid Search*, o conjunto de dados $(X_g, y_g)$ é dividido randomicamente cinco vezes em conjuntos de treinamento e teste, com 60% e 40%, respectivamente. O Erro Absoluto Médio (*Mean Absolute Error* - MAE) é calculado para as cinco divisões e o valor da média é então obtido. O conjunto de parâmetros que alcançar o menor valor de média é considerado o conjunto que leva ao melhor desempenho.

Este processo de busca do conjunto de parâmetros pode introduzir no processo de busca um custo computacional indesejável. Devido a isso, duas abordagens foram criadas de modo a minimizar este custo computacional e maximizar a qualidade da aproximação, sendo elas:

- **Busca de Parâmetros na Primeira Geração:** a busca pelo conjunto de parâmetros pelo *Grid Search* ocorre apenas na primeira geração. O conjunto encontrado na primeira geração é então utilizado no treinamento das gerações seguintes.
- **Busca de Parâmetros no Processo Geracional:** a busca pelo conjunto de parâmetros pelo *Grid Search* ocorre apenas na primeira geração e nas gerações em que se observa a melhoria na solução do problema. O conjunto encontrado nestas gerações é então utilizado no treinamento das gerações seguintes até ocorrer uma nova busca de parâmetros ou terminar processo de busca do AG.

### 5.1.3 Processo de busca local

Ao se realizar o processo de busca local na função aproximada $\hat{f}(x)$, deve-se tomar cuidado com o espaço de busca das variáveis de projeto. Como a solução encontrada no processo de busca local deve ser inserida no processo de busca principal, deve-se garantir que a mesma não viole as restrições de limite inferior e superior das variáveis de projeto.

Ao se criar a função aproximada $\hat{f}(x)$ através do conjunto de treinamento $(X_g, y_g)$, obtêm-se boas aproximações próximas do conjunto. Já as possíveis soluções que estão longe da região formada pelo conjunto de treinamento acabam apresentando valores pouco representativos. Devido a isso, como o AG vai convergindo a população para uma sub-região do espaço de busca ao passar das gerações, a busca local limitada por este espaço pode proporcionar soluções que não são representativas.

Para definir um espaço de busca representativo para a busca local, utiliza-se o conjunto $(X_g, y_g)$ para extração dos seus valores mínimos e máximos das variáveis de projeto. Com isso, o espaço de busca do otimizador local é formado pelos mínimos e máximos do conjunto de treinamento normalizado que gerou a função aproximada $\hat{f}(x)$.

Ao se utilizar o algoritmo L-BFGS-B como otimizador local, pode-se definir o espaço de busca das variáveis de projeto de modo a encontrar a solução ótima $\hat{x}_g^*$ dentro de uma região representativa da função aproximada $\hat{f}(x)$. Além disso, o algoritmo L-BFGS-B necessita de um ponto de partida para inicializar o seu processo de busca, onde o melhor indivíduo da população corrente é utilizado para isso (indivíduo dentro da região representativa).

### 5.1.4 Avaliação e inserção de possível solução

A solução $\hat{x}_g^*$ encontrada na busca local deve ser transformada em um indivíduo do processo de busca do AG. Consequentemente, o mesmo deve ser avaliado para obtenção do seu valor de aptidão. Neste processo, é importante notar que nem sempre a solução encontrada na busca local vai refletir em uma solução boa/ótima na busca do AG, ou seja, as soluções presentes no AG podem ser melhores do que a solução $\hat{x}_g^*$.

Após todo processo de seleção, recombinação, mutação e avaliação do AG, é criada uma subpopulação $G_g$. Esta subpopulação $G_g$ é então combinada com as $k$ melhores soluções de $P_g$ (elitismo) e com o indivíduo gerado pela busca local, formando uma população mista $Q_g$.

Como todos os indivíduos da população mista $Q_g$ estão avaliados, pode-se realizar uma ordenação dos mesmos de modo a identificar os piores indivíduos. Se o tamanho da população mista $Q_g$ for maior do que $N_{POP}$, realiza-se uma seleção dos $N_{POP}$ melhores indivíduos para constituir a população $P_{g+1}$.

Neste processo, a solução ótima $\hat{x}_g^*$ da busca local pode ser descartada e não constituir a população seguinte. Isso ocorreria se a solução não fizesse parte dos $N_{POP}$ melhores indivíduos da população mista $Q_g$.

### 5.1.5 Observações de testes preliminares

Durante todo o desenvolvimento do metamodelo, informações relevantes foram observadas e utilizadas no processo de construção da estrutura final do algoritmo. Estas informações são:

- **Coleta dos dados de treinamento:** A criação de um conjunto de dados a cada geração agrega maior contribuição no processo de busca do que um conjunto que contem todas soluções geradas pelo AG até a geração corrente.
  - O conjunto de dados criado a cada geração leva em consideração a busca já realizada pelo AG, de modo que o otimizador local apenas direcionaria a busca na sub-região que o AG esta convergindo.

  - O conjunto que contem todas soluções geradas pelo AG leva em consideração o comportamento do problema, de modo que o otimizador local otimizaria em toda região já percorrida pelo AG.

- **Conjunto ótimo de parâmetros:** A busca do conjunto ótimo de parâmetros em cada geração do AG pode introduzir um alto custo computacional no processo de busca, mas garante que o conjunto de parâmetros encontrado é o que possui melhor desempenho na métrica utilizada.

- **Processo de busca local:** O processo de busca local na função aproximada $\hat{f}(x)$ utilizando o espaço de busca original das variáveis de projeto pode gerar soluções $\hat{x}_g^*$ que não condizem com o problema real. A Figura 18 apresenta um exemplo da busca local na função aproximada $\hat{f}(x)$ utilizando o espaço de busca original.

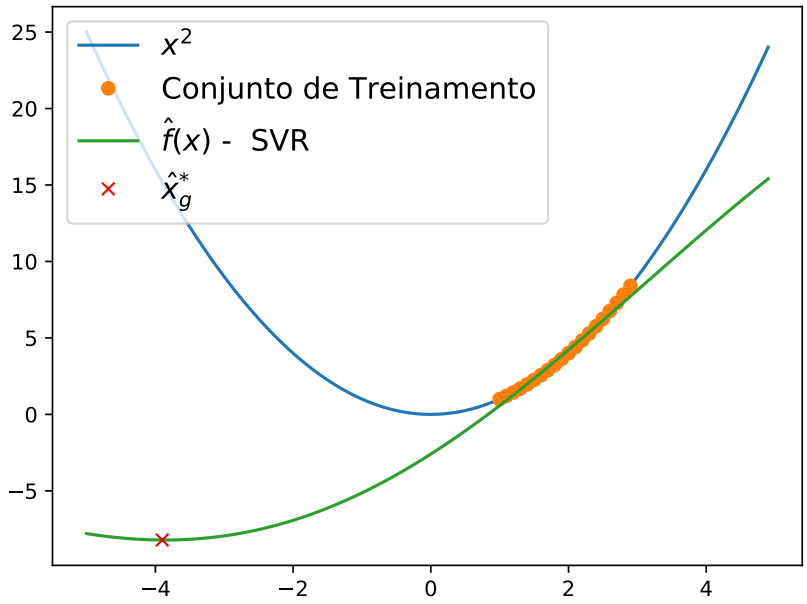



Figura 18 – Exemplo de busca local utilizando o espaço de busca original.

- **Inserção da possível solução sem avaliação:** O tratamento de restrições através do APM, a função aproximada $\hat{f}(x)$ pode tentar representar regiões infactíveis onde o correto seria regiões factíveis e reciprocamente o inverso. Devido a isso, é necessário que a solução $\hat{x}_g^*$ do otimizador local seja reavaliada no problema original para que o processo de busca principal não seja enviezado.

# 6 EXPERIMENTOS NUMÉRICOS

Para a validação do método híbrido de otimização proposto nesta dissertação, foram realizados seis experimentos numéricos envolvendo problemas encontrados na engenharia. Inicialmente foram tratados cinco problemas tradicionais da engenharia estrutural, que são a minimização do peso das treliças de 10, 25, 60, 72 e 942 barras (Lemonge e Barbosa, 2004; Krempser et al., 2017) para os casos discreto e contínuo. O último problema tratado refere-se à otimização em Dinâmica dos Fluídos Computacional (CFD - *Computational Fluid Dynamics*): a otimização de um trocador de calor (Thévenin e Janiga, 2008).

Todos os experimentos utilizaram um algoritmo genético com codificação real, com os seguintes operadores: recombinação SBX e mutação Polinomial. O método de seleção utilizado foi a Seleção por Ranking. A probabilidade de Recombinação e a taxa de Mutação é de 80% e 100%, respectivamente, e a probabilidade de recombinação utilizada pelo SBX e a taxa de mutação utilizada pela mutação Polinomial é de 100% e 5%, respectivamente. Para o parâmetro $\eta$ da recombinação SBX, foi atribuído o valor 2, pois, de acordo com a referência (Deb e Agrawal, 1995), este valor é o mais indicado. Para o parâmetro $\eta_m$ da mutação Polinomial, foi atribuído o valor 100, pois, conforme a referência (Deb e Goyal, 1996), também é o mais indicado.

Para os problemas envolvendo a minimização de pesos de treliças, uma população foi formada por 100 indivíduos e o processo evolutivo iterou por 150 gerações, sendo realizadas 30 execuções independentes. Já para o problema do trocador de calor, a população foi formada por 40 indivíduos e o processo evolutivo iterou por 20 gerações, sendo realizada uma execução independente devido ao alto custo computacional do problema. Além disso, foi adotado o elitismo na população, selecionando o melhor indivíduo para a geração seguinte em todos os problemas, ou seja, $k = 1$ no Algoritmo 4.

Conforme descrito no Capítulo 4, os modelos de regressão possuem um conjunto de parâmetros que não são aprendidos diretamente pelo treinamento. A Tabela 5 mostra o conjunto de parâmetros usados na busca exaustiva para todos os experimentos numéricos.

Tabela 5 – Parâmetros dos modelos usados na busca exaustiva.

| Modelos | Parâmetros | Valores Possíveis |
|---|---|---|
| SVR | Parâmetro de Regularização (C) | $10^2, 10^3, 10^4$ |
| | Parâmetro de Regressão ($\varepsilon$) | $10^{-6}, 10^{-5}, 10^{-4}$ |
| | Kernel | RBF |
| | Kernel RBF - Parâmetro $\gamma$ | $1, 10, 100$ |
| RBFN | Neurônios Escondidos | $30, 50, 80$ |
| | Função de Ativação | Gaussiana, Multiquadrática, Multiquadrática Inversa |
| | Largura RBF | $1.0, 1.5, 2.0$ |

Os resultados dos problemas envolvendo a minimização de pesos de treliças apre-

sentam: a melhor solução encontrada, a pior solução encontrada, a média, a mediana e o desvio-padrão (DP) de todas as soluções, o tempo médio ($\bar{\Delta T}$) de execução e um diagrama de caixa. Para o problema do trocador de calor, são apresentados a solução encontrada e o tempo de execução.

Foram testados cinco algoritmos diferentes, sendo quatro deles referentes à proposta desta dissertação. Os algoritmos são:

- **AG:** Algoritmo Genético comum;
- **AG-SVR$^P$:** Algoritmo Genético com SVR e Busca de Parâmetros na Primeira Geração;
- **AG-SVR$^G$:** Algoritmo Genético com SVR e Busca de Parâmetros no Processo Geracional;
- **AG-RBFN$^P$:** Algoritmo Genético com RBFN e Busca de Parâmetros na Primeira Geração;
- **AG-RBFN$^G$:** Algoritmo Genético com RBFN e Busca de Parâmetros no Processo Geracional.

Um notebook com processador Intel® Core™ i7-4500U com 4 *threads*, memória RAM de 8GB, 1 TB de disco rígido e sistema operacional Ubuntu 16.04 foi utilizado para realizar os experimentos. Todos os experimentos utilizaram os 4 *threads* para execução em paralelo.

## 6.1 Formulação do problema de otimização dimensional

Os problemas de otimização estrutural analisados nesta dissertação consistem em encontrar o conjunto ótimo de áreas $A = \{A_1, \ldots, A_n\}$, tal que o mesmo minimiza o peso $W(A)$ da estrutura, que pode ser escrito como:

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

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

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

onde $\rho$ é a densidade do material, $L_i$ é o comprimento do $i$-ésimo membro da treliça, $u_{max}$ é o deslocamento admissível, $\sigma_{max}$ é a tensão admissível, $m$ é o número de graus de liberdade da treliça e $n$ é o número de barras.

Ao se definir o problema como caso discreto ou contínuo, estipula-se o domínio das variáveis de projeto no processo de busca. Para o caso discreto, cada variável de projeto representa um índice em uma tabela, e esta tabela apresenta os possíveis valores discretos de áreas a serem escolhidas. Para o caso contínuo, cada variável de projeto representa exatamente o valor da área a ser utilizado.

## 6.2 Treliça de 10 barras

O problema de otimização dimensional da treliça de 10 barras (Figura 19) consiste em encontrar o conjunto de áreas $A = \{A_1, \ldots, A_{10}\}$ que minimiza o peso $W(A)$ da estrutura.

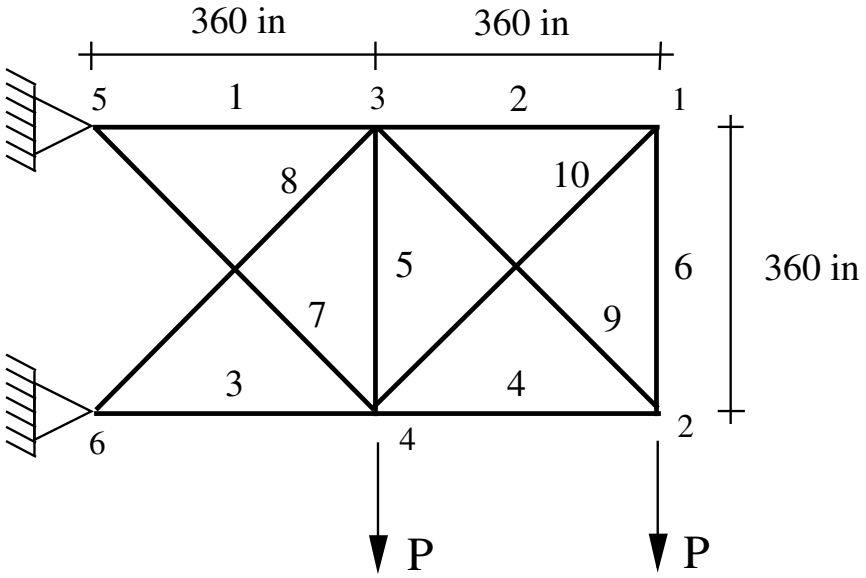



Figura 19 – Treliça de 10 barras.

A tensão admissível em cada barra é de ±25.0 ksi e o deslocamento admissível em cada nó é de 2.0 in nas direções x e y. O material possui massa específica ($\rho$) igual a 0.1 lb/in$^3$ e módulo de elasticidade (E) igual a $10.0^4$ ksi. O problema possui cargas concentradas verticais para baixo com valor de P = 100 kips aplicadas nos nós 2 e 4.

### 6.2.1 Caso discreto

Para o caso discreto, as áreas das secções transversais são escolhidas dentro do conjunto $S = \{1.62, 1.80, 1.99, 2.13, 2.38, 2.62, 2.63, 2.88, 2.93, 3.09, 3.13, 3.38, 3.47, 3.55, 3.63, 3.84, 3.87, 3.88, 4.18, 4.22, 4.49, 4.59, 4.80, 4.97, 5.12, 5.74, 7.22, 7.97, 11.50, 13.50, 13.90, 14.20, 15.50, 16.00, 16.90, 18.80, 19.90, 22.00, 22.90, 26.50, 30.00, 33.50\}$, com unidade in$^2$.

A Tabela 6 e a Figura 20 apresentam os resultados obtidos para a Treliça de 10 Barras no caso discreto, em que os melhores resultados estão em negrito (o mesmo ocorre para os outros experimentos). Observa-se que tanto a proposta AG-SVR$^P$ quanto o AG obtiveram o melhor resultado para o problema. Além disso, entre as piores soluções encontradas, o AG obteve o melhor valor, e a proposta AG-SVR$^P$ obteve o melhor valor entre os valores de média e mediana. Apesar disso, observa-se que a proposta AG-RBFN$^P$ obteve o menor valor para o DP.

Uma Análise de Variância (ANOVA) (St et al., 1989) foi realizada para comparar a distribuição dos 5 algoritmos em amostras independentes. O ANOVA testa a hipótese nula, verificando se as médias dos algoritmos são iguais. Para isso, foi utilizado $\alpha = 0.1$, $\alpha = 0.05$ e $\alpha = 0.01$ para verificação dos resultados com nível de confiança de 90%, 95% e 99%, respectivamente.

Utilizando as soluções encontradas pelas 30 execuções independentes de cada algoritmo analisado, obtém-se no ANOVA o valor p-valor = 0.19194. Deste modo, como p-valor $> 0.1$, p-valor $> 0.05$ e p-valor $> 0.01$, não há evidências de diferenças significativas entre os algoritmos, ao nível $\alpha$ de significância utilizados.

Como o AG não possui nenhuma técnica de aprendizado no seu processo de busca e o tamanho da população e de gerações são iguais para todos os algoritmos, o valor de $\bar{\Delta T}$ sempre vai ser maior para a proposta do que para o AG, e este comportamento se repete para todos os experimentos. O mesmo comportamento pode ser observado entre as propostas em relação ao tipo de busca de parâmetros utilizado.

Tabela 6 – Resultados da Treliça de 10 Barras – Caso Discreto.

| | Melhor | Pior | Média | Mediana | DP | $\bar{\Delta T}$ |
|---|---|---|---|---|---|---|
| AG | **5490.7378** | **5686.4737** | 5559.3338 | 5551.0369 | 41.3317 | 33.65s |
| AG-SVR$^P$ | **5490.7378** | 5712.6459 | **5559.1422** | **5547.3589** | 53.7096 | 38.15s |
| AG-SVR$^G$ | 5491.7173 | 5708.3029 | 5582.5005 | 5591.1278 | 51.4201 | 53.09s |
| AG-RBFN$^P$ | 5491.7173 | 5695.3330 | 5562.3228 | 5550.6099 | **41.2807** | 67.48s |
| AG-RBFN$^G$ | 5507.7584 | 5765.8890 | 5577.4406 | 5574.6053 | 49.3059 | 107.00s |

### 6.2.2 Caso contínuo

Para o caso contínuo, as áreas das secções transversais são escolhidas dentro do conjunto $S = \{x \in \mathbb{R} \mid 0.1 \leq x \leq 40.0\}$, com unidade in$^2$.

A Tabela 7 e a Figura 21 apresentam os resultados obtidos para a Treliça de 10 Barras no caso contínuo. Observa-se que a proposta, em todas as suas variantes, obteve melhor resultado em relação à melhor solução e à mediana, comparando diretamente com o AG. O melhor valor foi obtido pelo AG-SVR$^P$. Além disso, percebe-se que, apesar de a proposta AG-SVR$^G$ não ter encontrado a melhor solução ou a melhor mediana, ela foi a que obteve os melhores resultados em todas as outras métricas.

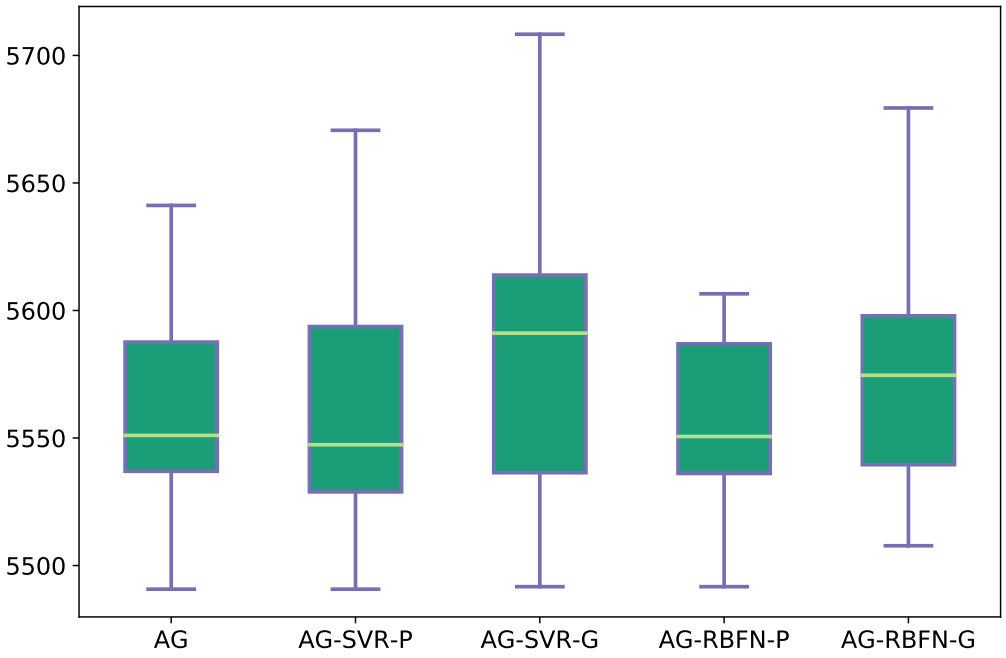



Figura 20 – Diagrama de caixa da Treliça de 10 Barras - Caso Discreto.

Utilizando as soluções encontradas pelas 30 execuções independentes de cada algoritmo analisado, obtém-se no ANOVA o valor p-valor = 0.68591. Deste modo, como p-valor $> 0.1$, p-valor $> 0.05$ e p-valor $> 0.01$, não há evidências de diferenças significativas entre os algoritmos, ao nível $\alpha$ de significância utilizados.

Tabela 7 – Resultados da Treliça de 10 Barras – Caso Contínuo

| | Melhor | Pior | Média | Mediana | DP | $\bar{\Delta T}$ |
|---|---|---|---|---|---|---|
| AG | 5070.6875 | 5544.8164 | 5122.2837 | 5090.4193 | 99.8632 | 32.72s |
| AG-SVR$^P$ | **5065.7067** | 5712.4408 | 5120.1539 | **5081.8233** | 131.4307 | 44.35s |
| AG-SVR$^G$ | 5067.5394 | **5342.6346** | **5108.2839** | 5083.6185 | **70.8436** | 69.37s |
| AG-RBFN$^P$ | 5066.8791 | 5504.4216 | 5134.9053 | 5085.5676 | 118.0962 | 50.99s |
| AG-RBFN$^G$ | 5067.2723 | 6003.0807 | 5156.9038 | 5086.6534 | 202.0293 | 111.70s |

## 6.3 Treliça de 25 barras

O problema de otimização dimensional da treliça de 25 barras (Figura 22) consiste em encontrar o conjunto de áreas $A = \{A_1, \ldots, A_8\}$ que minimiza o peso $W(A)$ da estrutura. A fim de manter a simetria do projeto, cada área representa um grupo de barras conforme mostrado na Tabela 8.

A tensão admissível em cada barra é de $\pm 40.0$ ksi e o deslocamento admissível nos nós 1 e 2 é de 0.35 in nas direções x e y. O material possui massa específica ($\rho$) igual a 0.1 lb/in$^3$ e módulo de elasticidade (E) igual a $10.0^4$ ksi. Os carregamentos que são aplicados sobre a Treliça de 25 barras são apresentados na Tabela 9.

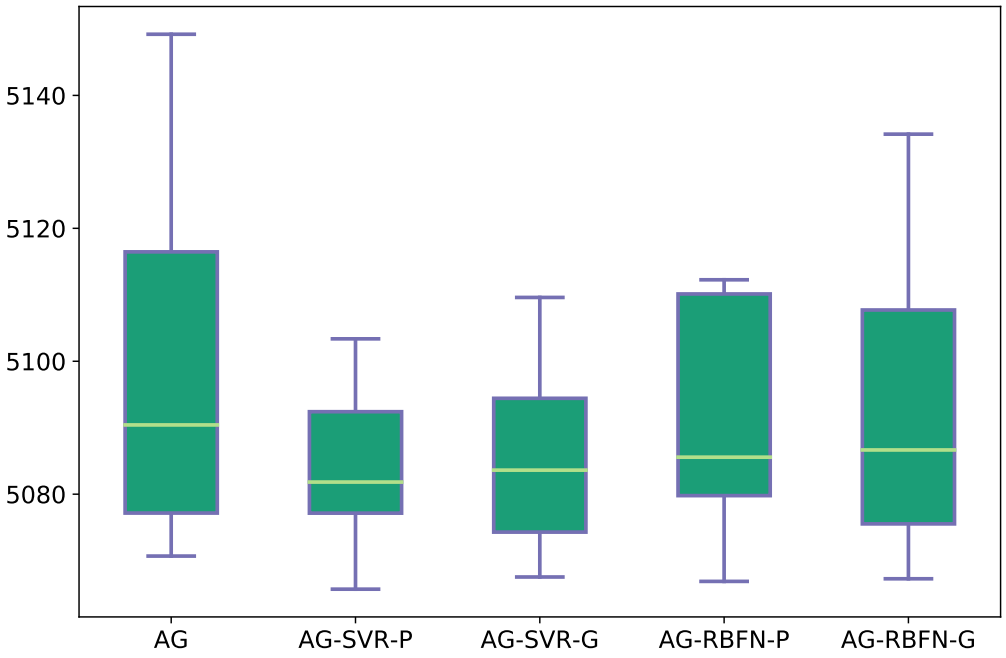



Figura 21 – Diagrama de caixa da Treliça de 10 Barras - Caso Contínuo.

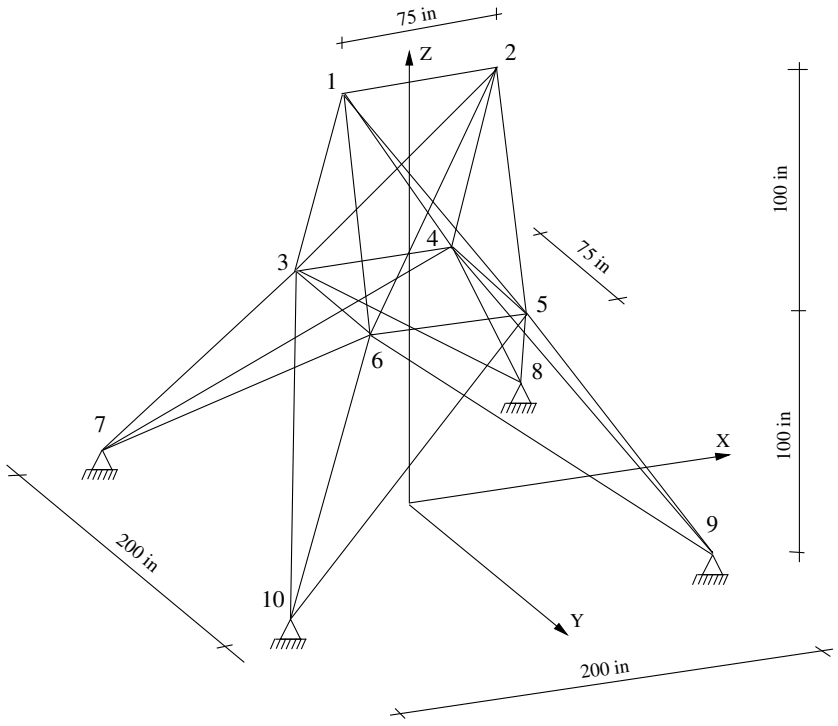



Figura 22 – Treliça de 25 barras.

### 6.3.1 Caso discreto

Para o caso discreto, as áreas das secções transversais são escolhidas dentro do conjunto $S = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7,$

Tabela 8 – Agrupamento de barras para a treliça de 25 barras.

| Grupo | Conectividade |
|---|---|
| $A_1$ | 1-2 |
| $A_2$ | 1-4, 1-5, 2-3, 2-6 |
| $A_3$ | 1-3, 1-6, 2-4, 2-5 |
| $A_4$ | 3-6, 4-5 |
| $A_5$ | 3-4, 5-6 |
| $A_6$ | 3-10, 4-9, 5-8, 6-7 |
| $A_7$ | 3-8, 4-7, 5-10, 6-9 |
| $A_8$ | 3-7, 4-8, 5-9, 6-10 |

Tabela 9 – Carga para a Treliça de 25 barras (kips).

| Nó | $F_x$ | $F_y$ | $F_z$ |
|---|---|---|---|
| 1 | 1 | -10.0 | -10.0 |
| 2 | 0 | -10.0 | -10.0 |
| 3 | 0.5 | 0 | 0 |
| 6 | 0.6 | 0 | 0 |

1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.8, 3.0, 3.2, 3.4}, com unidade $in^2$.

A Tabela 10 e a Figura 23 apresentam os resultados obtidos para a Treliça de 25 Barras no caso discreto. Observa-se que o AG e a proposta AG-SVR$^P$ e AG-SVR$^G$ produziram o melhor resultado. Percebe-se que a proposta AG-RBFN$^P$ obteve a melhor mediana e a proposta AG-SVR$^P$ obteve os melhores resultados em todas as métricas restantes. Apesar disso, observa-se, com base no Diagrama, que a proposta AG-SVR$^G$ possui uma variabilidade baixa.

Utilizando as soluções encontradas pelas 30 execuções independentes de cada algoritmo analisado, obtém-se no ANOVA o valor p-valor = 0.63086. Deste modo, como p-valor $> 0.1$, p-valor $> 0.05$ e p-valor $> 0.01$, não há evidências de diferenças significativas entre os algoritmos, ao nível $\alpha$ de significância utilizados.

Tabela 10 – Resultados da Treliça de 25 Barras – Caso Discreto.

| | Melhor | Pior | Média | Mediana | DP | $\bar{\Delta T}$ |
|---|---|---|---|---|---|---|
| AG | **484.8541** | 507.3520 | 489.9725 | 488.6034 | 5.7621 | 35.91s |
| AG-SVR$^P$ | **484.8541** | **497.1964** | **487.8341** | 486.2944 | **3.5353** | 41.15s |
| AG-SVR$^G$ | **484.8541** | 505.7603 | 488.5418 | 486.2561 | 5.3247 | 99.75s |
| AG-RBFN$^P$ | 485.0487 | 504.9635 | 488.6623 | **486.0998** | 5.4127 | 46.03s |
| AG-RBFN$^G$ | 485.0487 | 506.4038 | 489.1528 | 486.4599 | 6.1419 | 127.74s |

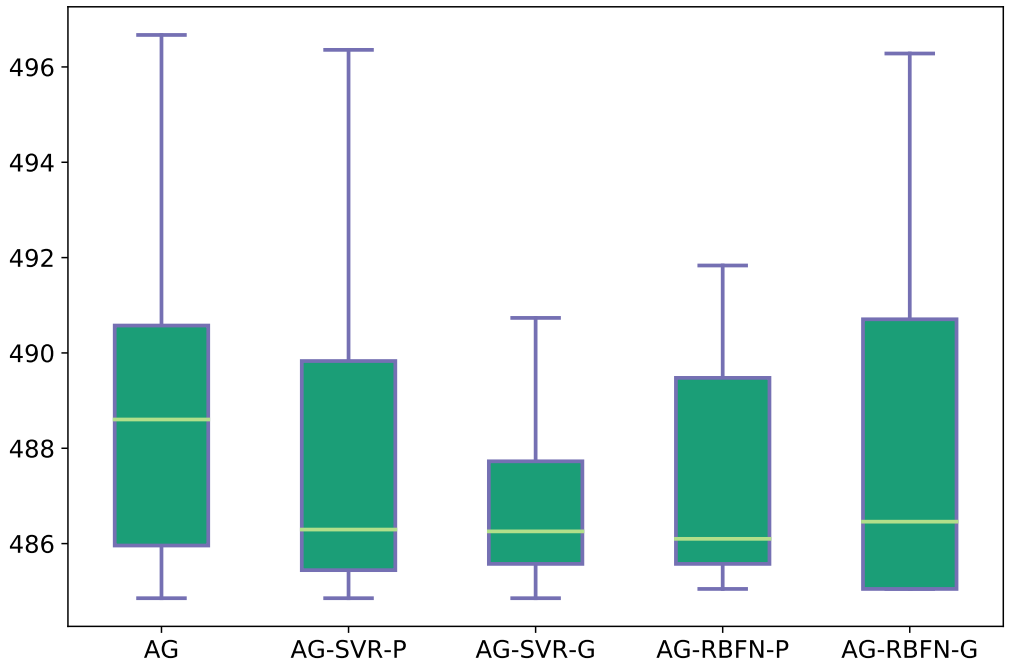



Figura 23 – Diagrama de caixa da Treliça de 25 Barras - Caso Discreto.

### 6.3.2 Caso contínuo

Para o caso contínuo, as áreas das secções transversais são escolhidas dentro do conjunto $S = \{x \in \mathbb{R} \mid 0.1 \leq x \leq 3.4\}$, com unidade $\text{in}^2$.

A Tabela 11 e a Figura 24 apresentam os resultados obtidos para a Treliça de 25 Barras no caso contínuo. Observa-se que a proposta AG-SVR$^G$ obteve o melhor resultado em relação à melhor solução. Além disso, a proposta AG-SVR$^P$ obteve o melhor resultado em relação à média e mediana e a proposta AG-RBFN$^P$ obteve o melhor valor entre as piores soluções e o menor DP.

Utilizando as soluções encontradas pelas 30 execuções independentes de cada algoritmo analisado, obtém-se no ANOVA o valor p-valor $= 0.56519$. Deste modo, como p-valor $> 0.1$, p-valor $> 0.05$ e p-valor $> 0.01$, não há evidências de diferenças significativas entre os algoritmos, ao nível $\alpha$ de significância utilizados.

Tabela 11 – Resultados da Treliça de 25 Barras – Caso Contínuo.

| | Melhor | Pior | Média | Mediana | DP | $\bar{\Delta T}$ |
|---|---|---|---|---|---|---|
| AG | 484.4022 | 498.5389 | 486.8777 | 486.0317 | 3.1483 | 35.76s |
| AG-SVR$^P$ | 484.4464 | 498.8288 | **485.9607** | **485.1765** | 2.5795 | 45.03s |
| AG-SVR$^G$ | **484.2616** | 495.4146 | 486.3799 | 485.6164 | 2.6163 | 77.59s |
| AG-RBFN$^P$ | 484.3963 | **494.6850** | 486.5196 | 485.9928 | **2.2067** | 50.82s |
| AG-RBFN$^G$ | 484.3867 | 506.8162 | 487.2904 | 485.6092 | 4.6312 | 100.58s |

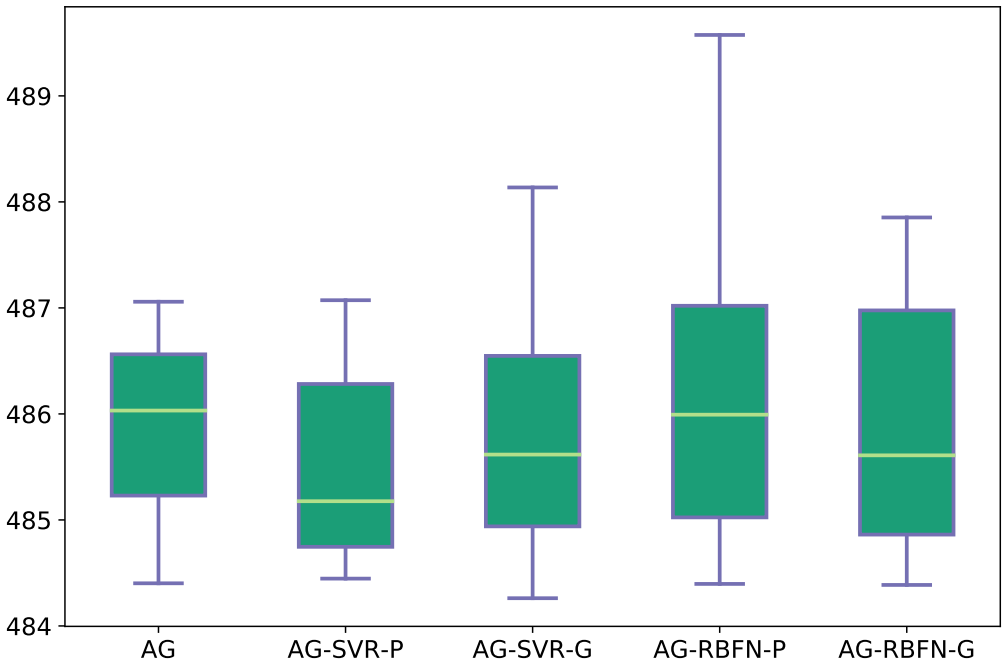



Figura 24 – Diagrama de caixa da Treliça de 25 Barras - Caso Contínuo.

## 6.4 Treliça de 60 barras

O problema de otimização dimensional da treliça anelar de 60 barras (Figura 25) consiste em encontrar o conjunto de áreas $A = \{A_1, \ldots, A_{25}\}$ que minimiza o peso $W(A)$ da estrutura. O raio externo do anel tem valor igual a 100 in e o raio interno tem valor igual a 90 in. Cada área representa um grupo de barras conforme mostrado na Tabela 12.

Tabela 12 – Agrupamento de barras para a treliça anelar de 60 barras.

| Grupo | Barras | Grupo | Barras |
|---|---|---|---|
| $A_1$ | 49 até 60 | $A_{14}$ | 25, 37 |
| $A_2$ | 1, 13 | $A_{15}$ | 26, 38 |
| $A_3$ | 2, 14 | $A_{16}$ | 27, 39 |
| $A_4$ | 3, 15 | $A_{17}$ | 28, 40 |
| $A_5$ | 4, 16 | $A_{18}$ | 29, 41 |
| $A_6$ | 5, 17 | $A_{19}$ | 30, 42 |
| $A_7$ | 6, 18 | $A_{20}$ | 31, 43 |
| $A_8$ | 7, 19 | $A_{21}$ | 32, 44 |
| $A_9$ | 8, 20 | $A_{22}$ | 33, 45 |
| $A_{10}$ | 9, 21 | $A_{23}$ | 34, 46 |
| $A_{11}$ | 10, 22 | $A_{24}$ | 35, 47 |
| $A_{12}$ | 11, 23 | $A_{25}$ | 36, 48 |
| $A_{13}$ | 12, 24 | | |

A tensão admissível em cada barra é de $\pm 60.0$ ksi e os deslocamentos admissíveis

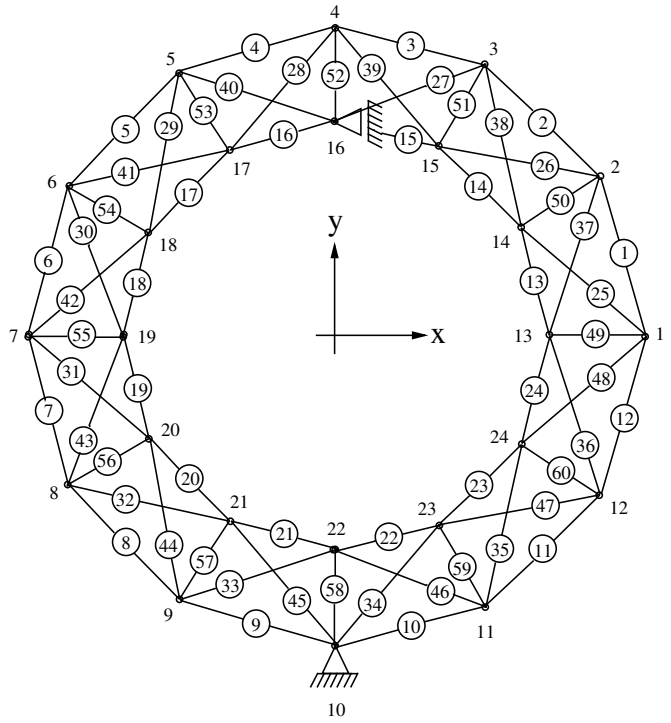



Figura 25 – Treliça de 60 barras.

são: 1.75 no nó 4, 2.25 no nó 13 e 2.75 no nó 19. O material possui massa específica ($\rho$) igual a 0.1 lb/in$^3$ e módulo de elasticidade (E) igual a 10.0$^4$ ksi. Três casos de carregamentos são aplicados sobre a Treliça de 60 barras, conforme apresentado na Tabela 13.

Tabela 13 – Carga para a Treliça de 60 barras (kips).

| Carga | Nó | $F_x$ | $F_y$ |
|---|---|---|---|
| 1 | 1 | -10.0 | 0 |
| | 7 | 9.0 | 0 |
| 2 | 15 | -8.0 | 3.0 |
| | 18 | -8.0 | 3.0 |
| 3 | 22 | -20.0 | 10.0 |

### 6.4.1 Caso discreto

Para o caso discreto, as áreas das secções transversais são escolhidas dentro do conjunto $S = \{0.5, 0.6, 0.7, ..., 5.0\}$, com unidade in$^2$.

A Tabela 14 e a Figura 26 apresentam os resultados obtidos para a Treliça de 60 Barras no caso discreto. Observa-se que a proposta AG-SVR$^G$ obteve os melhores resultados em todas as métricas utilizadas.

Utilizando as soluções encontradas pelas 30 execuções independentes de cada algoritmo analisado, obtém-se no ANOVA o valor p-valor = 0.01293. Avaliando a significância da estatística F através de p-valor, não foi possível encontrar evidências para apoiar a hipótese nula, ao nível de 5% de probabilidade, entre os algoritmos, com relação ao resultado. Rejeita-se, portanto, a hipótese de nulidade.

Tabela 14 – Resultados da Treliça de 60 Barras – Caso Discreto.

| | Melhor | Pior | Média | Mediana | DP | $\bar{\Delta T}$ |
|---|---|---|---|---|---|---|
| AG | 315.7745 | 348.7664 | 321.6147 | 320.7014 | 5.9001 | 57.36s |
| AG-SVR$^P$ | 316.9369 | 335.6703 | 322.3674 | 321.3578 | 5.1748 | 74.84s |
| AG-SVR$^G$ | **314.7910** | **327.0351** | **319.6437** | **319.3344** | **2.4726** | 94.35s |
| AG-RBFN$^P$ | 316.7392 | 330.2307 | 321.9561 | 321.6850 | 3.1176 | 77.53s |
| AG-RBFN$^G$ | 316.9557 | 334.0798 | 323.5181 | 323.2873 | 4.4553 | 148.39s |

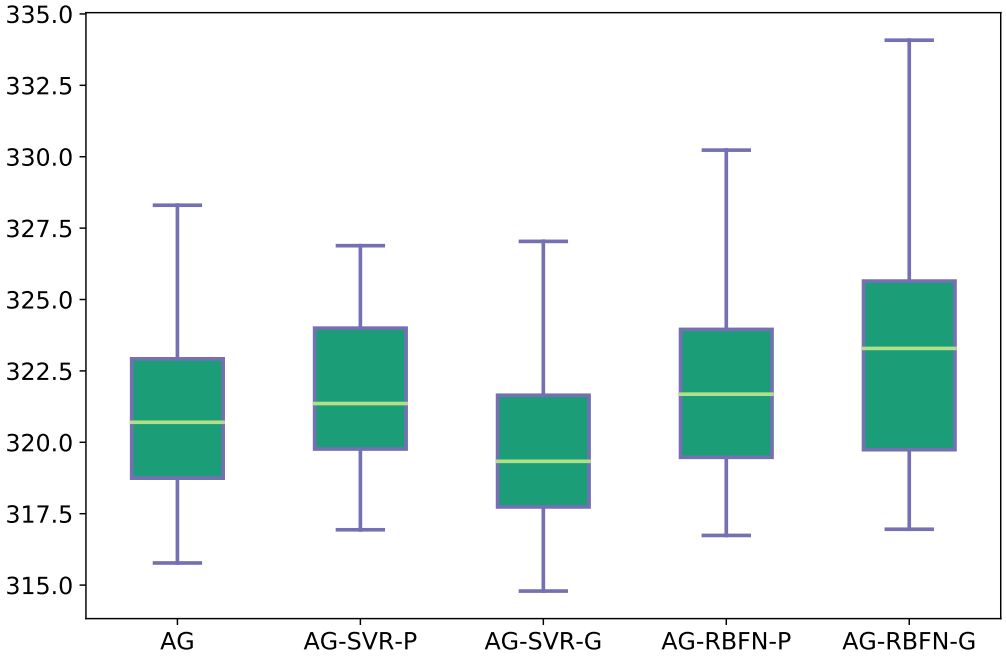



Figura 26 – Diagrama de caixa da Treliça de 60 Barras - Caso Discreto.

### 6.4.2 Caso contínuo

Para o caso contínuo, as áreas das secções transversais são escolhidas dentro do conjunto $S = \{x \in \mathbb{R} \mid 0.5 \leq x \leq 5.0\}$, com unidade in$^2$.

A Tabela 15 e a Figura 27 apresentam os resultados obtidos para a Treliça de 60 Barras no caso contínuo. Observa-se que a proposta AG-SVR$^G$ obteve os melhores resultados em todas as métricas utilizadas.

Utilizando as soluções encontradas pelas 30 execuções independentes de cada algoritmo analisado, obtém-se no ANOVA o valor p-valor = 0.03656. Avaliando a significância

da estatística F através de p-valor, não foi possível encontrar evidências para apoiar a hipótese nula, ao nível de 5% de probabilidade, entre os algoritmos, com relação ao resultado. Rejeita-se, portanto, a hipótese de nulidade.

Tabela 15 – Resultados da Treliça de 60 Barras – Caso Contínuo.

| | Melhor | Pior | Média | Mediana | DP | $\bar{\Delta T}$ |
|---|---|---|---|---|---|---|
| AG | 316.7800 | 330.5319 | 320.9386 | 320.4208 | 2.9703 | 56.03 |
| AG-SVR$^P$ | 316.1664 | 329.0764 | 320.2571 | 320.2034 | 2.8056 | 87.10s |
| AG-SVR$^G$ | **316.0679** | **322.5920** | **319.6154** | **319.7204** | **1.4315** | 113.62s |
| AG-RBFN$^P$ | 316.8969 | 331.8086 | 321.7266 | 321.3295 | 3.2758 | 88.35s |
| AG-RBFN$^G$ | 316.8612 | 336.3851 | 321.6863 | 320.6227 | 4.0800 | 170.09s |

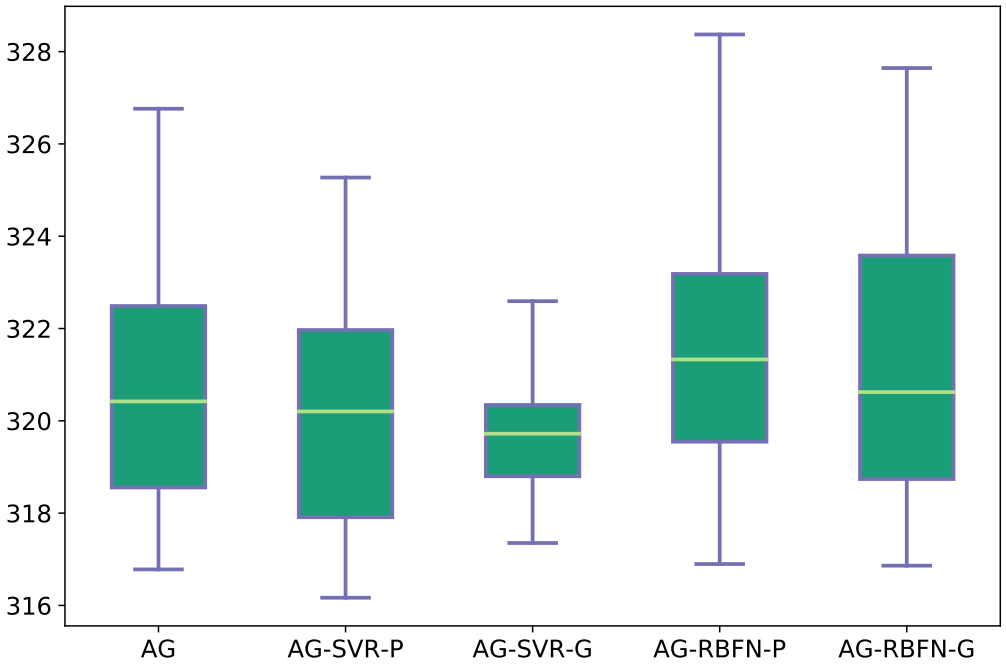



Figura 27 – Diagrama de caixa da Treliça de 60 Barras - Caso Contínuo.

## 6.5 Treliça de 72 barras

O problema de otimização dimensional da treliça de 72 barras (Figura 28) consiste em encontrar o conjunto de áreas $A = \{A_1, \ldots, A_{16}\}$ que minimiza o peso $W(A)$ da estrutura. Cada área representa um grupo de barras conforme mostrado na Tabela 16.

A tensão admissível em cada barra é de $\pm 25.0$ ksi e o deslocamento admissível nos nós 1 até 16 é de 0.25 in nas direções x e y. O material possui massa específica ($\rho$) igual a 0.1 lb/in$^3$ e módulo de elasticidade (E) igual a $10.0^4$ ksi. Dois casos de carregamentos são aplicados sobre a Treliça de 72 barras, conforme apresentado na Tabela 17.

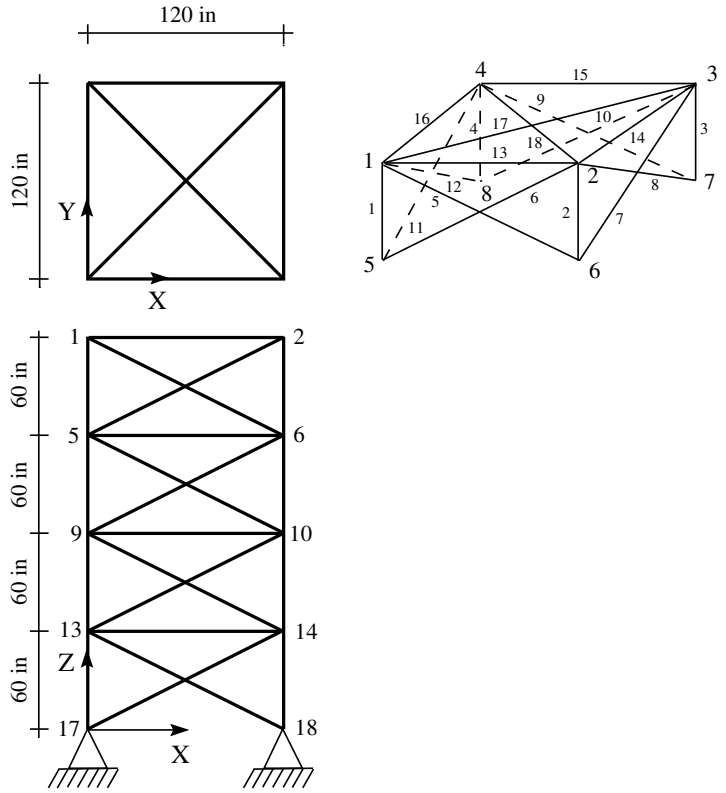



Figura 28 – Treliça de 72 barras.

Tabela 16 – Agrupamento de barras para a treliça de 72 barras.

| Grupo | Barra | Grupo | Barra |
|---|---|---|---|
| $A_1$ | 1, 2, 3, 4 | $A_9$ | 37, 38, 39, 40 |
| $A_2$ | 5, 6, 7, 8, 9, 10, 11, 12 | $A_{10}$ | 41, 42, 43, 44, 45, 46, 47, 48 |
| $A_3$ | 13, 14, 15, 16 | $A_{11}$ | , 49, 50, 51, 52 |
| $A_4$ | 17, 18 | $A_{12}$ | 53, 54 |
| $A_5$ | 19, 20, 21, 22 | $A_{13}$ | 55, 56, 57, 58 |
| $A_6$ | 23, 24, 25, 26, 27, 28, 29, 30 | $A_{14}$ | 59, 60, 61, 62, 63, 64, 65, 66 |
| $A_7$ | 31, 32, 33, 34 | $A_{15}$ | 67, 68, 69, 70 |
| $A_8$ | 35, 36 | $A_{16}$ | 71, 72 |

Tabela 17 – Carga para a Treliça de 72 barras (kips).

| Carga | Nó | $F_x$ | $F_y$ | $F_z$ |
|---|---|---|---|---|
| 1 | 1 | 5 | 5 | -5 |
| 2 | 1 | 0 | 0 | -5 |
| | 2 | 0 | 0 | -5 |
| | 3 | 0 | 0 | -5 |
| | 4 | 0 | 0 | -5 |

### 6.5.1 Caso discreto

Para o caso discreto, as áreas das secções transversais são escolhidas dentro do conjunto $S = \{0.1, 0.2, 0.3, ..., 2.5\}$, com unidade $\text{in}^2$.

A Tabela 18 e a Figura 29 apresentam os resultados obtidos para a Treliça de 72 Barras no caso discreto. Observa-se que todos os algoritmos produziram o melhor resultado. Além disso, percebe-se que o AG obteve os melhores resultados em todas as métricas.

Utilizando as soluções encontradas pelas 30 execuções independentes de cada algoritmo analisado, obtém-se no ANOVA o valor p-valor = 0.47481. Deste modo, como p-valor $> 0.1$, p-valor $> 0.05$ e p-valor $> 0.01$, não há evidências de diferenças significativas entre os algoritmos, ao nível $\alpha$ de significância utilizados.

Tabela 18 – Resultados da Treliça de 72 Barras – Caso Discreto

| | Melhor | Pior | Média | Mediana | DP | $\bar{\Delta T}$ |
|---|---|---|---|---|---|---|
| AG | **385.5426** | **390.4816** | **388.0597** | **387.9426** | **1.2491** | 55.20s |
| AG-SVR$^P$ | **385.5426** | 393.8757 | 388.3433 | **387.9426** | 2.0523 | 63.13s |
| AG-SVR$^G$ | **385.5426** | 392.8816 | 388.0657 | 388.0816 | 2.0858 | 85.53s |
| AG-RBFN$^P$ | **385.5426** | 394.8699 | 388.5172 | 388.0816 | 2.0457 | 65.64s |
| AG-RBFN$^G$ | **385.5426** | 396.2757 | 388.9594 | 388.0816 | 2.9172 | 107.92s |

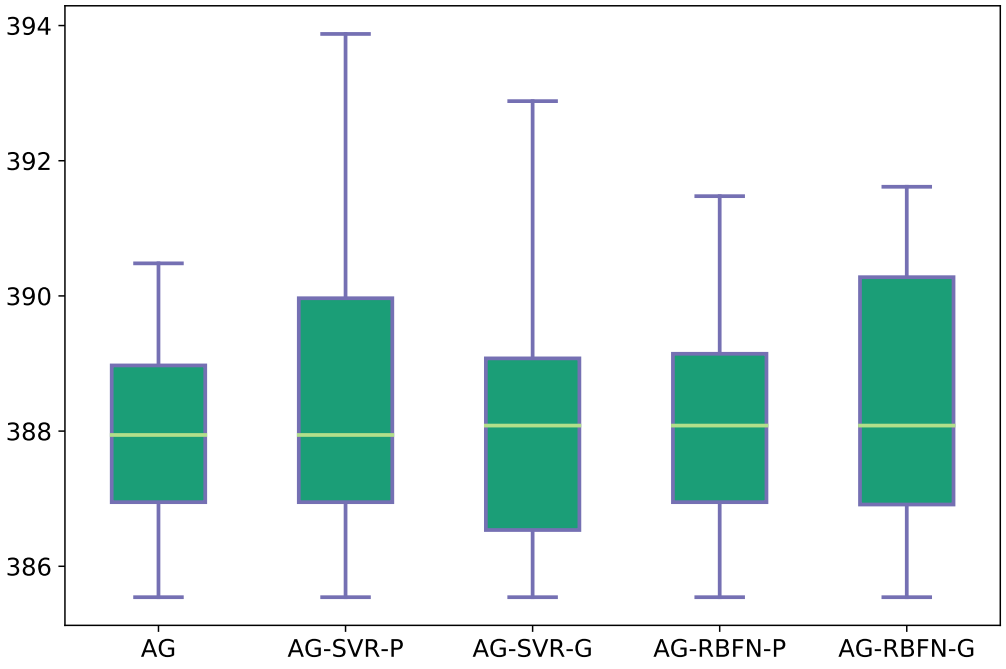



Figura 29 – Diagrama de caixa da Treliça de 72 Barras - Caso Discreto.

### 6.5.2 Caso contínuo

Para o caso contínuo, as áreas das secções transversais são escolhidas dentro do conjunto $S = \{x \in \mathbb{R} \mid 0.1 \leq x \leq 2.5\}$, com unidade $\text{in}^2$.

A Tabela 19 e a Figura 30 apresentam os resultados obtidos para a Treliça de 72 Barras no caso contínuo. Observa-se que a proposta AG-SVR$^P$ obteve o melhor resultado referente ao conjunto de melhores e piores soluções, média e mediana. O AG obteve o menor valor para o DP.

Utilizando as soluções encontradas pelas 30 execuções independentes de cada algoritmo analisado, obtém-se no ANOVA o valor p-valor = 0.34169. Deste modo, como p-valor $> 0.1$, p-valor $> 0.05$ e p-valor $> 0.01$, não há evidências de diferenças significativas entre os algoritmos, ao nível $\alpha$ de significância utilizados.

Tabela 19 – Resultados da Treliça de 72 Barras – Caso Contínuo

| | Melhor | Pior | Média | Mediana | DP | $\bar{\Delta T}$ |
|---|---|---|---|---|---|---|
| AG | 381.9382 | 384.5872 | 382.8630 | 382.8077 | **0.6620** | 54.97s |
| AG-SVR$^P$ | **380.6095** | **383.8723** | **382.5714** | **382.6957** | 0.6868 | 71.51s |
| AG-SVR$^G$ | 381.5848 | 384.1818 | 382.9531 | 383.0501 | 0.7354 | 100.88s |
| AG-RBFN$^P$ | 381.7313 | 384.4982 | 382.7827 | 382.7232 | 0.7148 | 77.87s |
| AG-RBFN$^G$ | 381.3455 | 384.7306 | 382.8224 | 382.9380 | 0.7771 | 141.56s |

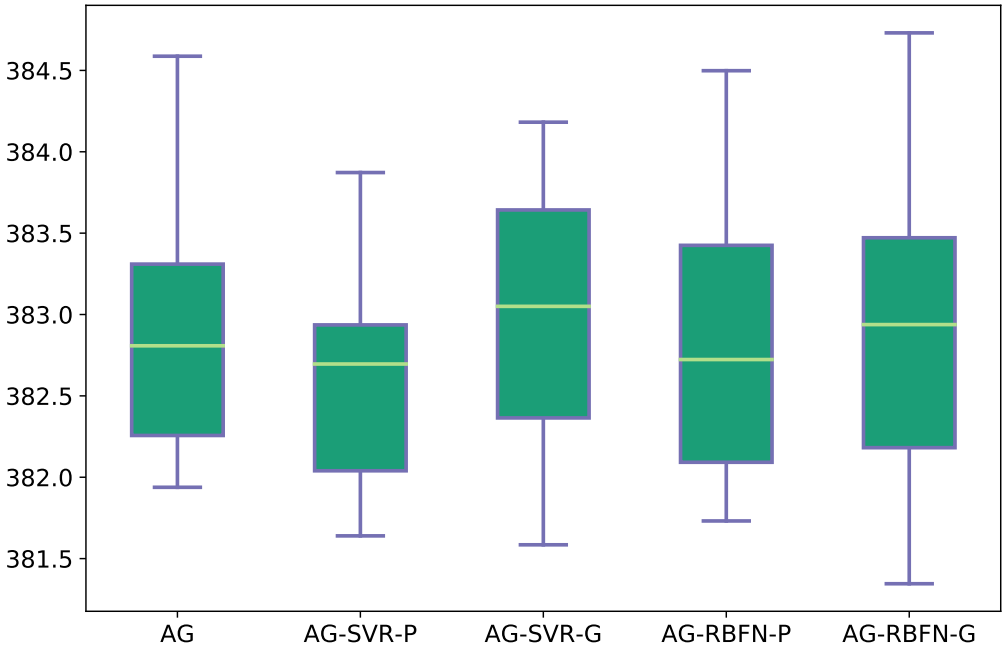



Figura 30 – Diagrama de caixa da Treliça de 72 Barras - Caso Contínuo.

## 6.6 Treliça de 942 barras

O problema de otimização dimensional da treliça de 942 barras (Figura 31) consiste em encontrar o conjunto de áreas $A = \{A_1, \ldots, A_{59}\}$ que minimiza o peso $W(A)$ da estrutura. A fim de manter a simetria do projeto, cada área representa um grupo de barras conforme mostrado na Tabela 20.

Tabela 20 – Agrupamento de barras para a treliça de 942 barras.

| Grupo | Barra | Grupo | Barra |
|---|---|---|---|
| $A_1$ | 1-2 | $A_{31}$ | 391-398 |
| $A_2$ | 3-10 | $A_{32}$ | 399-430 |
| $A_3$ | 11-18 | $A_{33}$ | 431-446 |
| $A_4$ | 19-34 | $A_{34}$ | 447-462 |
| $A_5$ | 35-46 | $A_{35}$ | 463-486 |
| $A_6$ | 47-58 | $A_{36}$ | 487-498 |
| $A_7$ | 59-82 | $A_{37}$ | 499-510 |
| $A_8$ | 83-86 | $A_{38}$ | 511-558 |
| $A_9$ | 87-90 | $A_{39}$ | 559-582 |
| $A_{10}$ | 91-98 | $A_{40}$ | 583-606 |
| $A_{11}$ | 99-106 | $A_{41}$ | 607-630 |
| $A_{12}$ | 107-122 | $A_{42}$ | 631-642 |
| $A_{13}$ | 123-130 | $A_{43}$ | 643-654 |
| $A_{14}$ | 131-162 | $A_{44}$ | 655-702 |
| $A_{15}$ | 163-170 | $A_{45}$ | 703-726 |
| $A_{16}$ | 171-186 | $A_{46}$ | 727-750 |
| $A_{17}$ | 187-194 | $A_{47}$ | 751-774 |
| $A_{18}$ | 195-226 | $A_{48}$ | 775-786 |
| $A_{19}$ | 227-234 | $A_{49}$ | 787-798 |
| $A_{20}$ | 235-258 | $A_{50}$ | 799-846 |
| $A_{21}$ | 259-270 | $A_{51}$ | 847-870 |
| $A_{22}$ | 271-318 | $A_{52}$ | 871-894 |
| $A_{23}$ | 319-330 | $A_{53}$ | 895-902 |
| $A_{24}$ | 331-338 | $A_{54}$ | 903-906 |
| $A_{25}$ | 339-342 | $A_{55}$ | 907-910 |
| $A_{26}$ | 343-350 | $A_{56}$ | 911-918 |
| $A_{27}$ | 351-358 | $A_{57}$ | 919-926 |
| $A_{28}$ | 359-366 | $A_{58}$ | 927-934 |
| $A_{29}$ | 367-382 | $A_{59}$ | 935-942 |
| $A_{30}$ | 383-390 | | |

A tensão normal máxima admissível em cada barra é de $\pm 25.0$ ksi e o deslocamento admissível nos quatro nós do topo é de 15.0 in. O material possui massa específica ($\rho$) igual a 0.1 lb/in$^3$ e módulo de elasticidade (E) igual a $10.0^4$ ksi.

A treliça de 942 barras está sujeita a uma única condição de carregamento consistindo em cargas horizontais e verticais, conforme segue: (i) As cargas verticais na direção

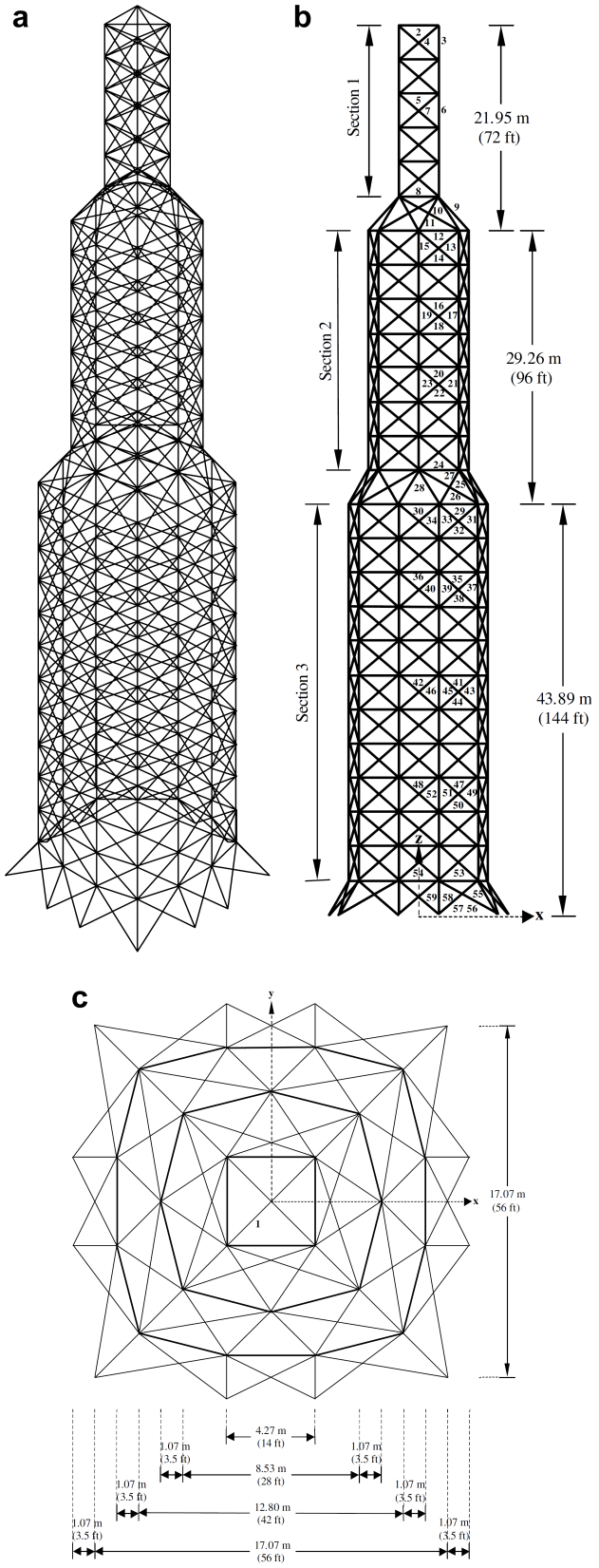



Figura 31 – Treliça de 942 barras.

z são $-3.0$ kips, $-6.0$ kips e $-9.0$ kips em cada nó na primeira, segunda e terceira seções,

respectivamente; (ii) As cargas laterais na direção y são $1,0$ kips em todos os nós da torre; (iii) As cargas laterais na direção x são 1.5 kips e 1.0 kips em cada nó nos lados esquerdo e direito da treliça, respectivamente.

### 6.6.1 Caso discreto

Para o caso discreto, as áreas das secções transversais são escolhidas dentro do conjunto $S = \{$ 1, 2, 3, ..., 200$\}$, com unidade $\text{in}^2$.

A Tabela 21 e a Figura 32 apresentam os resultados obtidos para a Treliça de 942 Barras no caso discreto. Observa-se que a proposta AG-SVR$^G$ obteve os melhores resultados em todas as métricas utilizadas.

Utilizando as soluções encontradas pelas 30 execuções independentes de cada algoritmo analisado, obtém-se no ANOVA o valor p-valor = 0.0. Avaliando a significância da estatística F através de p-valor, não foi possível encontrar evidências para apoiar a hipótese nula, ao nível de 1% de probabilidade, entre os algoritmos, com relação ao resultado. Rejeita-se, portanto, a hipótese de nulidade.

Tabela 21 – Resultados da Treliça de 942 Barras – Caso Discreto

| | Melhor | Pior | Média | Mediana | DP | $\bar{\Delta T}$ |
|---|---|---|---|---|---|---|
| AG | 177937.24 | 211208.29 | 193845.58 | 192932.10 | 9123.32 | 145.09s |
| AG-SVR$^P$ | 177521.36 | 206545.94 | 192308.67 | 191304.97 | 8019.35 | 172.76s |
| AG-SVR$^G$ | **172748.11** | **203175.91** | **181477.21** | **179366.76** | **6921.24** | 212.99s |
| AG-RBFN$^P$ | 177568.19 | 215117.81 | 197332.60 | 197149.31 | 8159.69 | 189.60s |
| AG-RBFN$^G$ | 180342.78 | 212256.35 | 194939.62 | 195367.49 | 7293.82 | 300.19s |

### 6.6.2 Caso contínuo

Para o caso contínuo, as áreas das secções transversais são escolhidas dentro do conjunto $S = \{x \in \mathbb{R} \mid 1 \leq x \leq 200\}$, com unidade $\text{in}^2$.

A Tabela 22 e a Figura 33 apresentam os resultados obtidos para a Treliça de 942 Barras no caso contínuo. Observa-se que a proposta em AG-SVR$^G$ obteve os melhores resultados em todas as métricas utilizadas.

Utilizando as soluções encontradas pelas 30 execuções independentes de cada algoritmo analisado, obtém-se no ANOVA o valor p-valor = 0.0. Avaliando a significância da estatística F através de p-valor, não foi possível encontrar evidências para apoiar a hipótese nula, ao nível de 1% de probabilidade, entre os algoritmos, com relação ao resultado. Rejeita-se, portanto, a hipótese de nulidade.

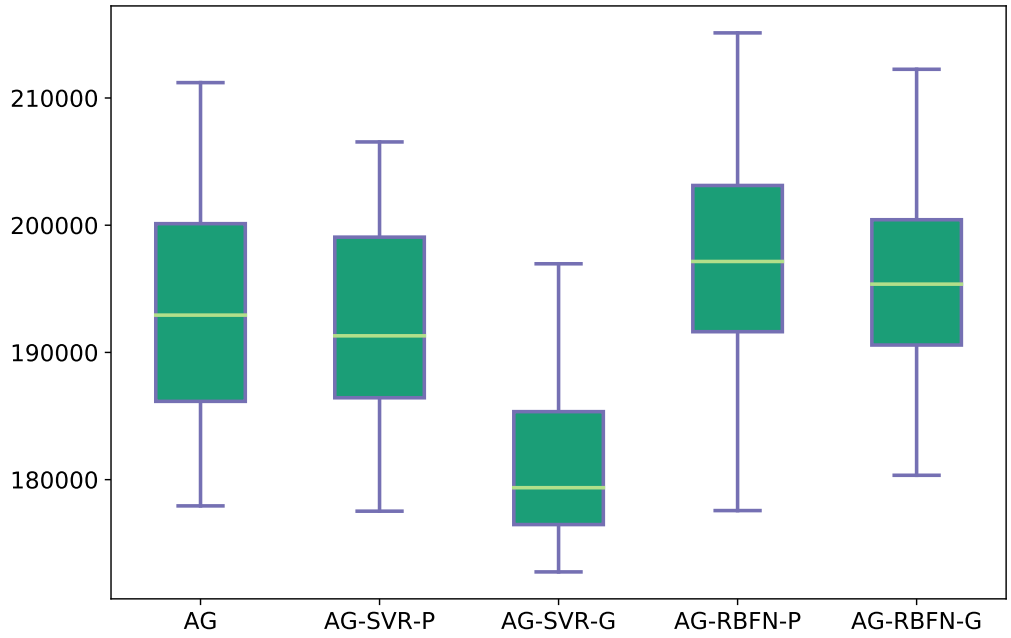



Figura 32 – Diagrama de caixa da Treliça de 942 Barras - Caso Discreto.

Tabela 22 – Resultados da Treliça de 942 Barras – Caso Contínuo

| | Melhor | Pior | Média | Mediana | DP | $\bar{\Delta T}$ |
|---|---|---|---|---|---|---|
| AG | 177432.66 | 213257.35 | 193619.33 | 193637.07 | 7699.37 | 142.73s |
| AG-SVR$^P$ | 173658.86 | 202174.85 | 191289.18 | 192021.79 | 6923.23 | 208.63s |
| AG-SVR$^G$ | **172670.57** | **190897.90** | **179729.22** | **179148.91** | **4747.59** | 262.56s |
| AG-RBFN$^P$ | 181064.70 | 204594.11 | 194079.25 | 194532.36 | 6787.74 | 223.63s |
| AG-RBFN$^G$ | 181431.17 | 212732.94 | 195955.08 | 195532.07 | 6652.13 | 345.52s |

## 6.7 Trocador de Calor

O problema de otimização de um modelo bidimensional de um Trocador de Calor (Figura 34) proposto por Thévenin e Janiga (2008) 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 . Uma configuração do Trocador de Calor é considerada utilizando a solução acoplada dos processos de transferência de fluxo e calor.

Um problema de CFD deve incluir um conjunto de condições de contorno adequado, assim como velocidade, temperatura, etc. Contudo, algumas vezes, é necessário utilizar uma condição de contorno artificial, visto que não é possível determinar o comportamento da região analisada (Zikanov, 2010), e uma possibilidade para esta condição de contorno artificial é atribuir pressão igual a zero.

Na parte superior, inferior e em todos os tubos, é aplicada uma condição de contorno de parede, em que 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

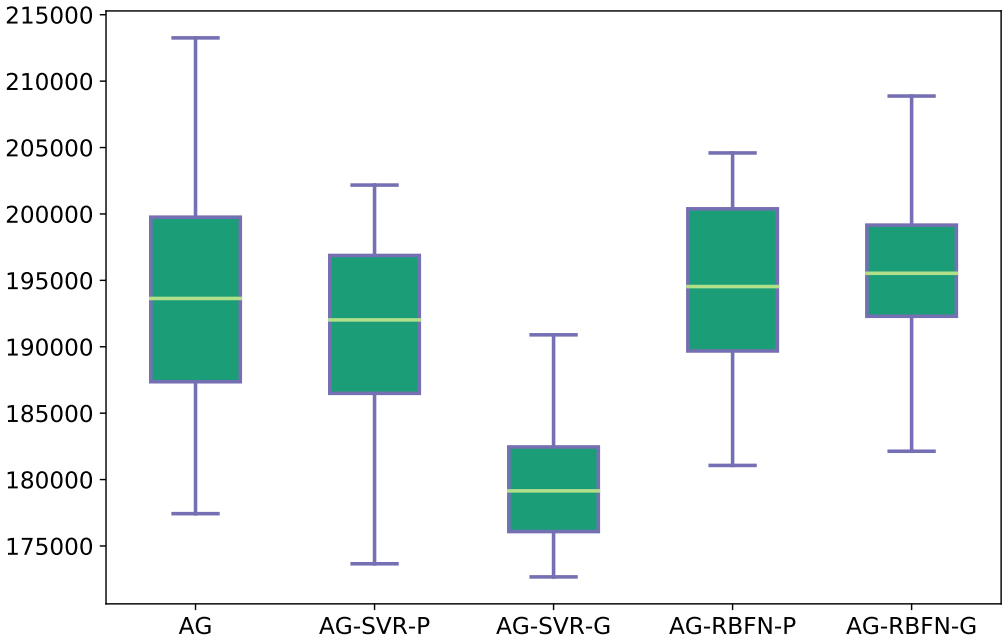



Figura 33 – Diagrama de caixa da Treliça de 942 Barras - Caso Contínuo.

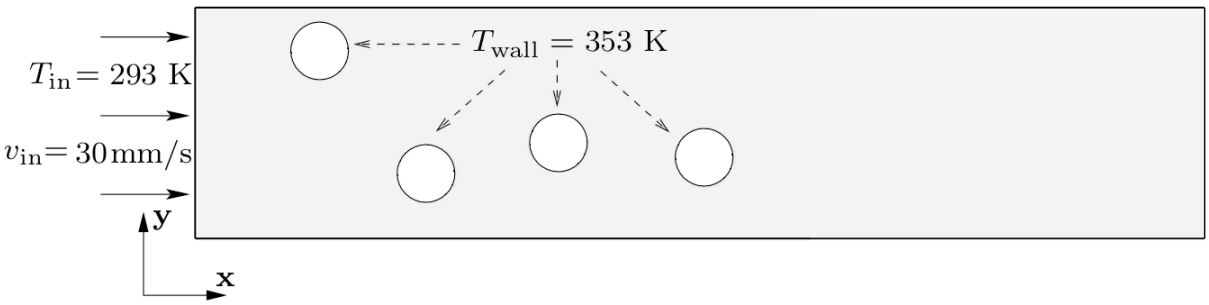



Figura 34 – Modelo Bidimensional de um Trocador de Calor.

fluxo gerado dentro do domínio, impossibilitando determinar o comportamento real do mesmo. Com isso, na saída, atribui-se pressão igual a zero como condição de contorno artificial.

O domínio do trocador de calor usado nos experimentos deste trabalho é $x = [0; 350]$ e $y = [0; 80]$, em milímetros, sendo que, dentro dele, são posicionados os tubos de raio $r = 10$ 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 uniforme igual a 30 mm/s. O problema de otimização deve garantir algumas restrições do problema, de modo que os tubos não podem encostar na fronteira do domínio e nem entre si. A Tabela 23 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, calculado utilizando o diâmetro do tubo e a velocidade uniforme na entrada do domínio.

Desta forma, pode-se afirmar que o escoamento dentro do domínio é laminar, visto que $Re = \frac{\rho v D}{\mu} < 2000$ (Incropera et al., 2000).

Tabela 23 – Trocador de calor - Propriedades do Ar.

| Propriedade do Ar | Valor |
|---|---|
| Densidade ($\rho$) | $1.2047x10^{-9}$ kg/mm$^3$ |
| Viscosidade Dinâmica ($\mu$) | $1.8205x10^{-8}$ 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 |

A malha utilizada neste problema possui entre 15.000 e 18.000 elementos triangulares, construídos automaticamente pelo *software* Abaqus®. A variação da quantidade de elementos está diretamente relacionada com as posições de todos os tubos cilíndricos.

Neste experimento, realiza-se a otimização mono-objetivo procurando a melhor configuração que maximiza a troca de calor. O trocador de calor deve possuir em sua configuração no mínimo um tubo e no máximo quatro tubos.

Como os tubos não podem encostar na fronteira do domínio e nem entre si, foi adotada a limitação 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:

$$
\begin{aligned}
&\text{min} && -\ \Delta\, Temp \\
&\text{sujeito a} && \\
& && 1 \ \leq\ n_{tubos} \ \leq\ 4 \\
& && 15 \ \leq\ x_i \ \leq 35,\ \forall\, i \ =\ 1,\ 2,\ 3\ e\ 4 \\
& && 15 \ \leq\ y_i \ \leq 65,\ \forall\, i \ =\ 1,\ 2,\ 3\ e\ 4
\end{aligned}
$$

e as posições $x_{ci}$ absolutas dos tubos são dadas pela seguinte equação (Figura 35):

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

O limite inferior e superior de $y_i$ é definido para garantir a restrição de contato com a fronteira. Já o limite inferior e superior de $x_i$ é definido para garantir que $x_{ci}$ satisfaça a restrição de contato com a fronteira, a restrição de contato entre os tubos e garantir que o final do domínio não possua tubos para observação do comportamento do fluido.

É importante notar que as posições absolutas dependem diretamente da quantidade de tubos presentes no sistema, o que possibilita manter o espaço de busca do problema,

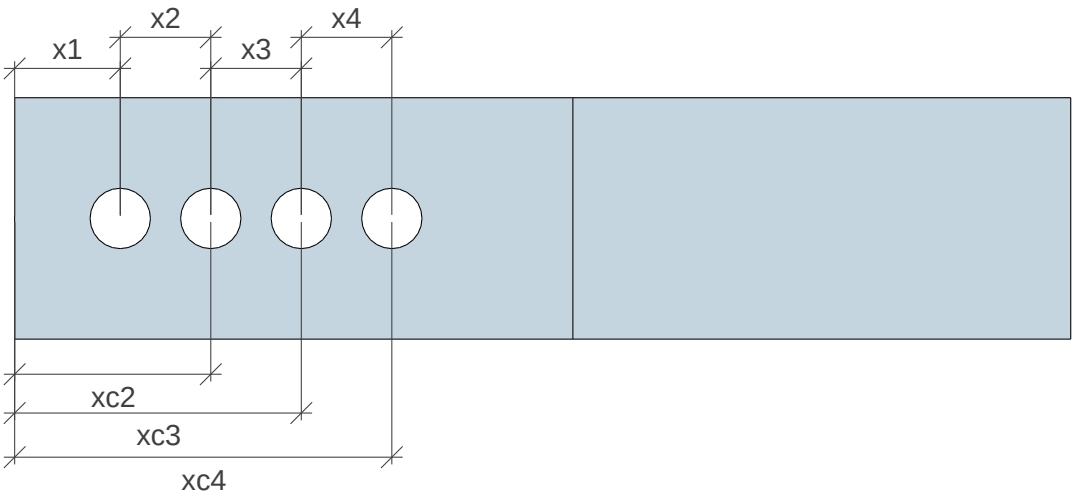



Figura 35 – Imagem Representativa da Distribuição dos Tubos.

independentemente do indivíduo analisado. Além disso, deve-se considerar o número de tubos como uma variável de projeto, sendo necessário introduzi-lo no processo de busca.

A Tabela 24 mostra os valores obtidos na otimização do trocador de calor. Observa-se que a proposta AG-RBFN$^P$ obteve o melhor resultado. Além disso, percebe-se que a proposta em todos os casos obteve resultados melhores do que o AG e que a diferença relativa de tempo das propostas em relação ao AG não é significativa para o problema.

Tabela 24 – Diferença de Temperatura finais do Trocador de Calor

| | Solução Encontrada | Tempo | Tempo/Tempo$_{AG}$ |
|---|---|---|---|
| AG | -28.3581 | 19538.47 s | 1.000 |
| AG-SVR$^P$ | -28.6291 | 19810.85 s | 1.014 |
| AG-SVR$^G$ | -29.2734 | 19840.50 s | 1.015 |
| AG-RBFN$^P$ | **-29.5992** | 19768.36 s | 1.012 |
| AG-RBFN$^G$ | -28.7944 | 19857.28 s | 1.016 |

A Figura 36 apresenta a configuração ótima encontrada do Trocador de Calor para cada algoritmo. Observa-se que, apesar de o AG, AG-SVR$^P$ e AG-SVR$^G$ apresentarem uma configuração final do trocador de calor similar, uma pequena variação entre eles gerou soluções diferentes, em que AG-SVR$^G$ se destacou por obter a melhor solução.

## 6.8 Perfis de desempenho

Além das análises realizadas anteriormente sobre cada problema teste, deseja-se verificar o desempenho de cada método de modo geral, identificando qual deles obteve o

melhor desempenho. Para isso, a comparação dos resultados é realizada através de uma ferramenta gráfica denominada perfis de desempenho (*Performance Profiles*).

Os perfis de desempenho foram propostos em (Dolan e Moré, 2002) com o intuito de facilitar a visualização e a compreensão dos resultados obtidos em experimentos com grande quantidade de dados. De modo geral, é feita uma comparação entre um conjunto de diferentes algoritmos para determinar aquele com o melhor desempenho sobre um certo conjunto de problemas.

Seja um conjunto de problemas teste $P = \{p_j | \forall j = 1, 2, \ldots, n_p\}$, um conjunto de algoritmos $S = \{s_i | \forall i = 1, 2, \ldots, n_s\}$ e $t_{p,s} > 0$ uma métrica de desempenho. A razão de desempenho é definida como:

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in S\}}$$

O perfil de desempenho de um algoritmo é definido como:

$$\rho_s(\tau) = \frac{1}{n_p} |p \in P : r_{p,s} \leq \tau|$$

onde $\rho_s(\tau)$ é a probabilidade de que a razão de desempenho $r_{p,s}$ do algoritmo $s \in S$ está dentro de um fator $\tau \geq 1$ do melhor desempenho, considerando todos os algoritmos em $S$.

A Figura 37 representa dois gráficos dos perfis de desempenho dos cinco métodos utilizados nesta dissertação, em que o **melhor** valor foi utilizado como a métrica de desempenho. O primeiro gráfico representa o início do intervalo, enquanto o segundo gráfico representa o intervalo completo. Para o problema do Trocador de Calor, foi utilizado o único valor obtido devido ao alto custo computacional para realizar mais execuções. Verifica-se que o AG-SVR$^G$ obteve o melhor desempenho em uma quantidade maior de problemas em relação aos outros métodos.

Os valores das áreas normalizadas sob as curvas dos perfis de desempenho utilizando o **melhor** valor podem ser observados na Figura 38 e na Tabela 25, em que se observa que AG-SVR$^G$ obteve o maior valor de área. Logo, AG-SVR$^G$ é considerado o método que possui o melhor desempenho global para a métrica e o conjunto de problemas utilizados.

Tabela 25 – Área normalizada sob as curvas dos perfis de desempenho utilizando o **melhor** valor.

| Método | AG-SVR$^G$ | AG-RBFN$^P$ | AG-SVR$^P$ | AG | AG-RBFN$^G$ |
|---|---|---|---|---|---|
| Área | 1.0 | 0.95701 | 0.92199 | 0.88056 | 0.88026 |

A Figura 39 representa o gráfico dos perfis de desempenho dos cinco métodos utilizados nesta dissertação, em que o valor da **média** foi utilizado como a métrica de desempenho. Verifica-se que o AG-SVR$^G$ obteve o melhor desempenho em uma quantidade maior de problemas em relação aos outros métodos.

Os valores das áreas normalizadas sob as curvas dos perfis de desempenho utilizando o valor da **média** podem ser observados na Figura 40 e na Tabela 26, em que se observa que AG-SVR$^G$ obteve o maior valor de área. Logo, AG-SVR$^G$ é considerado o método que possui o melhor desempenho global para a métrica e o conjunto de problemas utilizados.

Tabela 26 – Área normalizada sob as curvas dos perfis de desempenho utilizando o valor da **média**.

| Método | AG-SVR$^G$ | AG-RBFN$^P$ | AG-SVR$^P$ | AG-RBFN$^G$ | AG |
|---|---|---|---|---|---|
| Área | 1.0 | 0.88022 | 0.84925 | 0.80937 | 0.80684 |

## 6.9 Síntese dos resultados

Os resultados apresentados pelos experimentos numéricos, para os cinco algoritmos estudados, demonstram que o método híbrido proposto obteve um bom desempenho. Observa-se, pelas Figuras 38 e 40 e pelas Tabelas 25 e 26, que as propostas obtiveram um desempenho melhor do que o Algoritmo Genético, menos a proposta AG-RBFN$^G$ quando utiliza-se o **melhor** valor para análise.

Pode-se observar na Tabela 24 que a proposta AG-RBFN$^G$ demandou 5 horas e 30 minutos no processo de busca, sendo a proposta com maior tempo computacional. O AG obteve o menor tempo, demandando 5 horas e 25 minutos. Os 5 minutos de diferença não são significativos quando comparados com o tempo total gasto no processo de busca.

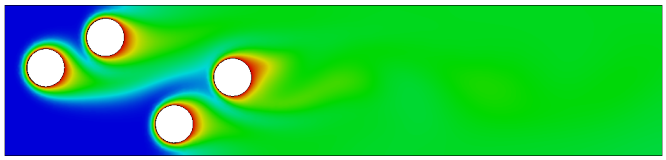
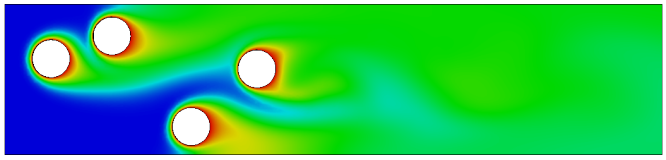
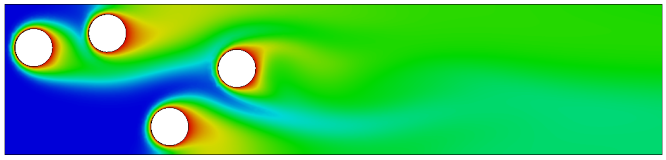
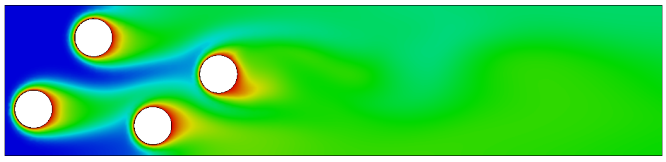
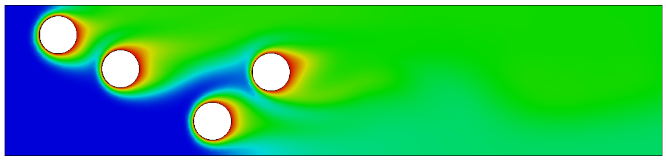
| Algoritmo utilizado e a configuração ótima encontrada |
| --- |
| AG |
|  |
| AG-SVR$^P$ |
|  |
| AG-SVR$^G$ |
|  |
| AG-RBFN$^P$ |
|  |
| AG-RBFN$^G$ |
|  |

Figura 36 – Configuração ótima encontrada do Trocador de Calor para cada algoritmo.

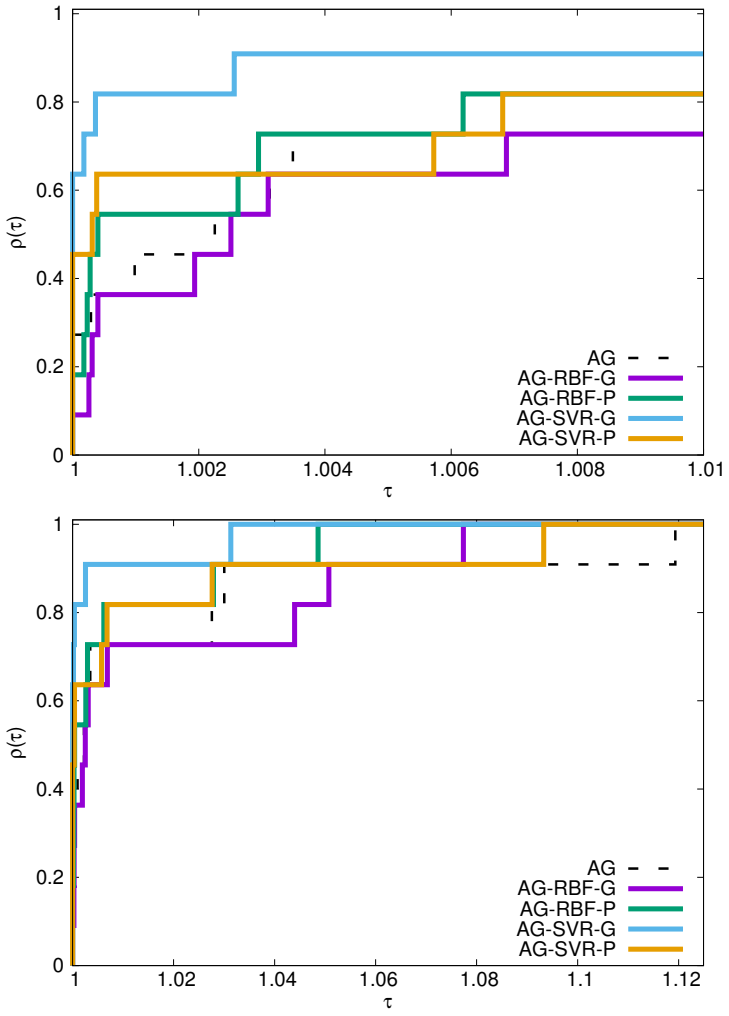



Figura 37 – Perfis de desempenho utilizando o **melhor** valor.

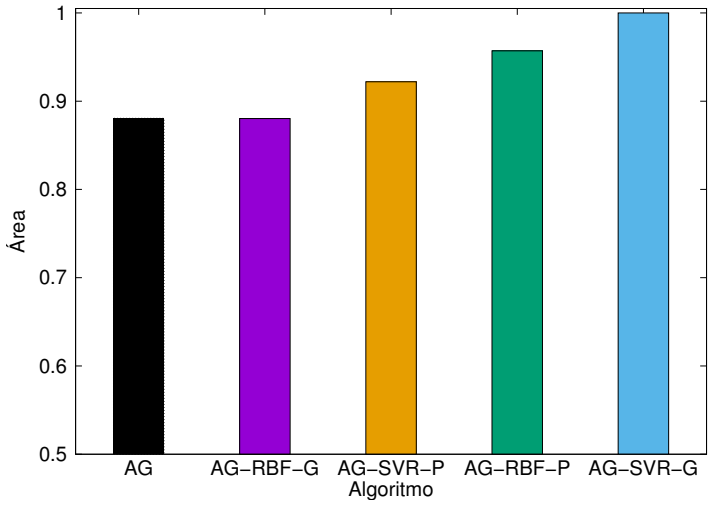



Figura 38 – Gráfico de barra da área normalizada sob as curvas dos perfis de desempenho utilizando o **melhor** valor.

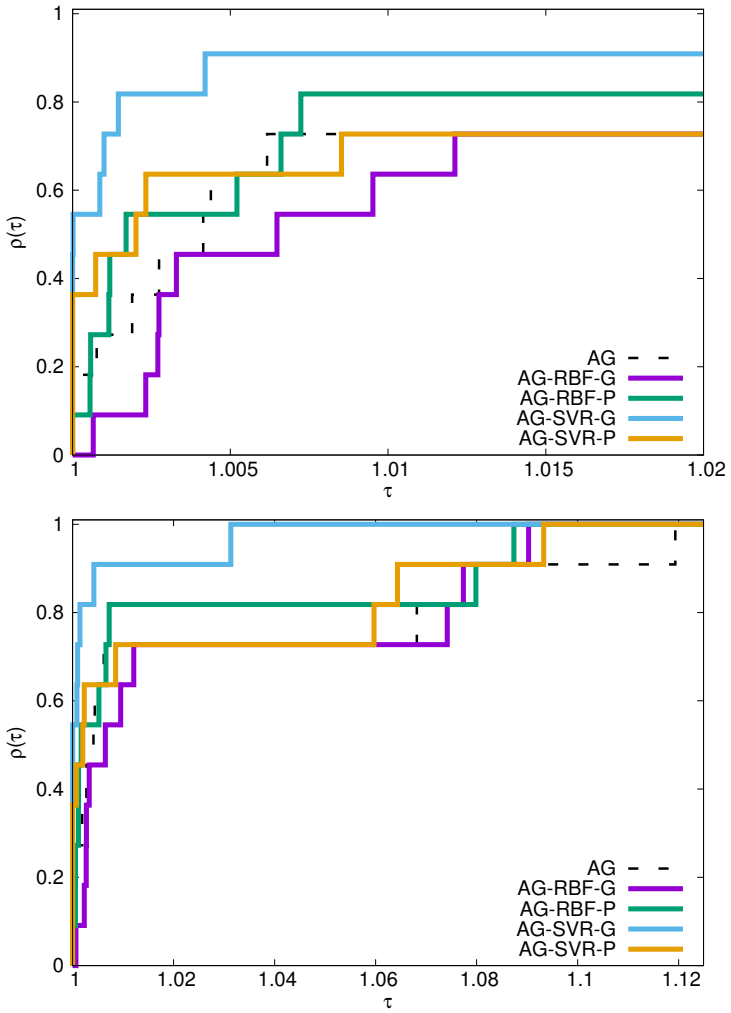



Figura 39 – Perfis de desempenho utilizando o valor da **média**.

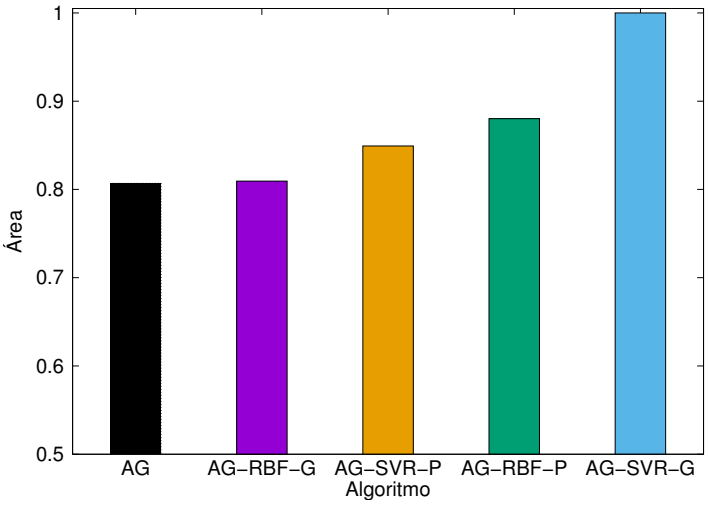



Figura 40 – Gráfico de barra da área normalizada sob as curvas dos perfis de desempenho utilizando o valor da **média**.

# 7 CONCLUSÃO

Nesta dissertação, foi proposto um método híbrido de otimização e um estudo sobre o desempenho do mesmo quando aplicado na resolução de problemas de otimização que demandam alto custo computacional.

Para a construção do método híbrido, o Algoritmo Genético e o algoritmo L-BFGS-B foram utilizados como algoritmos de busca e a Máquina de Vetor Suporte para Regressão e a Rede Neural de Função de Base Radial foram utilizadas como técnica de aprendizado de máquina.

Para avaliar a qualidade do método híbrido, foram feitos experimentos numéricos referentes a problemas de otimização estrutural tradicionalmente discutidos na literatura. Estes experimentos são referentes a minimizar o peso das treliças de 10, 25, 60, 72 e 942 barras para o caso em que as variáveis de projeto assumem valores discretos e contínuos e, por fim, a maximização da troca de calor em um trocador de calor, como um problema de otimização da dinâmica dos fluidos computacional.

Os experimentos demonstraram que a utilização do método híbrido de otimização resultou em um aumento do tempo computacional quando comparado com o Algoritmo Genético. Este comportamento é esperado devido à inclusão do aprendizado de máquina no processo de busca, sendo necessário realizar o treinamento da técnica a cada geração e a busca de parâmetros ótimos realizados em algumas gerações.

O método híbrido proposto é aconselhável para problemas de otimização que possuem alto custo computacional, de modo que o tempo gasto com o aprendizado de máquina não é significativo no processo de busca.

A proposta AG-SVR$^G$ foi a que apresentou o melhor desempenho para solucionar os experimentos propostos. Já a proposta AG-RBFN$^P$ foi a que apresentou o segundo melhor desempenho e o seu custo de computacional é menor do que a proposta AG-SVR$^G$ devido ao fato de não realizar a busca de parâmetros no processo geracional.

Os resultados demonstraram que a busca de parâmetros no processo geracional nem sempre produz resultados melhores do que a busca de parâmetros na primeira geração. Ao se utilizar a técnica RBFN com a busca de parâmetros na primeira geração, observa-se que o desempenho do método híbrido foi melhor do que a técnica RBFN com busca de parâmetros no processo geracional. O inverso ocorre ao se utilizar o SVR.

Destaca-se que o objetivo principal desta dissertação foi a proposta de um método híbrido de otimização capaz de solucionar problemas com um alto custo computacional e, deste modo, a proposta AG-SVR$^G$ foi a que apresentou melhor resultado de modo geral.

As sugestões para trabalhos futuros são:

- Utilização de outra técnica de busca baseada em populações, como o método de Enxame de Partículas (*Particle Swarm Optimization* - PSO) ou Evolução Diferencial (*Differential Evolution* - DE);
- Emprego de outra técnica de busca local, podendo ser aplicados algoritmos evolutivos (AG, PSO e DE);
- Uso de outras técnicas de aprendizado de máquina, como o MLP, o Processo Gaussiano (*Kriging*) e a Árvore de Decisão;
- Aplicação de outra técnica de busca de parâmetros, como *Random Search* e Validação Cruzada;
- Verificação de qual proposta converge para uma solução preestabelecida de um problema de otimização em um menor número de gerações percorridas;
- Aplicação do método híbrido em problemas de engenharia mais complexos.

# REFERÊNCIAS


Adida, E. e Perakis, G. (2006). A robust optimization approach to dynamic pricing and inventory control with no backorders. *Mathematical Programming*, 107(1):97–129.

Ahmed, S., Ramm, G., e Faltin, G. (1984). Some salient features of the time-averaged ground vehicle wake. Technical report, SAE Technical Paper.

Aizerman, M. A., Braverman, E. A., e Rozonoer, L. (1964). Theoretical foundations of the potential function method in pattern recognition learning. Em *Automation and Remote Control,*, number 25 in Automation and Remote Control„ pgs. 821–837.

Albrecht, C. H. (2005). *Algoritmos Evolutivos aplicados a síntese e otimização de sistemas de ancoragem*. PhD thesis, Universidade Federal do Rio de Janeiro, Rio de Janeiro.

Alexandrov, N. M., Dennis, J., Lewis, R. M., e Torczon, V. (1998). A trust-region framework for managing the use of approximation models in optimization. *Structural and Multidisciplinary Optimization*, 15(1):16–23.

Andersen, O. W. (1967). Optimum design of electrical machines. *IEEE Transactions on Power Apparatus and Systems*, PAS-86(6):707–711.

Arora, J. (1989). *Introduction to optimum design*. Academic Press.

Back, T. (1996). *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press.

Balakrishnan, V. e Boyd, S. (1993). Global optimization in control systems analysis. Em *Aerospace Control Systems, 1993. Proceedings. The First IEEE Regional Conference on*, pgs. 421–425. IEEE.

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

Barbosa, H. J. e Lemonge, A. C. (2008). An adaptive penalty method for genetic algorithms in constrained optimization problems. Em *Frontiers in Evolutionary Robotics*. InTech.

Bennett, K. P. e Mangasarian, O. L. (1992). Robust linear programming discrimination of two linearly inseparable sets. *Optimization methods and software*, 1(1):23–34.

Billings, S. A. e Zheng, G. L. (1995). Radial basis function network configuration using genetic algorithms. *Neural Networks*, 8(6):877–890.

Blanning, R. W. (1974). The sources and uses of sensitivity information. *Interfaces*, 4(4):32–38.

Blanning, R. W. (1975). Response to michel, kleijnen and permut. *Interfaces*, 5(3):24–25.

Booker, A. J., Dennis, J. E., Frank, P. D., Serafini, D. B., Torczon, V., e Trosset, M. W. (1999). A rigorous framework for optimization of expensive functions by surrogates. *Structural and Multidisciplinary Optimization*, 17(1):1–13.

Boukouvala, F. e Ierapetritou, M. G. (2013). Surrogate-based optimization of expensive flowsheet modeling for continuous pharmaceutical manufacturing. *Journal of Pharmaceutical Innovation*, 8(2):131–145.

Broomhead, D. e Lowe, D. (1988a). Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355.

Broomhead, D. S. e Lowe, D. (1988b). Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, Royal Signals and Radar Establishment Malvern (United Kingdom).

Büche, D., Schraudolph, N. N., e Koumoutsakos, P. (2003). Accelerating evolutionary algorithms using fitness function models. Em *Proc. GECCO Workshop Learn., Adapt. Approx. Evol. Comput*, pgs. 166–169.

Büche, D., Schraudolph, N. N., e Koumoutsakos, P. (2005). Accelerating evolutionary algorithms with gaussian process fitness function models. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 35(2):183–194.

Burke, E. K., Gustafson, S., e Kendall, G. (2004). Diversity in genetic programming: an analysis of measures and correlation with fitness. *IEEE Transactions on Evolutionary Computation*, 8(1):47–62.

Byrd, R. H., Lu, P., Nocedal, J., e Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208.

Cardarelli, E. e Pelagagge, P. M. (1995). Simulation tool for design and management optimization of automated interbay material handling and storage systems for large wafer fab. *IEEE Transactions on Semiconductor Manufacturing*, 8(1):44–49.

Carson, Y. e Maria, A. (1997). Simulation optimization: methods and applications. Em *Proceedings of the 29th conference on Winter simulation*, pgs. 118–126. IEEE Computer Society.

Cochocki, A. e Unbehauen, R. (1993). *Neural networks for optimization and signal processing*. John Wiley & Sons, Inc.

Coppola, A. e Stewart, B. M. (2014). lbfgs: Efficient l-bfgs and owl-qn optimization in r. `https://cran.r-project.org/web/packages/lbfgs/vignettes/Vignette.pdf`.

Cortes, C. e Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Cressie, N. (1990). The origins of kriging. *Mathematical geology*, 22(3):239–252.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314.

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. e Agrawal, R. B. (1995). Simulated binary crossover for continuous search space. *Complex Systems*, pgs. 115–148.

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

Dolan, E. D. e Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213.

Dorigo, M. e Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation*, 1(1):53–66.

Drucker, H., Burges, C. J., Kaufman, L., Smola, A. J., e Vapnik, V. (1997). Support vector regression machines. Em *Advances in neural information processing systems*, pgs. 155–161.

El-Beltagy, M. A., Nair, P. B., e Keane, A. J. (1999). Metamodeling techniques for evolutionary optimization of computationally expensive problems: Promises and limitations. Em *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1*, pgs. 196–203. Morgan Kaufmann Publishers Inc.

Estes, A. C. e Frangopol, D. M. (2001). Minimum expected cost-oriented optimal maintenance planning for deteriorating structures: Application to concrete bridge decks. *Reliability Engineering & System Safety*, 73(3):281–291.

Fesanghary, M., Damangir, E., e Soleimani, I. (2009). Design optimization of shell and tube heat exchangers using global sensitivity analysis and harmony search algorithm. *Applied Thermal Engineering*, 29(5):1026–1031.

Fletcher, R. (1987). *Practical Methods of Optimization; (2Nd Ed.)*. Wiley-Interscience, New York, NY, USA.

Fonseca, L. G. d. (2009). *Algoritmos genéticos assistidos por metamodelos baseados em similaridade*. PhD thesis, LNCC/Modelagem Computacional.

Forrester, A., Keane, A., et al. (2008). *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons.

Fox, R. e Willmert, K. (1967). Optimum design of curve-generating linkages with inequality constraints. *Journal of Engineering for Industry*, 89(1):144–152.

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

Gelle, E. M., Faltings, B. V., e Smith, I. F. (2000). Structural engineering design support by constraint satisfaction. Em *Artificial Intelligence in Design'00*, pgs. 311–331. Springer.

Giannakoglou, K. (2002). Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence. *Progress in Aerospace Sciences*, 38(1):43–76.

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

Goldberg, D. E., Deb, K., e Clark, J. H. (1991). Genetic algorithms, noise, and the sizing of populations. *Urbana*, 51:61801.

Goldberg, D. E., Deb, K., e Clark, J. H. (2014). Accounting for noise in the sizing of populations. *Whitley*, 2419:127–140.

Grefenstette, J. J. e Fitzpatrick, J. M. (1985). Genetic search with approximate function evaluations. Em *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, pgs. 112–120.

Grossmann, I. e Sargent, R. (1978). Optimum design of chemical plants with uncertain parameters. *AIChE Journal*, 24(6):1021–1028.

Gunn, S. R. et al. (1998). Support vector machines for classification and regression. *ISIS technical report*, 14:85–86.

Hastie, T. (1998). *Neural network*. Wiley Online Library.

Haykin, S. (1994). *Neural networks: a comprehensive foundation*. Prentice Hall PTR.

Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., e Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28.

Hegazy, T. (1999). Optimization of resource allocation and leveling using genetic algorithms. *Journal of construction engineering and management*, 125(3):167–175.

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

Holland, J. H. (1995). *Hidden Order: How Adaptation Builds Complexity*. Helix books. Basic Books.

Incropera, F. P., DeWitt, D. P., e Bergman, T. L. (2000). *Fundamentos de Transferência de Calor E de Massa* . Grupo Gen-LTC.

Jin, Y., Olhofer, M., e Sendhoff, B. (2002). A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation*, 6(5):481–494.

Karakasis, M. K., Giotis, A. P., e Giannakoglou, K. C. (2003). Inexact information aided, low-cost, distributed genetic algorithms for aerodynamic shape optimization. *International Journal for Numerical Methods in Fluids*, 43(10-11):1149–1166.

Kleijnen, J. P. C. (1975). A comment on blanning's "metamodel for sensitivity analysis: The regression metamodel in simulation". *Interfaces*, 5(3):21–23.

Knight, U. t. (1960). The logical design of electrical networks using linear programming methods. *Proceedings of the IEE-Part A: Power Engineering*, 107(33):306–314.

Kovács, Z. L. (2002). *Redes neurais artificiais*. Editora Livraria da Física.

Koza, J. R. (1992). *Genetic programming: on the programming of computers by means of natural selection*. Complex adaptive systems. MIT Press, 1 edition.

Krempser, E., Bernardino, H. S., Barbosa, H., e Lemonge, A. C. d. C. (2017). Performance evaluation of local surrogate models in differential evolution based optimum design of truss structures. *Engineering Computations*, 34(2).

Kubat, M. (1998). Decision trees can initialize radial-basis function networks. *IEEE Transactions on Neural Networks*, 9(5):813–821.

Kuo, M.-S. (2011). Optimal location selection for an international distribution center by using a new hybrid method. *Expert Systems with Applications*, 38(6):7208–7221.

Lee, Y.-J. e Mangasarian, O. L. (2001). Ssvm: A smooth support vector machine for classification. *Computational optimization and Applications*, 20(1):5–22.

Lemonge, A. (1999). *Aplicação de Algoritmos Genéticos em Problemas de Otimização Estrutural*. PhD thesis, Programa de Engenharia Civil da COPPE/UFRJ, Rio de Janeiro, Brasil.

Lemonge, A. C. e Barbosa, H. J. (2004). An adaptive penalty scheme for genetic algorithms in structural optimization. *International Journal for Numerical Methods in Engineering*, 59(5):703–736.

Lesh, F. H. (1959). Multi-dimensional least-squares polynomial curve fitting. *Commun. ACM*, 2(9):29–30.

Liu, D. C. e Nocedal, J. (1989). On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45(1):503–528.

Liu, J., Song, W.-P., Han, Z.-H., e Zhang, Y. (2017). Efficient aerodynamic shape optimization of transonic wings using a parallel infilling strategy and surrogate models. *Structural and Multidisciplinary Optimization*, 55(3):925–943.

McCulloch, W. S. e Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

Mezura-Montes, E. e Coello, C. A. C. (2011). Constraint-handling in nature-inspired numerical optimization: past, present and future. *Swarm and Evolutionary Computation*, 1(4):173–194.

Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs*. Artificial Intelligence. Springer Berlin Heidelberg.

Michel, A. J. e Permut, S. E. (1975). A comment on blanning's "the sources and uses of sensitivity information". *Interfaces*, 5(3):19–20.

Minsky, M. e Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, USA.

Mitchell, M. (1998). *An Introduction to Genetic Algorithms*. Complex Adaptive Systems. The MIT Press, third printing edition.

Moody, J. e Darken, C. J. (1989). Fast learning in networks of locally-tuned processing units. *Neural computation*, 1(2):281–294.

Mukherjee, I. e Ray, P. K. (2006). A review of optimization techniques in metal cutting processes. *Computers & Industrial Engineering*, 50(1):15–34.

Nilsson, N. J. (1965). *Learning machines: foundations of trainable pattern-classifying systems*. McGraw-Hill.

Ong, Y. S., Nair, P. B., e Keane, A. J. (2003). Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA journal*, 41(4):687–696.

Patrick, J. e Puterman, M. L. (2008). Reducing wait times through operations research: optimizing the use of surge capacity. *Healthcare Policy*, 3(3):75.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830.

Prager, W. e Shield, R. (1967). A general theory of optimal plastic design. *Journal of Applied Mechanics*, 34(1):184–186.

Qian, Z., Seepersad, C. C., Joseph, V. R., Allen, J. K., e Wu, C. J. (2006). Building surrogate models based on detailed and approximate simulations. *Journal of Mechanical Design*, 128(4):668–677.

Quiñonero-Candela, J. e Rasmussen, C. E. (2005). A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6(Dec):1939–1959.

Ramteen Sioshansi, A. J. C. a. (2017). *Optimization in Engineering: Models and Algorithms*. Springer Optimization and Its Applications 120. Springer International Publishing, 1 edition.

Rao, S. S. e Rao, S. S. (2009). *Engineering optimization: theory and practice*. John Wiley & Sons.

Reuther, J., Jameson, A., Farmer, J., Martinelli, L., e Saunders, D. (1996). Aerodynamic shape optimization of complex aircraft configurations via an adjoint formulation. *AIAA paper*, 94.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Rumelhart, D. E., Hinton, G. E., e Williams, R. J. (1988). Learning representations by back-propagating errors. Em Anderson, J. A. e Rosenfeld, E., editores, *Neurocomputing: Foundations of Research*, pgs. 696–699. MIT Press, Cambridge, MA, USA.

Sachdeva, A., Kumar, D., e Kumar, P. (2008). Planning and optimizing the maintenance of paper production systems in a paper plant. *Computers & Industrial Engineering*, 55(4):817–829.

Salajegheh, E. e Heidari, A. (2005). Optimum design of structures against earthquake by wavelet neural network and filter banks. *Earthquake engineering & structural dynamics*, 34(1):67–82.

Saunders, C., Gammerman, A., e Vovk, V. (1998). Ridge regression learning algorithm in dual variables. *ICML-1998 Proceedings of the 15th International Conference on Machine Learning*.

Savic, D. A. e Walters, G. A. (1997). Genetic algorithms for least-cost design of water distribution networks. *Journal of water resources planning and management*, 123(2):67–77.

Schwenker, F., Kestler, H. A., e Palm, G. (2001). Three learning phases for radial-basis-function networks. *Neural networks*, 14(4):439–458.

Schwenker, F., Kestler, H. A., e Palm, G. (2002). Unsupervised and supervised learning in radial-basis-function networks. Em *Self-Organizing neural networks*, pgs. 217–243. Springer.

Serafini, D. B. (1999). *A Framework for Managing Models in Nonlinear Optimization of Computationally Expensive Functions*. PhD thesis, Rice University, Houston, TX, USA. AAI9928598.

Shi, J., Zhang, G., e Sha, J. (2011). Optimal production planning for a multi-product closed loop system with uncertain demand and return. *Computers & Operations Research*, 38(3):641–650.

Shi, L., Yang, R., e Zhu, P. (2012). A method for selecting surrogate models in crashworthiness optimization. *Structural and Multidisciplinary Optimization*, 46(2):159–170.

Shiau, C.-S. N. e Michalek, J. J. (2009). Optimal product design under price competition. *Journal of Mechanical Design*, 131(7):071003.

Smola, A. J. e Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222.

Soliman, F. e Murtagh, B. (1982). The solution of large-scale gas pipeline design problems. *Engineering Optimization*, 6(2):77–83.

St, L., Wold, S., et al. (1989). Analysis of variance (anova). *Chemometrics and intelligent laboratory systems*, 6(4):259–272.

Stein, M. L. (2012). *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media.

Tenne, Y. e Goh, C.-K. (2010). *Computational intelligence in expensive optimization problems*, volume 2. Springer Science & Business Media.

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

Tsirikoglou, P., Ghorbaniasl, G., Abraham, S., e Lacor, C. (2016). Heat transfer optimization of a ribbed surface using surrogate-assisted genetic algoritms. *VII European Congress on Computational Methods in Applied Sciences and Engineering*.

Vapnik, V. (2013). *The nature of statistical learning theory*. Springer science & business media.

Vapnik, V. N. e Vapnik, V. (1998). *Statistical learning theory*, volume 1. Wiley New York.

Walmsley, M., Heiligers, J., Ceriotti, M., e McInnes, C. (2016). Optimal trajectories for planetary pole-sitter missions. *Journal of Guidance, Control, and Dynamics*.

Wang, G. G. e Shan, S. (2007). Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical design*, 129(4):370–380.

Whelan, S. e Goldman, N. (2001). A general empirical model of protein evolution derived from multiple protein families using a maximum-likelihood approach. *Molecular biology and evolution*, 18(5):691–699.

Wortmann, T., Costa, A., Nannicini, G., e Schroepfer, T. (2015). Advantages of surrogate models for architectural design optimization. *AI EDAM*, 29(4):471–481.

Wu, Y., Wang, H., Zhang, B., e Du, K.-L. (2012). Using radial basis function networks for function approximation and classification. *ISRN Applied Mathematics*, 2012.

Yu, W., Liu, T., Valdez, R., Gwinn, M., e Khoury, M. J. (2010). Application of support vector machine modeling for prediction of common diseases: the case of diabetes and pre-diabetes. *BMC Medical Informatics and Decision Making*, 10(1):16.

Zhou, Z., Ong, Y. S., Nair, P. B., Keane, A. J., e Lum, K. Y. (2007). Combining global and local surrogate models to accelerate evolutionary optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(1):66–76.

Zhu, C., Byrd, R. H., Lu, P., e Nocedal, J. (1997). Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, 23(4):550–560.

Zhu, J.-H., Zhang, W.-H., e Xia, L. (2016). Topology optimization in aircraft and aerospace structures design. *Archives of Computational Methods in Engineering*, 23(4):595–622.

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

Zitzler, E., Deb, K., e Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, 8(2):173–195.

Zörnig, P. (2014). *Nonlinear programming*. de Gruyter Textbook. De Gruyter.

# APÊNDICE A – Variáveis de Projeto dos Experimentos Numéricos

As Tabelas apresentadas aqui mostram para cada algoritmo e para cada experimento numérico as variáveis de projeto encontradas na melhor execução.

Tabela 27 – Variáveis de Projeto e Peso da Treliça de 10 Barras - Caso Discreto.

| | AG | AG-SVR$^P$ | AG-SVR$^G$ | AG-RBFN$^P$ | AG-RBFN$^G$ |
|---|---|---|---|---|---|
| $A_1$ | 42.000 | 42.000 | 42.000 | 42.000 | 41.000 |
| $A_2$ | 1.000 | 1.000 | 43.000 | 43.000 | 1.000 |
| $A_3$ | 39.000 | 39.000 | 39.000 | 39.000 | 39.000 |
| $A_4$ | 32.000 | 32.000 | 33.000 | 33.000 | 35.000 |
| $A_5$ | 43.000 | 43.000 | 43.000 | 1.000 | 43.000 |
| $A_6$ | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| $A_7$ | 28.000 | 28.000 | 28.000 | 28.000 | 28.000 |
| $A_8$ | 39.000 | 39.000 | 38.000 | 38.000 | 39.000 |
| $A_9$ | 38.000 | 38.000 | 38.000 | 38.000 | 39.000 |
| $A_{10}$ | 43.000 | 1.000 | 1.000 | 1.000 | 43.000 |
| Peso | **5490.7378** | **5490.7378** | 5491.7173 | 5491.7173 | 5507.7584 |

Tabela 28 – Variáveis de Projeto e Peso da Treliça de 10 Barras - Caso Contínuo.

| | AG | AG-SVR$^P$ | AG-SVR$^G$ | AG-RBFN$^P$ | AG-RBFN$^G$ |
|---|---|---|---|---|---|
| $A_1$ | 30.613 | 30.162 | 30.962 | 29.939 | 30.863 |
| $A_2$ | 0.110 | 0.113 | 0.109 | 0.114 | 0.105 |
| $A_3$ | 23.209 | 23.540 | 22.899 | 23.403 | 22.547 |
| $A_4$ | 14.785 | 15.455 | 15.716 | 15.217 | 15.226 |
| $A_5$ | 0.103 | 0.101 | 0.106 | 0.101 | 0.101 |
| $A_6$ | 0.675 | 0.658 | 0.491 | 0.752 | 0.615 |
| $A_7$ | 7.461 | 7.467 | 7.485 | 7.421 | 7.573 |
| $A_8$ | 20.561 | 21.161 | 20.928 | 21.058 | 20.950 |
| $A_9$ | 22.331 | 21.251 | 21.325 | 21.778 | 21.790 |
| $A_{10}$ | 0.105 | 0.102 | 0.100 | 0.105 | 0.105 |
| Peso | 5070.6875 | **5065.7067** | 5067.5394 | 5066.8791 | 5067.2723 |

Tabela 29 – Variáveis de Projeto e Peso da Treliça de 25 Barras - Caso Discreto.

| | AG | AG-SVR$^P$ | AG-SVR$^G$ | AG-RBFN$^P$ | AG-RBFN$^G$ |
|---|---|---|---|---|---|
| $A_1$ | 1.000 | 1.000 | 1.000 | 2.000 | 1.000 |
| $A_2$ | 4.000 | 4.000 | 4.000 | 6.000 | 6.000 |
| $A_3$ | 31.000 | 32.000 | 31.000 | 32.000 | 32.000 |
| $A_4$ | 1.000 | 1.000 | 1.000 | 2.000 | 1.000 |
| $A_5$ | 22.000 | 22.000 | 22.000 | 20.000 | 20.000 |
| $A_6$ | 11.000 | 11.000 | 11.000 | 10.000 | 10.000 |
| $A_7$ | 6.000 | 6.000 | 6.000 | 6.000 | 6.000 |
| $A_8$ | 32.000 | 32.000 | 32.000 | 31.000 | 31.000 |
| Peso | **484.8541** | **484.8541** | **484.8541** | 485.0487 | 485.0487 |

Tabela 30 – Variáveis de Projeto e Peso da Treliça de 25 Barras - Caso Contínuo.

| | AG | AG-SVR$^P$ | AG-SVR$^G$ | AG-RBFN$^P$ | AG-RBFN$^G$ |
|---|---|---|---|---|---|
| $A_1$ | 0.103 | 0.120 | 0.107 | 0.101 | 0.101 |
| $A_2$ | 0.531 | 0.378 | 0.363 | 0.497 | 0.538 |
| $A_3$ | 3.397 | 3.394 | 3.399 | 3.394 | 3.400 |
| $A_4$ | 0.101 | 0.100 | 0.102 | 0.102 | 0.101 |
| $A_5$ | 1.873 | 1.912 | 1.993 | 1.938 | 1.821 |
| $A_6$ | 0.938 | 0.983 | 0.967 | 0.935 | 0.966 |
| $A_7$ | 0.439 | 0.495 | 0.502 | 0.453 | 0.414 |
| $A_8$ | 3.398 | 3.400 | 3.399 | 3.399 | 3.400 |
| Peso | 484.4022 | 484.4464 | **484.2616** | 484.3963 | 484.3867 |

Tabela 31 – Variáveis de Projeto e Peso da Treliça de 60 Barras - Caso Discreto.

| | AG | AG-SVR$^P$ | AG-SVR$^G$ | AG-RBFN$^P$ | AG-RBFN$^G$ |
|---|---|---|---|---|---|
| $A_1$ | 8.000 | 9.000 | 8.000 | 8.000 | 9.000 |
| $A_2$ | 18.000 | 16.000 | 17.000 | 19.000 | 18.000 |
| $A_3$ | 2.000 | 3.000 | 1.000 | 1.000 | 1.000 |
| $A_4$ | 12.000 | 14.000 | 12.000 | 12.000 | 12.000 |
| $A_5$ | 12.000 | 12.000 | 11.000 | 13.000 | 11.000 |
| $A_6$ | 2.000 | 1.000 | 2.000 | 2.000 | 3.000 |
| $A_7$ | 16.000 | 16.000 | 16.000 | 17.000 | 17.000 |
| $A_8$ | 15.000 | 15.000 | 15.000 | 17.000 | 17.000 |
| $A_9$ | 6.000 | 8.000 | 6.000 | 7.000 | 7.000 |
| $A_{10}$ | 13.000 | 13.000 | 13.000 | 13.000 | 12.000 |
| $A_{11}$ | 14.000 | 15.000 | 16.000 | 11.000 | 13.000 |
| $A_{12}$ | 1.000 | 2.000 | 1.000 | 1.000 | 1.000 |
| $A_{13}$ | 18.000 | 16.000 | 18.000 | 18.000 | 18.000 |
| $A_{14}$ | 10.000 | 9.000 | 9.000 | 9.000 | 9.000 |
| $A_{15}$ | 8.000 | 6.000 | 7.000 | 8.000 | 7.000 |
| $A_{16}$ | 3.000 | 3.000 | 2.000 | 3.000 | 2.000 |
| $A_{17}$ | 2.000 | 3.000 | 3.000 | 2.000 | 3.000 |
| $A_{18}$ | 7.000 | 8.000 | 9.000 | 6.000 | 6.000 |
| $A_{19}$ | 8.000 | 8.000 | 8.000 | 8.000 | 9.000 |
| $A_{20}$ | 8.000 | 8.000 | 8.000 | 9.000 | 8.000 |
| $A_{21}$ | 8.000 | 7.000 | 7.000 | 7.000 | 7.000 |
| $A_{22}$ | 7.000 | 7.000 | 7.000 | 7.000 | 7.000 |
| $A_{23}$ | 2.000 | 2.000 | 2.000 | 2.000 | 3.000 |
| $A_{24}$ | 7.000 | 7.000 | 8.000 | 7.000 | 8.000 |
| $A_{25}$ | 9.000 | 9.000 | 9.000 | 10.000 | 9.000 |
| Peso | 315.7745 | 316.9369 | **314.7910** | 316.7392 | 316.9557 |

Tabela 32 – Variáveis de Projeto e Peso da Treliça de 60 Barras - Caso Contínuo.

| | AG | AG-SVR$^P$ | AG-SVR$^G$ | AG-RBFN$^P$ | AG-RBFN$^G$ |
|---|---|---|---|---|---|
| $A_1$ | 1.230 | 1.197 | 1.176 | 1.236 | 1.253 |
| $A_2$ | 2.084 | 2.088 | 2.074 | 2.204 | 2.259 |
| $A_3$ | 0.504 | 0.509 | 0.517 | 0.506 | 0.522 |
| $A_4$ | 1.644 | 1.833 | 1.610 | 1.729 | 1.489 |
| $A_5$ | 1.688 | 1.615 | 1.556 | 1.499 | 1.420 |
| $A_6$ | 0.563 | 0.574 | 0.524 | 0.545 | 0.557 |
| $A_7$ | 2.084 | 1.934 | 1.940 | 1.995 | 2.062 |
| $A_8$ | 1.997 | 1.873 | 1.853 | 1.989 | 1.975 |
| $A_9$ | 1.094 | 1.103 | 1.071 | 1.109 | 1.163 |
| $A_{10}$ | 1.413 | 1.526 | 1.820 | 1.701 | 1.845 |
| $A_{11}$ | 1.673 | 1.809 | 1.949 | 1.592 | 1.962 |
| $A_{12}$ | 0.576 | 0.506 | 0.535 | 0.507 | 0.527 |
| $A_{13}$ | 2.059 | 2.140 | 2.178 | 2.104 | 2.075 |
| $A_{14}$ | 1.381 | 1.569 | 1.393 | 1.358 | 1.272 |
| $A_{15}$ | 1.278 | 1.225 | 1.048 | 1.102 | 1.058 |
| $A_{16}$ | 0.712 | 0.709 | 0.652 | 0.747 | 0.730 |
| $A_{17}$ | 0.932 | 0.793 | 0.617 | 0.592 | 0.648 |
| $A_{18}$ | 0.996 | 0.991 | 1.330 | 1.232 | 1.072 |
| $A_{19}$ | 1.174 | 1.220 | 1.174 | 1.176 | 1.150 |
| $A_{20}$ | 1.179 | 1.263 | 1.216 | 1.411 | 1.232 |
| $A_{21}$ | 1.018 | 1.029 | 1.166 | 1.087 | 1.233 |
| $A_{22}$ | 1.096 | 1.125 | 1.269 | 1.136 | 1.125 |
| $A_{23}$ | 0.794 | 0.607 | 0.544 | 0.633 | 0.517 |
| $A_{24}$ | 1.211 | 1.112 | 1.092 | 1.191 | 1.202 |
| $A_{25}$ | 1.306 | 1.286 | 1.330 | 1.322 | 1.352 |
| Peso | 316.7800 | 316.1664 | **316.0679** | 316.8969 | 316.8612 |

Tabela 33 – Variáveis de Projeto e Peso da Treliça de 72 Barras - Caso Discreto.

| | AG | AG-SVR$^P$ | AG-SVR$^G$ | AG-RBFN$^P$ | AG-RBFN$^G$ |
|---|---|---|---|---|---|
| $A_1$ | 2.000 | 2.000 | 2.000 | 2.000 | 2.000 |
| $A_2$ | 6.000 | 6.000 | 6.000 | 6.000 | 6.000 |
| $A_3$ | 4.000 | 4.000 | 4.000 | 4.000 | 4.000 |
| $A_4$ | 6.000 | 6.000 | 6.000 | 6.000 | 6.000 |
| $A_5$ | 5.000 | 5.000 | 5.000 | 5.000 | 5.000 |
| $A_6$ | 5.000 | 5.000 | 5.000 | 5.000 | 5.000 |
| $A_7$ | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| $A_8$ | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| $A_9$ | 14.000 | 13.000 | 13.000 | 14.000 | 14.000 |
| $A_{10}$ | 5.000 | 5.000 | 5.000 | 5.000 | 5.000 |
| $A_{11}$ | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| $A_{12}$ | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| $A_{13}$ | 19.000 | 20.000 | 20.000 | 19.000 | 19.000 |
| $A_{14}$ | 5.000 | 5.000 | 5.000 | 5.000 | 5.000 |
| $A_{15}$ | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| $A_{16}$ | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| Peso | **385.5426** | **385.5426** | **385.5426** | **385.5426** | **385.5426** |

Tabela 34 – Variáveis de Projeto e Peso da Treliça de 72 Barras - Caso Contínuo.

| | AG | AG-SVR$^P$ | AG-SVR$^G$ | AG-RBFN$^P$ | AG-RBFN$^G$ |
|---|---|---|---|---|---|
| $A_1$ | 0.161 | 0.155 | 0.158 | 0.162 | 0.155 |
| $A_2$ | 0.547 | 0.564 | 0.553 | 0.507 | 0.533 |
| $A_3$ | 0.385 | 0.398 | 0.405 | 0.407 | 0.388 |
| $A_4$ | 0.564 | 0.575 | 0.511 | 0.610 | 0.563 |
| $A_5$ | 0.570 | 0.559 | 0.633 | 0.482 | 0.578 |
| $A_6$ | 0.495 | 0.517 | 0.510 | 0.528 | 0.518 |
| $A_7$ | 0.100 | 0.103 | 0.102 | 0.110 | 0.119 |
| $A_8$ | 0.125 | 0.115 | 0.104 | 0.113 | 0.113 |
| $A_9$ | 1.329 | 1.304 | 1.219 | 1.313 | 1.234 |
| $A_{10}$ | 0.536 | 0.501 | 0.501 | 0.489 | 0.511 |
| $A_{11}$ | 0.105 | 0.102 | 0.103 | 0.100 | 0.102 |
| $A_{12}$ | 0.109 | 0.103 | 0.102 | 0.101 | 0.106 |
| $A_{13}$ | 1.889 | 1.823 | 2.038 | 2.005 | 1.907 |
| $A_{14}$ | 0.502 | 0.506 | 0.508 | 0.534 | 0.526 |
| $A_{15}$ | 0.103 | 0.104 | 0.103 | 0.100 | 0.100 |
| $A_{16}$ | 0.105 | 0.102 | 0.104 | 0.100 | 0.106 |
| Peso | 381.9382 | **380.6095** | 381.5848 | 381.7313 | 381.3455 |

Tabela 35 – Variáveis de Projeto e Peso da Treliça de 942 Barras - Caso Discreto.

| | AG | AG-SVR$^P$ | AG-SVR$^G$ | AG-RBFN$^P$ | AG-RBFN$^G$ |
|---|---|---|---|---|---|
| $A_1$ | 16.000 | 10.000 | 16.000 | 1.000 | 42.000 |
| $A_2$ | 3.000 | 2.000 | 6.000 | 4.000 | 20.000 |
| $A_3$ | 3.000 | 4.000 | 9.000 | 24.000 | 2.000 |
| $A_4$ | 8.000 | 3.000 | 2.000 | 3.000 | 2.000 |
| $A_5$ | 3.000 | 2.000 | 1.000 | 3.000 | 15.000 |
| $A_6$ | 22.000 | 38.000 | 17.000 | 29.000 | 13.000 |
| $A_7$ | 3.000 | 4.000 | 3.000 | 9.000 | 3.000 |
| $A_8$ | 30.000 | 17.000 | 9.000 | 19.000 | 4.000 |
| $A_9$ | 21.000 | 13.000 | 29.000 | 9.000 | 54.000 |
| $A_{10}$ | 2.000 | 4.000 | 1.000 | 5.000 | 2.000 |
| $A_{11}$ | 5.000 | 8.000 | 30.000 | 10.000 | 14.000 |
| $A_{12}$ | 3.000 | 6.000 | 8.000 | 4.000 | 4.000 |
| $A_{13}$ | 23.000 | 22.000 | 26.000 | 20.000 | 16.000 |
| $A_{14}$ | 3.000 | 4.000 | 3.000 | 2.000 | 3.000 |
| $A_{15}$ | 46.000 | 2.000 | 18.000 | 11.000 | 18.000 |
| $A_{16}$ | 4.000 | 3.000 | 2.000 | 9.000 | 4.000 |
| $A_{17}$ | 17.000 | 23.000 | 29.000 | 19.000 | 9.000 |
| $A_{18}$ | 4.000 | 2.000 | 3.000 | 4.000 | 4.000 |
| $A_{19}$ | 8.000 | 64.000 | 10.000 | 17.000 | 15.000 |
| $A_{20}$ | 1.000 | 2.000 | 2.000 | 2.000 | 1.000 |
| $A_{21}$ | 43.000 | 43.000 | 41.000 | 30.000 | 29.000 |
| $A_{22}$ | 3.000 | 6.000 | 3.000 | 3.000 | 3.000 |
| $A_{23}$ | 17.000 | 15.000 | 15.000 | 43.000 | 11.000 |
| $A_{24}$ | 31.000 | 28.000 | 40.000 | 27.000 | 29.000 |
| $A_{25}$ | 42.000 | 39.000 | 33.000 | 47.000 | 71.000 |
| $A_{26}$ | 18.000 | 14.000 | 6.000 | 8.000 | 9.000 |
| $A_{27}$ | 5.000 | 9.000 | 17.000 | 4.000 | 7.000 |
| $A_{28}$ | 4.000 | 18.000 | 8.000 | 15.000 | 8.000 |
| $A_{29}$ | 12.000 | 8.000 | 21.000 | 21.000 | 17.000 |
| $A_{30}$ | 27.000 | 25.000 | 32.000 | 7.000 | 32.000 |
| $A_{31}$ | 43.000 | 43.000 | 34.000 | 33.000 | 66.000 |
| $A_{32}$ | 6.000 | 3.000 | 3.000 | 3.000 | 5.000 |
| $A_{33}$ | 6.000 | 8.000 | 7.000 | 3.000 | 5.000 |
| $A_{34}$ | 6.000 | 1.000 | 4.000 | 3.000 | 5.000 |
| $A_{35}$ | 2.000 | 3.000 | 3.000 | 2.000 | 2.000 |
| $A_{36}$ | 6.000 | 8.000 | 3.000 | 13.000 | 6.000 |
| $A_{37}$ | 52.000 | 51.000 | 57.000 | 54.000 | 55.000 |
| $A_{38}$ | 3.000 | 4.000 | 5.000 | 6.000 | 3.000 |
| $A_{39}$ | 7.000 | 2.000 | 2.000 | 1.000 | 3.000 |
| $A_{40}$ | 4.000 | 4.000 | 4.000 | 8.000 | 9.000 |
| $A_{41}$ | 3.000 | 7.000 | 5.000 | 5.000 | 2.000 |
| $A_{42}$ | 3.000 | 6.000 | 4.000 | 17.000 | 2.000 |
| $A_{43}$ | 76.000 | 74.000 | 60.000 | 64.000 | 88.000 |
| $A_{44}$ | 4.000 | 5.000 | 5.000 | 4.000 | 3.000 |
| $A_{45}$ | 2.000 | 1.000 | 4.000 | 1.000 | 1.000 |
| $A_{46}$ | 7.000 | 6.000 | 4.000 | 5.000 | 12.000 |
| $A_{47}$ | 2.000 | 4.000 | 4.000 | 1.000 | 4.000 |
| $A_{48}$ | 8.000 | 3.000 | 5.000 | 3.000 | 3.000 |
| $A_{49}$ | 84.000 | 93.000 | 86.000 | 90.000 | 85.000 |
| $A_{50}$ | 9.000 | 5.000 | 4.000 | 5.000 | 6.000 |
| $A_{51}$ | 6.000 | 3.000 | 15.000 | 3.000 | 8.000 |
| $A_{52}$ | 10.000 | 6.000 | 3.000 | 9.000 | 8.000 |
| $A_{53}$ | 16.000 | 8.000 | 44.000 | 16.000 | 24.000 |
| $A_{54}$ | 23.000 | 22.000 | 9.000 | 13.000 | 17.000 |
| $A_{55}$ | 59.000 | 59.000 | 61.000 | 74.000 | 57.000 |
| $A_{56}$ | 10.000 | 11.000 | 5.000 | 2.000 | 7.000 |
| $A_{57}$ | 47.000 | 40.000 | 37.000 | 44.000 | 37.000 |
| $A_{58}$ | 4.000 | 9.000 | 11.000 | 2.000 | 17.000 |
| $A_{59}$ | 6.000 | 23.000 | 9.000 | 29.000 | 10.000 |
| Peso | 177937.24 | 177521.36 | **172748.11** | 177568.19 | 180342.78 |

Tabela 36 – Variáveis de Projeto e Peso da Treliça de 942 Barras - Caso Contínuo.

| | AG | AG-SVR$^P$ | AG-SVR$^G$ | AG-RBFN$^P$ | AG-RBFN$^G$ |
|---|---|---|---|---|---|
| $A_1$ | 79.249 | 12.728 | 14.469 | 28.547 | 17.295 |
| $A_2$ | 5.799 | 5.827 | 22.194 | 1.332 | 9.658 |
| $A_3$ | 12.131 | 12.287 | 13.171 | 23.860 | 5.467 |
| $A_4$ | 6.149 | 4.047 | 3.704 | 5.767 | 3.149 |
| $A_5$ | 2.170 | 3.388 | 2.826 | 5.719 | 3.740 |
| $A_6$ | 13.463 | 14.791 | 21.054 | 8.863 | 14.847 |
| $A_7$ | 6.679 | 2.685 | 3.650 | 2.424 | 1.756 |
| $A_8$ | 17.272 | 6.971 | 11.353 | 29.948 | 80.722 |
| $A_9$ | 38.711 | 51.962 | 27.017 | 53.186 | 38.461 |
| $A_{10}$ | 3.597 | 5.989 | 2.572 | 8.639 | 6.865 |
| $A_{11}$ | 5.190 | 5.879 | 9.137 | 11.671 | 9.563 |
| $A_{12}$ | 15.439 | 17.398 | 8.233 | 3.202 | 3.924 |
| $A_{13}$ | 36.797 | 15.674 | 14.516 | 14.061 | 25.570 |
| $A_{14}$ | 5.224 | 3.596 | 4.274 | 3.015 | 1.421 |
| $A_{15}$ | 6.068 | 4.309 | 10.981 | 3.728 | 11.911 |
| $A_{16}$ | 2.372 | 10.099 | 1.618 | 3.357 | 8.756 |
| $A_{17}$ | 39.050 | 25.705 | 13.637 | 17.439 | 27.254 |
| $A_{18}$ | 4.708 | 2.059 | 2.213 | 1.950 | 3.338 |
| $A_{19}$ | 8.942 | 7.477 | 33.965 | 6.285 | 12.764 |
| $A_{20}$ | 1.807 | 3.192 | 1.196 | 4.868 | 3.997 |
| $A_{21}$ | 26.889 | 23.144 | 25.141 | 25.632 | 42.384 |
| $A_{22}$ | 4.262 | 3.251 | 2.338 | 6.196 | 3.642 |
| $A_{23}$ | 23.564 | 18.028 | 41.098 | 22.607 | 21.127 |
| $A_{24}$ | 15.171 | 28.747 | 23.150 | 16.158 | 17.349 |
| $A_{25}$ | 52.448 | 49.845 | 42.918 | 30.345 | 22.397 |
| $A_{26}$ | 6.556 | 5.705 | 14.146 | 8.420 | 4.576 |
| $A_{27}$ | 8.992 | 9.070 | 6.106 | 16.563 | 17.501 |
| $A_{28}$ | 26.568 | 35.093 | 22.436 | 13.887 | 20.964 |
| $A_{29}$ | 19.305 | 10.611 | 31.857 | 16.558 | 16.385 |
| $A_{30}$ | 11.857 | 35.555 | 21.104 | 15.437 | 20.706 |
| $A_{31}$ | 56.408 | 51.558 | 43.781 | 49.997 | 40.168 |
| $A_{32}$ | 3.643 | 2.593 | 3.604 | 2.397 | 5.888 |
| $A_{33}$ | 6.220 | 7.041 | 3.943 | 14.598 | 12.468 |
| $A_{34}$ | 3.568 | 3.662 | 3.760 | 1.344 | 3.176 |
| $A_{35}$ | 2.361 | 3.390 | 1.500 | 1.107 | 3.342 |
| $A_{36}$ | 6.523 | 7.103 | 8.215 | 4.958 | 2.671 |
| $A_{37}$ | 44.440 | 47.721 | 54.014 | 65.395 | 46.981 |
| $A_{38}$ | 4.673 | 3.582 | 4.814 | 3.352 | 2.540 |
| $A_{39}$ | 2.759 | 2.795 | 2.725 | 3.941 | 3.921 |
| $A_{40}$ | 4.855 | 4.576 | 1.791 | 9.789 | 6.313 |
| $A_{41}$ | 6.230 | 5.368 | 2.196 | 5.283 | 1.718 |
| $A_{42}$ | 8.712 | 5.292 | 2.377 | 9.138 | 14.823 |
| $A_{43}$ | 53.676 | 65.788 | 61.791 | 77.051 | 80.706 |
| $A_{44}$ | 5.454 | 4.228 | 3.182 | 5.411 | 5.172 |
| $A_{45}$ | 1.241 | 2.390 | 1.878 | 5.005 | 5.902 |
| $A_{46}$ | 4.915 | 2.939 | 5.483 | 3.266 | 7.700 |
| $A_{47}$ | 4.964 | 1.568 | 2.284 | 2.219 | 1.145 |
| $A_{48}$ | 5.312 | 7.551 | 7.442 | 9.041 | 6.511 |
| $A_{49}$ | 70.078 | 88.475 | 92.787 | 106.271 | 84.449 |
| $A_{50}$ | 5.400 | 6.589 | 6.789 | 5.042 | 3.927 |
| $A_{51}$ | 3.563 | 7.685 | 4.081 | 7.731 | 8.547 |
| $A_{52}$ | 6.287 | 5.940 | 3.436 | 3.209 | 5.013 |
| $A_{53}$ | 17.755 | 26.135 | 30.026 | 40.197 | 30.504 |
| $A_{54}$ | 22.837 | 29.901 | 15.102 | 25.852 | 27.576 |
| $A_{55}$ | 39.193 | 72.874 | 49.216 | 71.162 | 42.694 |
| $A_{56}$ | 8.172 | 1.551 | 5.600 | 1.193 | 2.594 |
| $A_{57}$ | 53.827 | 44.143 | 46.749 | 33.960 | 55.860 |
| $A_{58}$ | 7.610 | 17.770 | 7.432 | 26.039 | 10.723 |
| $A_{59}$ | 7.897 | 5.531 | 10.164 | 3.212 | 12.608 |
| Peso | 177432.66 | 173658.86 | **172670.57** | 181064.70 | 181431.17 |

Tabela 37 – Variáveis de Projeto e Diferença de Temperatura do Trocador de Calor.

| | AG | AG-SVR$^P$ | AG-SVR$^G$ | AG-RBFN$^P$ | AG-RBFN$^G$ |
|---|---|---|---|---|---|
| $x_1$ | 21.888 | 24.720 | 15.510 | 15.395 | 28.514 |
| $y_1$ | 46.760 | 51.054 | 56.929 | 24.785 | 64.258 |
| $x_2$ | 16.755 | 17.408 | 24.212 | 16.940 | 18.213 |
| $y_2$ | 62.914 | 63.112 | 64.567 | 62.770 | 46.216 |
| $x_3$ | 21.621 | 27.174 | 18.110 | 16.589 | 33.892 |
| $y_3$ | 16.861 | 14.991 | 14.958 | 15.990 | 18.383 |
| $x_4$ | 16.079 | 19.931 | 20.764 | 19.944 | 16.178 |
| $y_4$ | 41.702 | 45.648 | 45.895 | 43.370 | 44.491 |
| $n$ | 4 | 4 | 4 | 4 | 4 |
| $\Delta T$ | 28.3581 | 28.6291 | 29.2734 | **29.5992** | 28.7944 |