



UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO DE ENGENHARIA CIVIL
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA CIVIL

Gledson Carlos de Oliveira Assis

Metodologia baseada em metamodelos para a otimização global de problemas *black-box* com restrições computacionalmente onerosas.

Florianópolis
2021

Gledson Carlos de Oliveira Assis

Metodologia baseada em metamodelos para a otimização global de problemas *black-box* com restrições computacionalmente onerosas.

Dissertação do Programa de Pós-Graduação em Engenharia Civil do Centro de Engenharia Civil da Universidade Federal de Santa Catarina para a obtenção do título de Mestre em Engenharia Civil.
Orientador: Prof. Rafael Holdorf Lopez, Dr.

Florianópolis
2021

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Assis, Gledson Carlos de Oliveira
Metodologia baseada em metamodelos para a otimização
global de problemas black-box com restrições
computacionalmente onerosas. / Gledson Carlos de Oliveira
Assis ; orientador, Rafael Holdorf Lopez, 2021.
114 p.

Dissertação (mestrado) - Universidade Federal de Santa
Catarina, Centro Tecnológico, Programa de Pós-Graduação em
Engenharia Civil, Florianópolis, 2021.

Inclui referências.

1. Engenharia Civil. 2. Otimização global. 3. Restrições
onerosas. 4. Kriging. I. Lopez, Rafael Holdorf. II.
Universidade Federal de Santa Catarina. Programa de Pós
Graduação em Engenharia Civil. III. Título.

Gledson Carlos de Oliveira Assis

Metodologia baseada em metamodelos para a otimização global de problemas *black-box* com restrições computacionalmente onerosas.

O presente trabalho em nível de mestrado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof. Henrique Machado Kroetz, Dr.
Universidade Federal do Paraná

Prof. Wellison José de Santana Gomes, Dr.
Universidade Federal de Santa Catarina

Olavo Mecias da Silva Junior, Dr.
Universidade Federal de Santa Catarina

Certificamos que esta é a **versão original** e **final** do trabalho de conclusão que foi julgado adequado para obtenção do título de mestre em Engenharia Civil.

Coordenação do Curso

Prof. Rafael Holdorf Lopez, Dr.
Orientador

Florianópolis, 11 de Fevereiro de 2021.

Este trabalho é dedicado aos meus colegas de classe, aos meus queridos pais e a todos os meus amigos que acreditaram que eu poderia chegar até aqui.

AGRADECIMENTOS

Agradeço primeiramente à Deus, pela dom da vida. Por permitir-me errar, aprender, crescer, evoluir sem me deixar perder a essência de quem sou. Por sua eterna paciência, compreensão e tolerância. Por seu infinito amor. Pela sua proteção a todo momento que estive longe de casa, da minha família e de todos que amo.

Agradeço à minha esposa, Ana Paula, que partilhou de todos os momentos comigo, sejam eles fáceis ou difíceis. Seu apoio e atenção foram vitais não só para os momentos que estive só em uma cidade distante, como também para a conclusão dessa etapa da minha vida acadêmica. Agradeço pelas ligações demoradas quando me sentia só e pelo carinho e amor que sempre esteve aqui me esperando quando eu voltava para os seus braços.

Agradeço aos meus pais, Francisco e Meire. A criação que me deram, as oportunidades que me forneceram e o apoio independente da situação me fizeram chegar até aqui, por isso, muito obrigado.

Ao Professor Rafael Holdorf Lopez o meu muito obrigado por todos os ensinamentos, orientações e confiança, que, de uma forma excepcional me guiou não só na realização deste trabalho, mas também em vários outros projetos ao longo dessa caminhada.

Aos amigos de laboratório, em especial Vinícius Favareto, Victor Belafonte, Felipe Carraro, Matheus Gonçalves e Caroline Dapieve agradeço pela paciência, os momentos de descontração, as discussões prolongadas sobre os diversos temas de estudo, e a forma como me acolheram em um momento que eu não conhecia nada nem ninguém da cidade.

Ao CNPq pela concessão de bolsa de pesquisa para conclusão deste mestrado.

Por fim, agradeço aos professores, amigos e demais colegas de curso, de alguma forma, me influenciaram positivamente, apoiaram e torceram pela conclusão desse trabalho.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001

*“É muito melhor lançar-se em busca
de conquistas grandiosas, mesmo expondo-se
ao fracasso, do que alinhar-se com os pobres de espírito,
que nem gozam muito nem sofrem muito,
porque vivem numa penumbra cinzenta,
onde não conhecem nem vitória, nem derrota.”*
(Theodore Roosevelt)

RESUMO

A otimização consiste em escolher a melhor solução possível para um dado problema. Através do emprego de técnicas para a seleção das melhores alternativas, o processo de otimizar pode ser usado para definir, por exemplo, o melhor trajeto de casa para o trabalho ou determinar as menores seções transversais admissíveis de elementos que formam a estrutura de uma torre de transmissão. Na atualidade, as metodologias de otimização buscam representar mais fielmente os problemas reais, tornando os modelos mais complexos e, conseqüentemente, provocando uma maior exigência de tempo de processamento. Em virtude disso, técnicas que otimizem problemas de elevado custo computacional de análise vem sendo estudadas. Neste trabalho é apresentada uma nova metodologia para otimização de problemas com avaliações computacionalmente onerosas. Esse novo algoritmo trata de problemas que possuem função objetivo com baixo custo computacional e funções de restrição de cálculo onerosas na qual a forma analítica não é conhecida, também conhecidas como funções *black-box*. A metodologia utiliza-se de modelos aproximados gerados pelo método de *Kriging* (KRIGE, 1951) para representar as funções de restrição computacionalmente caras e foi inspirada no algoritmo GOSAC (MÜLLER; WOODBURY, 2017). O método foi capaz de determinar, de forma satisfatória, o valor mínimo em problemas clássicos de otimização e em problemas de otimização estrutural. Com uma média de avaliações das funções de restrição inferior a 300 execuções, cada problema foi otimizado diversas vezes com o objetivo de se expor um modelo estatístico dos resultados obtidos com a aplicação do método. Os resultados foram comparados a valores obtidos pelo GOSAC e aos encontrados na literatura, comprovando a eficiência do método proposto.

Palavras-chave: Otimização global. Restrições onerosas. *Kriging*.

ABSTRACT

Optimization consists of choosing the best possible solution for a given problem. Through the use of techniques for the selection of the best alternatives, the optimization process can be used to define, for example, the best commute from home to work or to determine the smallest permissible cross sections of elements that form the structure of a tower. Nowadays, optimization methodologies seek to more accurately represent real problems, making models more complex and, consequently, causing a greater demand for processing time. As a result, techniques that optimize problems of high computational cost of analysis have been studied. This work presents a new methodology for optimization of problems with computationally costly evaluations. This new algorithm deals with problems that have an objective function with low computational cost and costly calculation restriction functions in which the analytical form is not known, also known as black-box functions. The methodology uses approximate models generated by the Kriging (KRIGE, 1951) method to represent computationally expensive constraint functions and was inspired by the GOSAC (MÜLLER; WOODBURY, 2017) algorithm. The method was able to satisfactorily determine the minimum value in classical optimization problems and in structural optimization problems. With an average of evaluations of the restriction functions of less than 300 executions, each problem was optimized several times in order to expose a statistical model of the results obtained with the application of the method. The results were compared to values obtained by GOSAC and those found in the literature, proving the efficiency of the proposed method.

Keywords: Global optimization. Expensive restrictions. Kriging.

LISTA DE FIGURAS

Figura 1 – Órtese <i>fix-it</i> modelada em elementos finitos.	22
Figura 2 – Exemplo de metamodelo aproximado por <i>Kriging</i>	23
Figura 3 – Exemplo de um problema de otimização	25
Figura 4 – Exemplo de um problema de otimização com restrições	27
Figura 5 – Classificação dos problemas de otimização.	28
Figura 6 – Exemplo de um problema multimodal	29
Figura 7 – Hipercubo Latino 3D.	38
Figura 8 – Correlação com variação de \mathbf{p}	41
Figura 9 – Correlação com variação de θ	42
Figura 10 – Aproximação do modelo de <i>kriging</i> com refinamento através da inserção de pontos.	45
Figura 11 – Interpretação gráfica da espereça de melhoria.	47
Figura 12 – Fluxograma Estrutura principal.	56
Figura 13 – Função objetivo e mínimo local - <i>Toy Problem</i>	62
Figura 14 – Função objetivo e pontos iniciais.	63
Figura 15 – Exemplo unidimensional com duas restrições.	65
Figura 16 – Exemplo de uma região de busca.	66
Figura 17 – Iteração 1 - Representação com os diferentes modelos de restrição . . .	68
Figura 18 – Iteração 2 - Representação com os diferentes modelos de restrição . . .	69
Figura 19 – Iterações da etapa 2 para o <i>Toy Problem</i>	71
Figura 20 – Iteração 7 - Otimização local	72
Figura 21 – Iterações da etapa 3 para o <i>Toy Problem</i>	73
Figura 22 – Função objetivo e mínimo local - <i>Toy Problem</i> Modificado.	74
Figura 23 – Iteração 1 - <i>Toy Problem</i> Modificado.	77
Figura 24 – Iteração 2 - <i>Toy Problem</i> Modificado.	78
Figura 25 – Iteração 38 - <i>Toy Problem</i> Modificado.	78
Figura 26 – Exemplo gráfico violino	81
Figura 27 – Distribuição dos resultados obtidos para o <i>Toy Problem</i>	84
Figura 28 – Problema com restrição <i>Eggholder</i> modificada.	85
Figura 29 – Distribuição dos resultados obtidos para o problema com restrição <i>Eggholder</i> modificada.	86
Figura 30 – Problema com restrição <i>Rastrigin</i> modificada.	87
Figura 31 – Distribuição dos resultados obtidos para o problema com restrição <i>Ras-</i> <i>trigin</i> modificada.	88
Figura 32 – Problema com restrição <i>Cross-in-Tray</i> modificada.	89
Figura 33 – Distribuição dos resultados obtidos para o problema com restrição <i>Cross-in-Tray</i> modificada.	90

Figura 34 – Distribuição dos resultados obtidos para o problema 12 do <i>paper GOSAC</i>	91
Figura 35 – Distribuição dos resultados obtidos para o <i>Toy Problem Modificado</i>	93
Figura 36 – Representação mola sob tração/compressão.	94
Figura 37 – Representação Viga soldada.	98
Figura 38 – Representação Torre 25 barras.	100

LISTA DE TABELAS

Tabela 1 – Pontos de referência para o exemplo <i>Toy Problem</i> Modificado.	75
Tabela 2 – Número de pontos inseridos em cada iteração - <i>Toy Problem</i> Modificado.	76
Tabela 3 – Resultados estatísticos do <i>Toy Problem</i>	84
Tabela 4 – Número de análise das restrições do <i>Toy Problem</i>	84
Tabela 5 – Resultados estatísticos para o problema com restrição <i>Eggholder</i> modificada.	86
Tabela 6 – Número de análise das restrições para o problema com restrição <i>Eggholder</i> modificada.	86
Tabela 7 – Resultados estatísticos para o problema com restrição <i>Rastrigin</i> modificada.	87
Tabela 8 – Número de análise das restrições para o problema com restrição <i>Rastrigin</i> modificada.	88
Tabela 9 – Resultados estatísticos para o problema com restrição <i>Cross-in-Tray</i> modificada.	89
Tabela 10 – Número de análise das restrições para o problema com restrição <i>Cross-in-Tray</i> modificada.	89
Tabela 11 – Número realizações convergentes utilizadas na análise estatística para o problema com restrição <i>Cross-in-Tray</i> modificada.	90
Tabela 12 – Resultados estatísticos para o problema 12 <i>paper GOSAC</i>	91
Tabela 13 – Número realizações convergentes utilizadas na análise estatística para o problema 12 do <i>paper GOSAC</i>	91
Tabela 14 – Número de análise das restrições para o problema 12 do <i>paper GOSAC</i> .	92
Tabela 15 – Resultados estatísticos para o <i>Toy Problem</i> Modificado.	92
Tabela 16 – Número de análise das restrições para o <i>Toy Problem</i> Modificado.	93
Tabela 17 – Resultados estatísticos para o problema de Mola sob tração/compressão.	95
Tabela 18 – Comparação dos resultados do problema de Mola sob tração/compressão com diferentes autores.	96
Tabela 19 – Resultados estatísticos para o problema de Viga soldada.	98
Tabela 20 – Comparação dos resultados do problema de Viga soldada com diferentes metodologias.	98
Tabela 21 – Grupos de elementos para a Treliça de 25 barras.	101
Tabela 22 – Esforços atuantes na Treliça de 25 barras (em <i>kips</i>	101
Tabela 23 – Resultados estatísticos para o problema Treliça de 25 barras - Determinística e contínua.	101
Tabela 24 – Comparação dos resultados do problema Treliça de 25 barras - Determinística e contínua.	101

Tabela 25 – Resultados estatísticos para o problema Treliça de 25 barras - Determinística e variáveis discretas.	102
Tabela 26 – Comparação dos resultados do problema Treliça de 25 barras - Determinística e variáveis discretas.	102

SUMÁRIO

1	INTRODUÇÃO	21
1.1	OBJETIVOS	23
1.2	OBJETIVO GERAL	24
1.3	OBJETIVOS ESPECÍFICOS	24
2	OTIMIZAÇÃO	25
2.1	CONCEITOS BÁSICOS	25
2.2	PROBLEMAS DE OTIMIZAÇÃO	27
2.2.1	Convexidade	28
2.2.2	Formulação	30
2.3	ALGORITMOS DE OTIMIZAÇÃO	30
2.3.1	Algoritmos determinísticos	30
2.3.2	Algoritmos Heurísticos	31
2.3.3	Algoritmos de busca local	31
2.3.4	Algoritmos de busca global	32
3	METAMODELOS	35
3.1	PLANO DE AMOSTRAGEM - <i>SAMPLING PLAN</i>	37
3.2	<i>KRIGING</i>	39
3.2.1	Formulação	40
3.2.2	Previsor do <i>Kriging</i>	44
3.3	REFINAMENTO DO METAMODELO	44
3.3.1	<i>Expected Improvement</i> - (EI)	46
4	TRABALHOS SEMELHANTES	49
4.1	CENÁRIO GERAL	49
4.2	ALGORITMO DE OTIMIZAÇÃO GLOBAL COM APROXIMAÇÃO SUBSTITUTIVA DE RESTRIÇÕES - <i>GOSAC</i>	51
5	METODOLOGIA PROPOSTA - ESTRUTURA	55
5.1	ESTRUTURA PRINCIPAL	55
5.2	ETAPA 1	56
5.3	ETAPA 2	57
5.4	ETAPA 3	58
6	METODOLOGIA PROPOSTA - EXEMPLOS COMENTADOS	61
6.1	<i>TOY PROBLEM</i>	61
6.1.1	Processo de otimização	61
6.1.1.1	Etapa 1 - Processo Pré-iterativo	62
6.1.1.2	Etapa 2 - Processo iterativo	63
6.1.1.3	Etapa 3 - Otimização local	70
6.2	<i>TOY PROBLEM</i> MODIFICADO	74

6.2.0.1	Etapa 1 - Processo Pré-iterativo	74
6.2.0.2	Etapa 2 - Processo iterativo	75
7	EXPERIMENTOS NUMÉRICOS	79
7.1	IMPLEMENTAÇÃO	79
7.1.1	Algoritmos utilizados	80
7.1.2	Apresentação dos resultados	80
7.2	PROBLEMAS MATEMÁTICOS	81
7.2.1	Toy problem	83
7.2.2	Problema com restrição por <i>Eggholder</i> Modificada	85
7.2.3	Problema com restrição por <i>Rastrigin</i> Modificada	86
7.2.4	Problema com restrição por <i>Cross-in-Tray</i> Modificada	87
7.2.5	Problema 12 <i>paper</i> GOSAC	89
7.2.6	<i>Toy Problem</i> modificado	92
7.2.7	Discussão dos problemas matemáticos	93
7.3	PROBLEMAS DA ENGENHARIA	94
7.3.1	Mola sob tração/compressão	94
7.3.2	Viga soldada	96
7.3.3	Discussões sobre os problemas clássicos da engenharia	98
7.4	PROBLEMAS DE OTIMIZAÇÃO ESTRUTURAL	99
7.4.1	Treliça de 25 barras - Determinística e contínua	99
7.4.2	Treliça 25 barras - Determinística e discreta	100
7.4.3	Discussões sobre os problemas de otimização estrutural	102
8	CONCLUSÃO E TRABALHOS FUTUROS	103
8.1	CONCLUSÃO	103
8.2	TRABALHOS FUTUROS	103
	REFERÊNCIAS	105

1 INTRODUÇÃO

Problemas de otimização são comumente encontrados em diversos aspectos no dia a dia da sociedade. Seja em algo mais simples como escolher um dentre dois produtos com melhor relação custo-benefício ou em algo mais complexo como determinar a melhor ação para investir parte do seu salário, esse processo, que visa determinar a melhor solução para um dado problema, é empregado mesmo que não seja notado.

Do ponto de vista conceitual, a otimização pode ser definida como um processo de busca por valores mínimos ou máximos de uma determinada função dentro de um domínio comumente restrito. Em geral, a resolução desses problemas é realizada com o auxílio de metodologias implementadas computacionalmente, cada um com uma característica diferente mas com o mesmo objetivo, encontrar o ponto do problema. Os algoritmos em geral são desenvolvidos e aperfeiçoados pensando-se em dois fatores: precisão dos resultados e velocidade de convergência. Outro ponto a ser levado em consideração é a capacidade do algoritmo em encontrar soluções locais e/ou globais e se esses são capazes de solucionar problemas cuja formulação analítica é desconhecida, tais quais os problemas modelados em elementos finitos. Esses problemas são conhecidos como problemas de caixa preta ou *black-box*.

No campo da engenharia, a utilização de métodos de otimização vem se popularizando cada vez mais. Seja na engenharia estrutural, onde deseja-se conceber projetos cada vez mais econômicos, quanto na engenharia espacial, encontrando rotas e situações com menor gasto de combustível, a otimização esta presente. Na maioria das vezes, esses problemas seguem diretrizes de restrição ligadas a sua resistência ou utilização, formando então os limites de aceitação e reprovação de uma solução.

Apesar do avanço no desenvolvimento de computadores cada vez mais potentes, a solução de problemas de otimização podem demorar meses para serem realizadas, seja devido a má otimização do problema em si ou da alta demanda computacional vinculada a realização de cada execução do problema. Um exemplo desse tipo de problema é a avaliação de resistência (restrição constitutiva) em uma órtese, ilustrada na Figura 1, cujas análises são realizadas utilizando o Método dos Elementos Finitos (MEF). Nesse exemplo, a verificação da integridade da órtese pode demorar horas e esta diretamente ligada ao refinamento da malha do modelo de elementos finitos.

Em virtude do custo computacional necessário para determinar os esforços atuantes nesse modelo órtese uma única vez, o uso de metodologias clássicas de otimizações se torna inviável para a solução desse problema. A otimização utilizando um algoritmo genético (GOLDBERG; HOLLAND, 1988) ou por enxame de partículas (EBERHART; KENNEDY, 1995), por exemplo, exigiria um grande número de execuções do modelo de elementos finitos para a determinação dos esforços atuantes. Essa quantidade de chamadas pode facilmente superar a marca de milhares, o que resulta em um processo de otimização

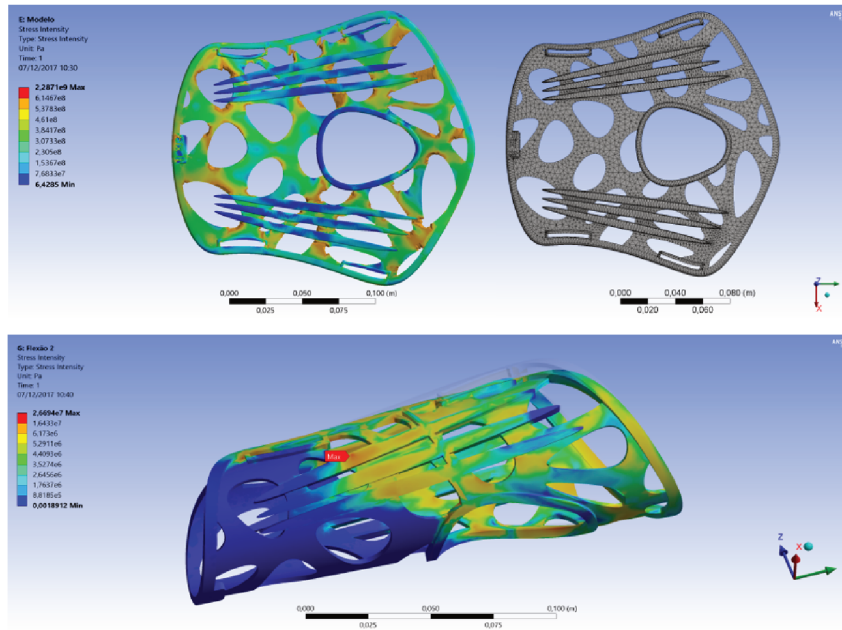


Figura 1 – Órtese *fix-it* modelada em elementos finitos.

demorado.

Problemas de otimização dependentes de modelos em elementos finitos são geralmente os mais relacionados a um grande custo computacional. Tendo dito isso, algoritmos capazes de determinar, com o menor número de chamadas possível, uma combinação de variáveis que resulte em uma solução ótima são os mais indicados no processo de solução desses problemas. Pensando nisso, a utilização de modelos substitutos se mostra uma alternativa viável para a solução de problemas computacionalmente onerosos.

Os modelos substitutos, também chamados de metamodelos, são funções matemáticas aproximadas que tentam representar uma dada função original. Essa representação é obtida a partir de um conjunto discreto de dados do problema original e pode ser realizada utilizando, por exemplo, métodos de ajuste de curvas ou de interpolação. Em geral, os modelos aproximados são computacionalmente mais baratos, tornando-os bons candidatos para realização de otimizações que envolvem um grande número de chamadas.

Diferentes métodos de geração de metamodelos podem ser encontrados na literatura. Modelos baseados em funções polinomiais são utilizados em grande escala. Todavia, esse método pode apresentar um gargalo em se tratando de custo computacional quando se trabalha com problemas com várias variáveis (SASENA, 2002). Outra técnica que vem ganhando espaço no campo dos metamodelos é a utilização de redes neurais artificiais, como é o caso proposto por Santana Gomes (2019). Essa metodologia vem se popularizando em virtude dos estudos de novas técnicas de treinamento de redes neurais, que vem tornando esse processo mais eficiente e performático. Outro método de geração de metamodelos que vem sendo empregado é o de *Kriging* e suas variações. Forrester e Keane (2009), Carraro (2017), Nascentes, Lopez e Fancello (2019), Fábio Nascentes *et al.* (2019) e Noronha (2018) utilizam esse método na geração de modelos aproximados de um problema real. Por sua

vez, Forrester, Sobester e Keane (2008) apresenta como se dá a utilização desse método na resolução de problemas de otimização.

O método de *Kriging* consiste na criação de um modelo de interpolação aproximado e possui como principal vantagem a precisão do metamodelo gerado a partir de quantidade baixa de pontos amostrais. Por se tratar de um modelo interpolado, a aproximação apresenta uma superfície com valor exato no pontos amostrais e um erro entre esses ponto. A Figura 2 apresenta um metamodelo gerado pelo método de *Kriging*.

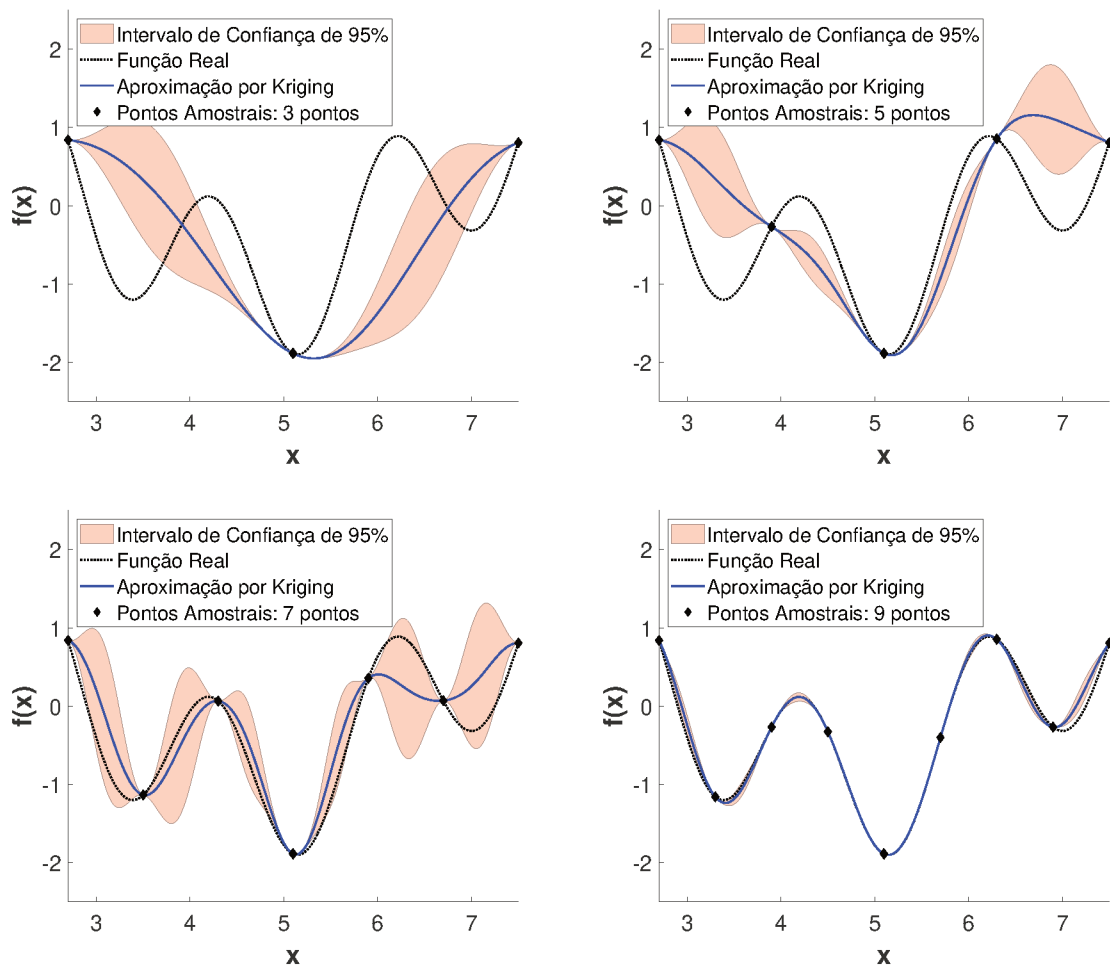


Figura 2 – Exemplo de metamodelo aproximado por *Kriging*

Tendo em vista a qualidade, rapidez e precisão dos metamodelos gerados pelo método de *Kriging*, atrelada a facilidade de implementação, facilmente encontrada na literatura, esta será a metodologia empregada para a geração das superfícies aproximadas utilizadas na estrutura de otimização apresentada neste trabalho.

1.1 OBJETIVOS

Dadas dificuldades apresentadas, esse trabalho tem como motivação a solução de problemas *black-box* cujas restrições são dadas por análises dispendiosas computaci-

onalmente. Por outro lado, a função custo do problema, ou seja, a função que deseja-se otimizar, deve apresentar baixo custo computacional e não esta sujeita a limitações quanto ao número de execuções.

1.2 OBJETIVO GERAL

O presente trabalho tem como objetivo principal o desenvolvimento de um algoritmo de otimização robusto para a solução de problemas cuja formação analítica é desconhecida, também conhecidas com funções *black-box*, com restrições computacionalmente onerosas.

1.3 OBJETIVOS ESPECÍFICOS

Dentro do objetivo geral supracitado pode-se elencar os seguintes objetivos específicos:

- Definir uma metodologia de geração de metamodelos, empregando-a à otimizações de problemas com alto custo computacional;
- Propor uma sequência lógica para a determinação de regiões factíveis no domínio com base na restrição do metamodelo;
- Definir um método de refinamento do modelo aproximado gerado;
- Definir e implementar o algoritmo em uma linguagem de programação;
- Validar a metodologia desenvolvida resolvendo problemas clássicos da literatura;
- Aplicar a metodologia a um problema de otimização estrutural de uma treliça espacial.

2 OTIMIZAÇÃO

Nesse capítulo são apresentados, de maneira sucinta, os conceitos básicos da otimização, bem como as aplicações destes conceitos em problemas de determinação de pontos mínimos, sejam eles com ou sem restrições. São comentadas ainda as dificuldades da aplicação de metodologias clássicas de otimização na solução de problemas com funções de restrição cuja formulação analítica é desconhecida.

2.1 CONCEITOS BÁSICOS

A otimização pode ser definida como um processo de busca pelo melhor valor, seja ele máximo ou mínimo, de um determinado problema matemático que, em geral, é derivado de um modelo que representa uma situação física da realidade (FLETCHER, 1987). Um exemplo de otimização pode ser visto na Figura 2.3 referente a busca do ponto ótimo para uma viga metálica bi-apoiada. Nesse exemplo, dada uma viga metálica bi-apoiada sujeita a um carregamento uniformemente distribuído, deseja-se encontrar o perfil comercial mais barato (f^*) que atenda as solicitações impostas.

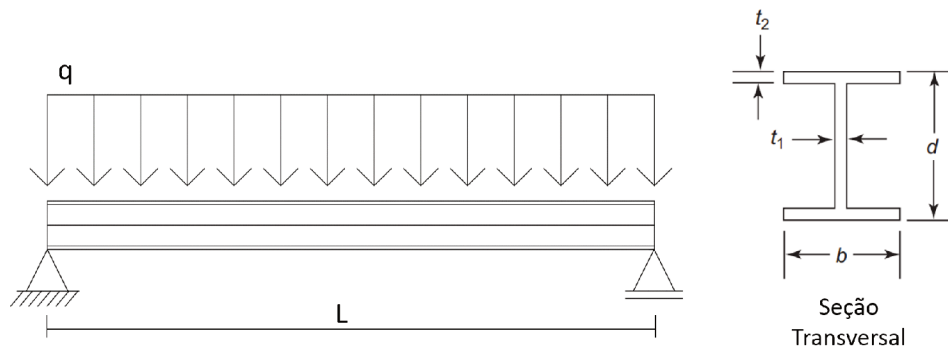


Figura 3 – Exemplo de um problema de otimização

Nesse problema, a busca pode ser resumida em encontrar o perfil com menor seção transversal que, quando multiplicado pelo vão fixo (L), resulte no menor volume possível. Contudo, encontrar o perfil mais leve não é o único fator a ser levado em consideração nesse problema. Esse elemento estrutural precisa ainda atender determinadas condições de restrição (g) que dizem respeito a fatores de segurança e/ou de utilização. Em outras palavras, o elemento precisa suportar tanto os esforços solicitantes, momento fletor e/ou esforço cortante, atuantes no sistema bem como fatores referentes a utilização da estrutura, como a deformação máxima, por exemplo. Portanto, temos uma variável de projeto que é a área da seção transversal do perfil metálico, uma função custo, dada pela multiplicação dessa área pelo comprimento do vão e pelo peso específico do material e três funções de restrição (momento fletor, esforço cortante e deflexão máxima) que determinam se o sistema pode ser aceito ou não.

Diferentes métodos de otimização são apresentados na literatura, cada um com uma característica própria que o difere dos demais. Todavia, esses métodos seguem uma premissa básica da otimização (ARORA, 2012):

Encontrar:

$$\mathbf{x}^* = \{x_1, x_2, \dots, x_m\}, \quad (1)$$

que minimize:

$$f(\mathbf{x}) = f(\{x_1, x_2, \dots, x_m\}), \quad (2)$$

sujeito a:

$$\begin{aligned} h_i(\mathbf{x}) &= h_i(\{x_1, x_2, \dots, x_m\}) = 0, i = 1, 2, \dots, n_h; \\ g_j(\mathbf{x}) &= g_j(\{x_1, x_2, \dots, x_m\}) \leq 0, j = 1, 2, \dots, n_g. \end{aligned} \quad (3)$$

Nesse modelo, \mathbf{x} é o vetor com m variáveis de projeto, \mathbf{x}^* é o vetor ótimo contendo os melhores valores para cada variável de projeto e \mathbf{c} é o vetor com l parâmetros fixos. $f(\mathbf{x}, \mathbf{c})$ é a função objetivo e depende das variáveis de projeto e/ou dos parâmetros fixos e é a função matemática que deseja-se encontrar o ponto de máximo ou mínimo. Por fim, $h(\mathbf{x}, \mathbf{c})$ e $g(\mathbf{x}, \mathbf{c})$ são as restrições do problema, que podem ser de igualdade ou não, respectivamente.

Com essa formulação clássica de um problema, a formação de três classes que definem um problema de otimização pode ser notada, sendo essas: variáveis de projeto, funções objetivo e restrições. As variáveis de projeto caracterizam o sistema a ser otimizado, essas são os parâmetros que podem ser modificados a fim de se buscar a melhor solução possível do problema. A função objetivo, também chamada de função custo, direciona o processo de otimização, é nessa função que são avaliados os valores à serem minimizados ou maximizados. Em um problema a ser otimizado, a função custo pode ser única, gerando apenas um resultado a ser avaliado, ou não, gerando mais de um valor cujo processo de otimização deverá encontrar o melhor ponto que minimize todas as funções custo ao mesmo tempo. Nesse segundo caso, os problemas de otimização são chamados de otimização multiobjetivo. Já as restrições caracterizam as condições a serem respeitadas, essas podem ser devido limitações de segurança, qualidade ou quaisquer outra condição limitante do problema a ser otimizado.

Dado um domínio no espaço gerado pelo intervalo de variação de cada variável de projeto, as restrições atuam como limitantes desse domínio, dividindo-o em regiões factíveis de solução, como ilustrado na Figura 4. Logo, uma das dificuldades a ser levada em conta em algoritmos de otimização é a possibilidade de encontrar regiões descontínuas no domínio admissível.

No exemplo da Figura 4 é possível ver ainda que, embora sejam bem definidas, essas regiões podem apresentar formas irregulares e diversas. Isso dificulta ainda mais a busca pelo mínimo do problema pois exige uma varredura de todo o domínio por novas regiões.

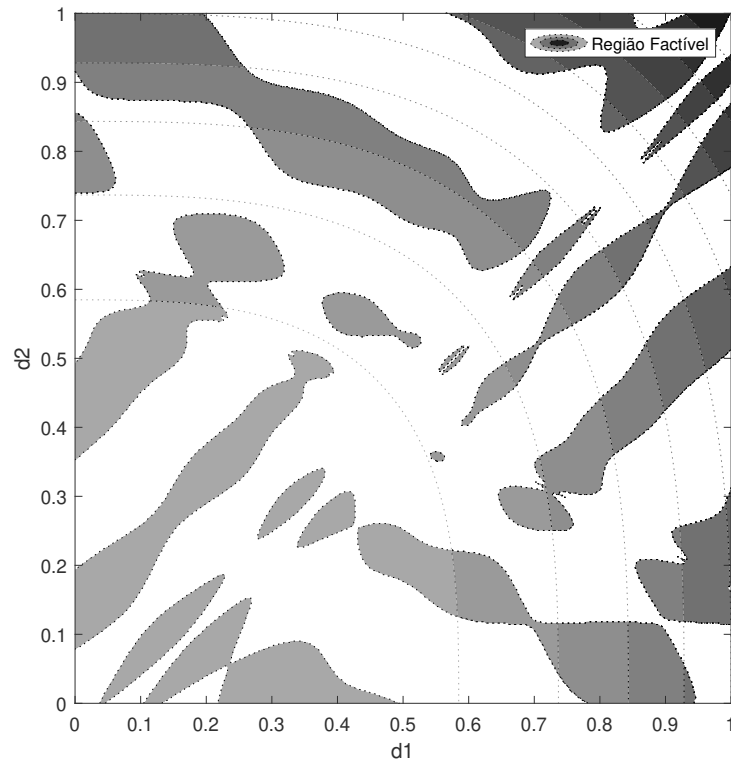


Figura 4 – Exemplo de um problema de otimização com restrições

Em geral, em problemas de otimização deseja-se minimizar o valor da função objetivo, todavia, ainda assim a formulação não é restrita e pode ser generalizada para a determinação do máximo valor de problema. Isso porque encontrar o ponto de máximo de $f(\mathbf{x}, \mathbf{c})$ é o mesmo que minimizar $-f(\mathbf{x}, \mathbf{c})$ (ARORA, 2012).

Com os conceitos de otimização definidos é preciso então entender sobre os tipos de problemas de otimização comumente encontrados e suas características. Entender o tipo de problema e sua classificação pode tornar a busca mais direta e precisa. Otimizar um problema cuja formulação da função custo seja descontínua em todo o domínio utilizando uma métrica com aplicação de derivadas, por exemplo, pode não ser a melhor opção. Regiões constantes do domínio da função custo podem tornar um processo iterativo de busca pelo ponto ótimo ineficiente e impreciso, nunca conseguindo sair daquela região constante. As características dos problemas de otimização são apresentadas na seção 2.2.

2.2 PROBLEMAS DE OTIMIZAÇÃO

Em geral os problemas de otimização podem ser classificados de diferentes formas como quanto a sua convexidade, tipos de variáveis, formulação, restrição, entre outras. Essas classificações ajudam a entender melhor o problema de forma a resolvê-lo de forma mais eficiente.

Zabinsky (2013) classifica os problemas de otimização como apresentado na Figura 5. A seguir são apresentadas algumas definições relevantes para esse trabalho a respeito dessas classificações.

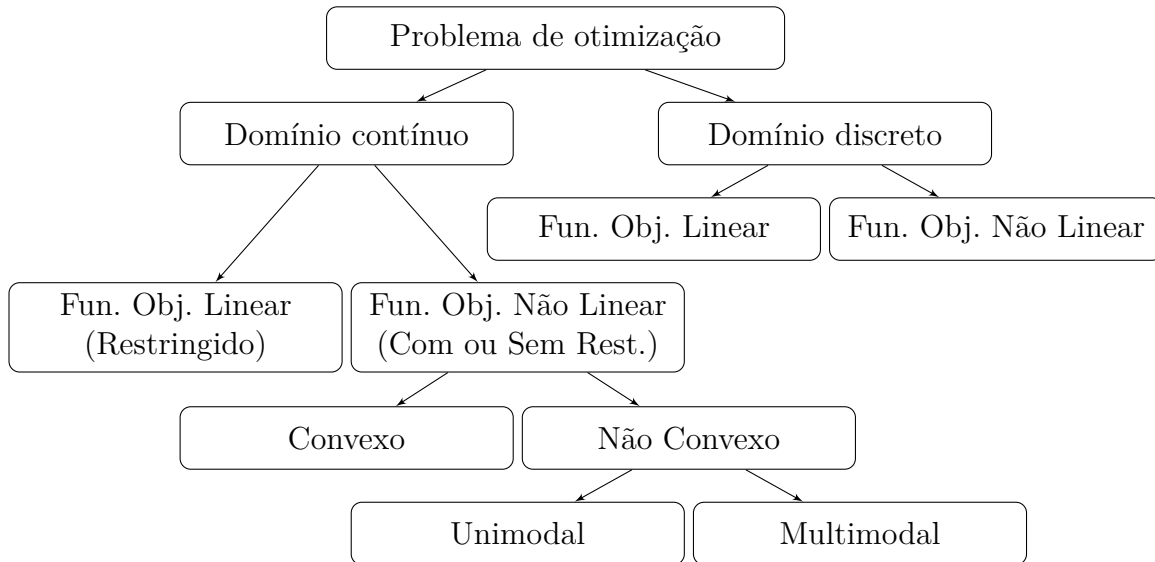


Figura 5 – Classificação dos problemas de otimização.

Fonte – Adaptado de (ZABINSKY, 2013).

2.2.1 Convexidade

Um problema de otimização pode ser definido como convexo ou não convexo, implicando na existência de um ou mais pontos mínimos no problema. Para entender um problema convexo faz-se necessário entender primeiramente o que é um conjunto convexo.

Por definição um conjunto S é dito convexo se, dados quaisquer dois pontos no domínio, a reta formada por esses dois pontos estiver contida em S (HIRIART-URRUTY; LEMARECHAL, 2013).

Matematicamente, essa definição pode ser expandida para o espaço n-dimensional através da representação paramétrica de um segmento de reta \mathbf{x} que une dois pontos quaisquer, \mathbf{x}_1 e \mathbf{x}_2 :

$$\mathbf{x} = \alpha\mathbf{x}_2 + (1 - \alpha)\mathbf{x}_1; 0 \leq \alpha \leq 1. \quad (4)$$

Se o segmento de reta estiver contido em S então o conjunto é dito convexo (ARORA, 2012).

Expandindo essa definição de conjuntos convexos para funções n-dimensionais pode-se então dizer que uma função qualquer é dita convexa em um dado intervalo se:

$$f(\alpha\mathbf{x}_2 + (1 - \alpha)\mathbf{x}_1) = \alpha f(\mathbf{x}_2) + (1 - \alpha)f(\mathbf{x}_1); 0 \leq \alpha \leq 1. \quad (5)$$

Contudo, resolver a Eq. (5) é um processo difícil e dispendioso uma vez que infinitos pontos podem existir em um intervalo qualquer e verificar cada um desses pontos seria

inviável. Um método alternativo e mais simples para se verificar se uma função é convexa é a verificação do Hessiano dessa equação. A matriz Hessiana de uma função é a matriz quadrada da derivada de segunda ordem dessa função e possui a dimensão do número de variáveis dessa mesma função. Através dessa matriz pode-se determinar a curvatura da função em um dado ponto, determinando se essa função é concava ou convexa naquele ponto.

Dado um problema de otimização classificado como convexo pode-se afirmar que no intervalo de análise desse problema existe apenas um ponto de mínimo. Por outro lado, problemas não convexos possuem a característica de poderem conter mais de um ponto de mínimo valor. Nos casos onde podem ocorrer mais de um ponto de mínimo, esses pontos possuem duas denominações possíveis, mínimos locais ($\hat{\mathbf{x}}$) e mínimo global (\mathbf{x}^*).

Mínimos locais são pontos ótimos que podem ocorrer em sub-regiões do domínio do problema. Nesses pontos, avaliações da função custo na vizinhança em quaisquer direção retornam valores maior que o ponto de ótimo local. O mínimo local com menor valor de função custo dentre todos os mínimos locais é chamado de mínimo global, definido como o menor ponto no domínio do problema. A Figura 6 mostra um problema não convexo com mínimos locais e mínimo global.

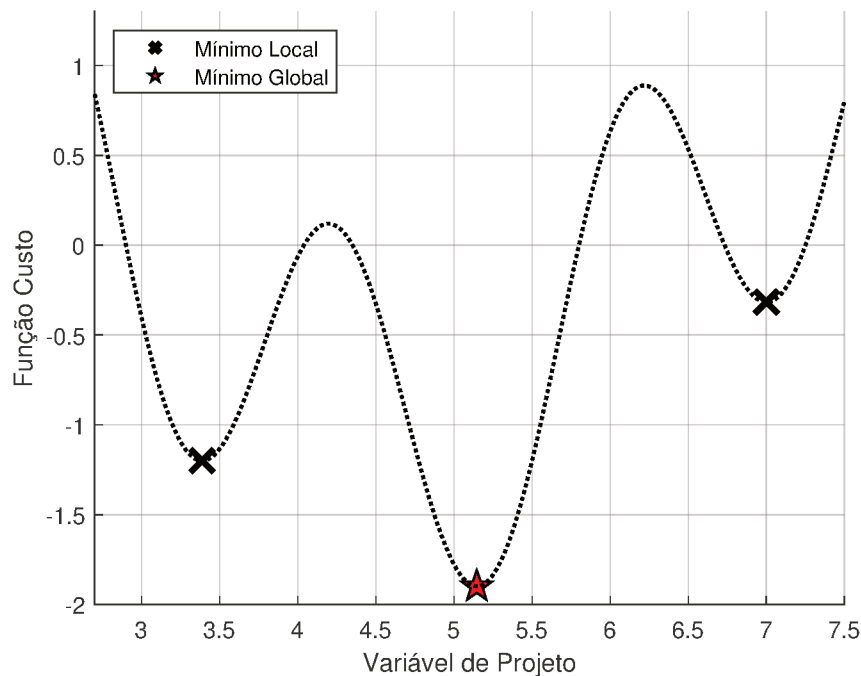


Figura 6 – Exemplo de um problema multimodal

Em grande parte dos casos o resultado esperado em uma otimização é o ponto de mínimo global do problema, logo, várias metodologias são aplicadas aos algoritmos de busca com o objetivo de fugir destes mínimos locais. O capítulo 2.3 apresentará melhor as características inerentes aos algoritmos de busca e suas vantagens.

2.2.2 Formulação

Conhecer a formulação matemática do problema que se deseja otimizar pode tornar o processo mais simples e preciso. Tomando o exemplo da Figura 4, as regiões factíveis nela podem ser mais facilmente encontradas quando se conhece a formulação matemática que descreve a restrição que delimita essas regiões factíveis. O conhecimento da formulação matemática que rege os problemas de otimização, por exemplo, possibilita o uso, em alguns casos, de métodos matemáticos para a determinação do ponto ótimo do problema. Tais métodos podem ser mais precisos, gerando resultados exatos. Esses métodos serão melhor apresentados na seção 2.3.

Contudo, boa parte dos problemas de otimização podem não apresentar uma formulação matemática explícita, um exemplo disso é a otimização estrutural de uma torre de transmissão de energia elétrica. Problemas onde não se conhece a expressão matemática ou informações referentes aos gradientes da função custo ou das funções de restrição são denominados *black-box* (JONES; SCHONLAU; WELCH, 1998). Esses problemas são comuns no campo da engenharia e sua solução, frequentemente, dependem de um grande número de execuções dessas funções desconhecida.

Essa falta de informação exige, cada vez mais, algoritmos mais robustos, capazes de realizar uma busca global no domínio, evitando mínimos locais. Um exemplos de algoritmos com essa filosofia são os algoritmos estocásticos.

2.3 ALGORITMOS DE OTIMIZAÇÃO

Um algoritmo pode ser definido como uma sequência de passos, ou processos, finitos que levam a solução de um dado problema. Na otimização, os algoritmos podem ser divididos em duas classes, determinísticos ou estocásticos.

2.3.1 Algoritmos determinísticos

Os algoritmos determinísticos são chamados assim pois dependem unicamente de condições iniciais e parâmetros de entrada determinísticos para determinar as variáveis de projeto (RENARD; ALCOLEA; GINGSBOURGER, 2013). Em geral, algoritmos de otimização matemáticos são do tipo determinísticos, pois o processo iterativo de melhoria do ponto é feito de acordo com regras determinísticas (CARRARO, 2017).

Métodos determinísticos de otimização são capazes de gerar uma solução única a partir de um mesmo conjunto de parâmetros iniciais. Alterações mínimas nos parâmetros de entrada do modelo determinístico podem influenciar drasticamente no resultado final, segundo afirma Lorenz (1963).

Os métodos determinísticos possuem como pontos positivos uma rápida convergência, uma rápida implementação e a não dependência de valores aleatórios (NASCENTES, F. F. d. S. *et al.*, 2019). Já os pontos negativos, ainda segundo Fábio Felipe dos Santos

Nascentes *et al.* (2019), são a possibilidade de realização de várias avaliações das funções até que se atinja um critério de parada, a dependência de gradiente e hessiana nos pontos analisados, sua sensibilidade aos parâmetros iniciais e, por fim, a difícil aplicação em problemas ruidosos.

2.3.2 Algoritmos Heurísticos

Algoritmos heurísticos são métodos iterativos e estocásticos em que, durante o processo de melhora do ponto ótimo a cada iteração, as decisões e transformações realizadas por meio de números aleatórios (NASCENTES, F. F. d. S. *et al.*, 2019). Essa característica faz com que cada execução do processo de otimização seja única, quando não se define uma semente de geração dos números aleatórios, e não se tenha certeza da obtenção do mesmo valor mínimo em cada simulação de otimização do problema.

Como pontos negativos dos algoritmos heurísticos pode-se listar a necessidade de um grande número de avaliações das funções do problemas, ainda que esses sejam de pequena dimensão. Para problemas cuja função é complexa de ser realizada ou o custo computacional seja algo, essa característica negativa pode tornar o processo inviável. Outro ponto negativo é a não garantia de obtenção do ponto ótimo global e a possível obtenção de resultados diferentes dado o mesmo conjunto inicial de parâmetros, dificultando a replicação da solução. Como pontos positivos, pode-se listar a independência de derivadas e hessianas das funções utilizadas, tratamento de funções descontínuas e otimização de problemas com formulação matemática desconhecida (NASCENTES, F. F. d. S. *et al.*, 2019).

Outra classificação que pode ser feita para os algoritmos de otimização diz respeito ao processo de busca, que pode ser realizado por um único ou vários indivíduos, denominados população, geralmente criados no início do processo de otimização. Os indivíduos, em uma metodologia de otimização, são pontos de análises utilizados em cada iteração no processo de busca pelo ponto ótimo do problema. Ou seja, cada nova iteração, que visa buscar o ponto ótimo do problema pode analisar um único ou vários pontos amostrais, à depender da metodologia adotada para a resolução do problema. Essa diferenciação interfere na capacidade de busca dos algoritmos, dividindo-os em dois grupos, algoritmos de busca local e algoritmos de busca global. Talbi (2009) afirma que, métodos baseados em um único indivíduo tendem a realizar buscas locais enquanto que algoritmos que utilizando populações de diferentes amostras tendem realizar uma busca mais ampla, chamada de busca global.

2.3.3 Algoritmos de busca local

Um processo interativo que tende a convergir para um valor, seja ele de mínimo ou máximo, em uma vizinhança próxima, pode ser chamando de busca local (RIBEIRO, 2006). Algoritmos de busca locais, geralmente, dependem da execução de várias avaliações em

pequenas regiões do domínio, aumentando a chance de encontrar o menor valor naquela região. Muitos algoritmos matemáticos se enquadram nessa categoria, em geral, esses métodos utilizam o gradiente ou a hessiana para encontrar o próximo ponto no processo iterativo. Alguns exemplos de algoritmos de busca local são:

- HLRF (HASOFER; LIND, 1974);
- Método do gradiente conjugado (HESTENES; STIEFEL *et al.*, 1952);
- *Line Search* com derivadas: Bisseção (apresentado em (BURDEN; FAIRES, 1985)) e o Método de Newton (apresentado em (RONCHI, s.d.)).
- *Line Search* sem derivadas: Pesquisa de Fibonacci (KIEFER, 1953).

2.3.4 Algoritmos de busca global

Diferente do processo de busca local, o método de busca global tenta avaliar, de forma mais ampla, o domínio do problema. Esse processo diminui a chance do algoritmo retornar um mínimo local como solução e não encontrar o melhor valor para o problema. Em geral, algoritmos de busca global partem de populações iniciais de amostras espalhadas no domínio do problema, isso faz com que mais regiões desse domínio possam ser exploradas, aumentando a chance de encontrar mínimos isolados. Grande parte dos algoritmos estocásticos seguem essa metodologia de partir de diversos pontos do domínio. Alguns exemplos de algoritmos de busca global são:

- *Genetic Algorithm* (GOLDBERG, 1989);
- *Probabilistic Restart* (LUERSEN; LE RICHE, 2004);
- *Particle Swarm Optimization* (KENNEDY; EBERHART, 1995);
- *Bat algorithm* (YANG, 2010);
- *Search Group Algorithm* (GONÇALVES; LOPEZ; MIGUEL, 2015);
- *Firefly Algorithm* (YANG, 2009).

Em sua grande maioria os algoritmos de otimização tendem a apresentar resultados aproximados do valor de mínimo ou máximo real, logo, entender como refinar o processo de busca é imprescindível para se chegar a um ponto de ótimo. Algoritmos que realizam apenas uma busca global tendem a encontrar regiões com possíveis mínimos, contudo, o valor encontrado por eles pode não ser o valor mínimo. Por outro lado, algoritmos que fazem apenas uma busca local, embora apresentem um resultado mais refinado, tendem a ficar presos em algumas regiões do domínio. Logo, utilizar metodologias que combinem as duas metodologias pode tornar o processo mais eficiente.

Nesse trabalho, propõe-se um novo algoritmo de otimização de problemas com formulação matemática desconhecida e funções de restrição computacionalmente onerosas. Para isso, o algoritmo deverá ser capaz de otimizar, de forma eficiente, problemas não convexos multimodais e que exijam muito tempo computacional para cada avaliação das restrições. A metodologia utiliza metamodelos, que são modelos aproximados das condições de restrição do problema. A principal vantagem na utilização dos metamodelos é a probabilidade de redução do custo computacional do problema. O funcionamento e uso dos metamodelos em problemas de otimização são discutidos no capítulo 3.

3 METAMODELOS

Neste capítulo são apresentados os conceitos sobre modelos substitutos (meta-modelos), suas características e utilizações na otimização de problemas. A geração do metamodelo utilizado na metodologia proposta neste trabalho é realizada pelo método de *Kriging*, proposto por Krige (1951).

Em diversas situações de otimização, o desejo pela representação mais fidedigna dos modelos matemáticos para com a realidade pode tornar o processo de determinação mais demorado. A otimização de uma única prótese para deficientes, sujeita a uma análise de confiabilidade, cujo os esforços são determinados a partir de uma simulação utilizando o método dos elementos finitos, é um exemplo de um processo de otimização que pode demandar dias para ser concluído. Nesse contexto, o uso de ferramentas e/ou técnicas e metodologias que possibilitem diminuir o custo computacional no processo de otimização tem se difundido cada vez mais, uma delas é a modelagem das funções matemáticas dos problemas de otimização baseada em superfícies aproximadas, ou metamodelos.

Segundo (HOYLE, 2006), os Metamodelos são gerados a partir do ajuste de uma curva ou superfície a um conjunto inicial de pontos de amostragem pertencentes ao domínio de busca do problema, determinado a partir dos valores limites de cada variáveis de projeto. Esses modelos aproximados tentam apresentar uma solução barata para um problema caro.

O uso de metamodelos se estende para diversas áreas como confiabilidade, geoestatística, engenharia de softwares, entre outros. Em virtude disso, diferentes abordagens foram criadas para se gerar tais modelos, algumas dessas abordagens são:

- Modelos de regressão, ilustrados por Zeviani, JÚNIOR e Bonat (2013);
- *Kriging* (KRIGE, 1951);
- *Gradient-enhanced kriging* (LIEM; MADER; MARTINS, 2015);
- Metamodelos por Redes Neurais (TRAVESSA *et al.*, 2017);
- Funções de Base Radial (RBF), ilustrados por (BJÖRKMAN; HOLMSTRÖM, 2000);
- Expansão por polinômios de caos, apresentados por RAMOS e PACHECO (2014).

No campo da engenharia estrutural, problemas de otimização computacionalmente caros podem ser encontrados quando se trata de otimizações que envolvem o método dos elementos finitos (MEF) e/ou simulações com análises de probabilidade de falha, por exemplo. Para entender melhor, suponha uma estrutura sujeita a uma combinação

de esforços (\mathbf{x}) cujo o deslocamento (d) em um dado ponto é obtido pelo método dos elementos finitos ($f(\mathbf{x})$), representado matematicamente por:

$$d = f(\mathbf{x}). \quad (6)$$

Avaliar essa função várias vezes pode demandar muito tempo computacional, ainda mais se esse modelo estrutural for submetido a uma análise de probabilidade de falha utilizando um método de simulação como Monte Carlo. Para evitar esse elevado custo, quando se utiliza um metamodelo, deseja-se encontrar uma função matemática aproximada ($\hat{f}(\mathbf{x})$) do tipo:

$$\hat{d} = \hat{f}(\mathbf{x}) \approx f(\mathbf{x}), \quad (7)$$

que forneça um resultado próximo do original (\hat{d}) cujo o tempo de análise em um dado ponto seja significativamente menor.

Uma classe de algoritmos que se utilizam da geração de modelos aproximados a partir de um plano amostral inicial para a resolução de problemas com alto custo de análise é a Otimização Global Eficiente (*Efficient Global Optimization* - EGO) (JONES; SCHONLAU; WELCH, 1998). Desenvolvido por Jones, Schonlau e Welch (1998), nesse método, a função objetivo é substituída por um metamodelo, realizando-se previsões sobre seus valores com a aplicação de um erro a essas. É através dessas informações que novos pontos são inseridos ao conjunto amostral, realizando uma busca pelo ótimo global do problema. A forma em que esses novos pontos são inseridos é o que distingue os vários métodos de EGO e tem grande importância na sua eficiência e robustez (NASCENTES, F. F. d. S. *et al.*, 2019).

Um dos processos mais conhecidos para a geração de metamodelos é o de regressão polinomial. Esse método consiste em gerar um polinômio aproximado com o menor erro possível entre os pontos amostrais. Nesse método o aumento do grau do polinômio resulta em uma melhor função de aproximação, ou seja, para diminuir o erro entre o modelo real e o aproximado basta aumentar o grau do polinômio de aproximação. Essa é uma solução simples e direta para se diminuir o erro, contudo, essa abordagem tende a apresentar problemas quando se deseja aproximar uma equação muito complexa, como ruídos, ou com várias variáveis. Tendo em vista isso, o uso desse método se torna inviável na elaboração de uma metodologia robusta de otimização. Exemplos da utilização desse método podem ser vistos nos trabalhos de Torii e Lopez (2012), Liu, Haftka e Akgün (2000), entre outros.

Outra metodologia a ser comentado é a de geração de metamodelos utilizando redes neurais artificiais (RNA). Esses métodos vem se popularizando devido a adaptatividade das RNAs. Uma rede neural pode ser definida como um processo maciço que tem objetivo adquirir conhecimento e tornar o aprendizado utilizável (HAYKIN, 2007). Para aprender, a RNA deve passar por um processo de treinamento para definir o peso a ser atribuído a

cada neurônio da rede. Na geração de um metamodelo, esse processo de aprendizagem é feito através da realização de vários testes, adquirindo-se um modelo aproximado genérico com erro mínimo entre o modelo gerado e o modelo original.

Nesse trabalho, o método de geração de metamodelos escolhido foi o de *Kriging*. Uma das principais características desse método é capacidade de produzir superfícies precisas, além de fornecer uma estimativa do erro cometido pela aproximação (CARRARO, 2017). Os conceitos e formulação do método utilizado são descritos na seção 3.2.

3.1 PLANO DE AMOSTRAGEM - SAMPLING PLAN

Como mencionado anteriormente, o objetivo do uso de metamodelos na otimização é gerar uma superfície matemática de baixo custo computacional que represente um dado problema *black-box*. Deseja-se que essa superfície seja a mais representativa possível em todo a população, denominada plano amostral. Para que isso aconteça, é desejável que os pontos do plano amostral estejam o mais espaçados possíveis, obtendo assim amostras em diferentes regiões do domínio do problema.

Na falta de um conjunto de pontos iniciais, diferentes métodos de geração dessa população podem ser adotados. Costuma-se utilizar abordagens de geração de populações aleatórias, sendo a mais comum a amostragem pelo Hipercubo Latino (*Latin Hypercube Sampling* - LHS).

O método a amostragem pelo Hipercubo Latino consiste na geração de valores aleatórios para as n variáveis do problema. Para que esses pontos amostras não estejam muito próximos, ou sejam os coincidentes, a metodologia gera um “*grid*”, dividindo o domínio em regiões. É nessa região que os pontos aleatórios serão inseridos. Forrester, Sobester e Keane (2008) apresenta uma abordagem mais detalhada sobre esse método, comparando-o com outros geradores de planos amostrais. A Figura 7 mostra um exemplo, composto por três variáveis de projeto, da geração de 10 pontos aleatórios pelo método LHS.

Em virtude da aleatoriedade na geração do plano amostral, nem sempre essa população é bem representativa. Embora a divisão dos domínios impeça que mais de um ponto seja inserido no mesmo quadrante, isso não impede que sub-regiões vizinhas e/ou adjacentes sejam escolhidas.

Visando melhorar a distribuição das amostras no domínio, Johnson, Moore e Ylvisaker (1990) introduziu a métrica *maximin*. Por definição, essa métrica consiste em, dado um conjunto amostral $\mathcal{S} = \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ ordenados de forma crescente, \mathcal{S} é um plano *maximin* se, dentre vários planos gerados, ele maximizar d_1 e minimizar J_1 . Em que $\mathbf{J} = J_1, J_2, \dots, J_n$ é o número de pares pontos separados pela distância d_j , calculada a partir da Equação:

$$l_1(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = \left(\sum_{j=1}^m |x_j^{(1)} - x_j^{(2)}|^p \right)^{1/p}. \quad (8)$$

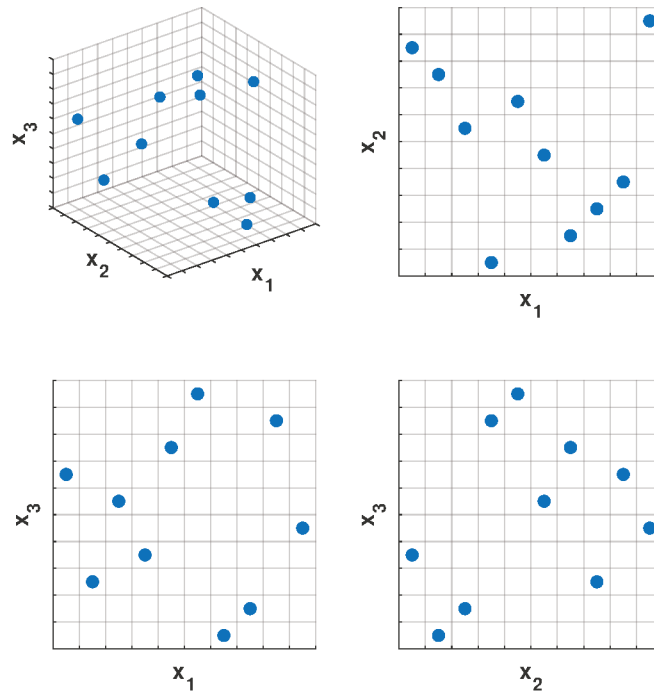


Figura 7 – Hipercubo Latino 3D.

Fonte – Adaptado de Forrester, Sobester e Keane (2008).

Nessa equação, a varável p pode assumir o valor de $p = 1$, para uma distância retangular (Norma Manhattan), ou $p = 2$, para uma Norma Euclidiana.

Morris e Mitchell (1995) expandiram essa métrica para vários conjuntos de planos, propondo uma definição mais completa que seleciona, entre vários planos espaçados, qual o melhor deles. Definindo \mathcal{S} como um plano amostral *maximin* ótimo, o plano que:

1. Maximize l_1 ;
2. Minimize j_1 ;
3. Maximize l_2 ;
4. Minimize j_2 ;
- ⋮
5. Maximize l_n ;
6. Minimize j_n .

Baseado nessa definição Morris e Mitchell (1995) sugeriram uma expressão para fins de otimização, dada por:

$$\Phi_q = (\mathcal{S}) = \left(\sum_{j=1}^n J_j l_x^{-q} \right)^{1/q}. \quad (9)$$

Nesse caso, deseja-se minimizar o valor de Φ_q comparando-se diferentes valores de q . Morris e Mitchell (1995) recomendam avaliar valores de $q = \{1, 2, 5, 10, 20, 50, 100\}$ e escolher o melhor dentre eles (FORRESTER; SOBESTER; KEANE, 2008).

Nesse trabalho, a geração dos planos amostrais foi realizada utilizando o método de Morris e Mitchell (1995) ilustrado por (FORRESTER; SOBESTER; KEANE, 2008) devido a melhor distribuição dos pontos no domínio do problema. Essa melhor distribuição permite um melhor ganho de informações sobre o problema, comportamento esse desejado, uma vez que se trata de funções com alto custo de análise.

3.2 KRIGING

Desenvolvido por Danie Krige (KRIGE, 1951) e batizado por Matheron (1963), *Kriging* é um método de geração de modelos aproximados de um dado modelo original. Utilizado originalmente no campo da geo-estatística para a determinação de depósitos de ouro (CRESSIE, 1992), esse método foi levado inicialmente ao campo da engenharia por Sacks *et al.* (1989) com o objetivo de gerar aproximações em experimentos computacionais.

O método de Krigagem é considerado um modelo de interpolação da superfície aproximada, ou seja, é um método que tem representação exata nos pontos de amostragem enquanto aproxima as regiões não conhecidas (FORRESTER; SOBESTER; KEANE, 2008). Essa característica é útil quando se quer realizar uma otimização precisa com um pequeno número de chamadas da função que rege o problema.

Assim como nos algoritmos de otimização vários estudos são realizados com o objetivo de melhorar o método de *Kriging*. Atualmente, diversas metodologias de geração de metamodelos de krigagem podem ser encontrados no meio acadêmico, sendo algumas delas:

- Método de Krigagem simples (KRIGE, 1951);
- Método de Krigagem ordinária (CAMARGO, 1997)
- Cokrigagem (JOURNEL; HUIJBREGTS, 1978; OLEA, 1991; MYERS, 1982).

Nesse trabalho, o metamodelo que representa a função de restrição cara do problema de otimização foi determinada pelo método de *Kriging* simples com função base de

covariância dada por:

$$\phi^{(i)} = \exp\left(-\sum_{k=1}^m \theta_k |x_k^{(i)} - x_k|^{p_k}\right). \quad (10)$$

Nota-se que, a Equação (10) apresenta uma similaridade a uma função de base gaussiana, característica que possibilita a estimativa do erro entre a função real e a aproximada. Contudo, diferente do que acontece no *RBF*, a função de correlação pode mudar para as n -dimensões (CARRARO, 2017). Características e diferenças entre esses métodos são melhores explicados por Forrester, Sobester e Keane (2008).

3.2.1 Formulação

Suponha um vetor de pontos amostrais inicial $\mathcal{S} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \dots, \mathbf{x}_n\}^T$ com valores observados $\mathbf{y} = f(\mathcal{S}) = \{y_1, y_2, y_3, \dots, y_n\}^T$. Deseja-se encontrar uma função de interpolação que represente $f(\mathcal{S})$ a partir de uma quantidade baixa de pontos amostrais, com o mínimo de erro possível. Pelo método de *Kriging* essa superfície aproximada é gerada pela combinação de um modelo global com um conjunto de variáveis aleatórias correlacionadas (SIMPSON *et al.*, 2001), ou seja:

$$y(x) \approx \hat{y}(\mathbf{x}) = f(\mathbf{x}) + Z(\mathbf{x}), \quad (11)$$

em que $y(x)$ é a função original que deseja-se representar por um modelo matemática aproximado, representado por $\hat{y}(\mathbf{x})$. $f(\mathbf{x})$ é uma função global de aproximação, essa função pode ser representado como uma combinação de termos polinomiais. Por sua vez, $Z(\mathbf{x})$ é um conjuntos de variáveis aleatórias com média zero, variância σ^2 e covariância diferente de zero.

A combinação desses dois termos possibilita a geração de funções de interpolação bem representativa. Enquanto $f(\mathbf{x})$ realiza uma aproximação global no domínio da função original, $Z(\mathbf{x})$ cria variações aleatórios, somados localmente, para que o modelo de *Kriging* interpole os n pontos.

Como mencionado anteriormente, no conjunto $Z(\mathbf{x})$, os pontos amostrais possuem covariância diferente de zero para cada par de ponto, ou seja, existe correlação entre cada ponto amostral analisado. No método de Krigagem, essa correlação é expressa pela Equação base (10) e é expressa como:

$$\text{cor}[Z(\mathbf{x}_i), Z(\mathbf{x}_j)] = \exp\left(-\sum_{k=1}^m \theta_k |x_k^{(i)} - x_k^{(j)}|^{p_k}\right). \quad (12)$$

Escrevendo na forma matricial, tem-se uma matriz de correlação $n \times n$ que apresenta as

correlações entre cada par dada da seguinte forma:

$$\Psi = \begin{pmatrix} \text{cor}[Z(\mathbf{x}_1), Z(\mathbf{x}_1)] & \cdots & \text{cor}[Z(\mathbf{x}_1), Z(\mathbf{x}_n)] \\ \vdots & \ddots & \vdots \\ \text{cor}[Z(\mathbf{x}_n), Z(\mathbf{x}_1)] & \cdots & \text{cor}[Z(\mathbf{x}_n), Z(\mathbf{x}_n)] \end{pmatrix}. \quad (13)$$

Na Equação (12), os parâmetros p_k e θ_k são valores que representam o comportamento da interpolação naquele ponto do domínio. Esses parâmetros podem assumir valores distintos para cada dimensão do problema, contudo, costuma-se adotar o expoente $p_k = 1$ ou $p_k = 2$ para $k = 1, 2, \dots, n$ (FORRESTER; SOBESTER; KEANE, 2008). A Figura 8 apresenta a influência do parâmetro p_k em um problema de uma dimensão. Pode-se notar que esse parâmetro influencia principalmente na suavidade da correlação.

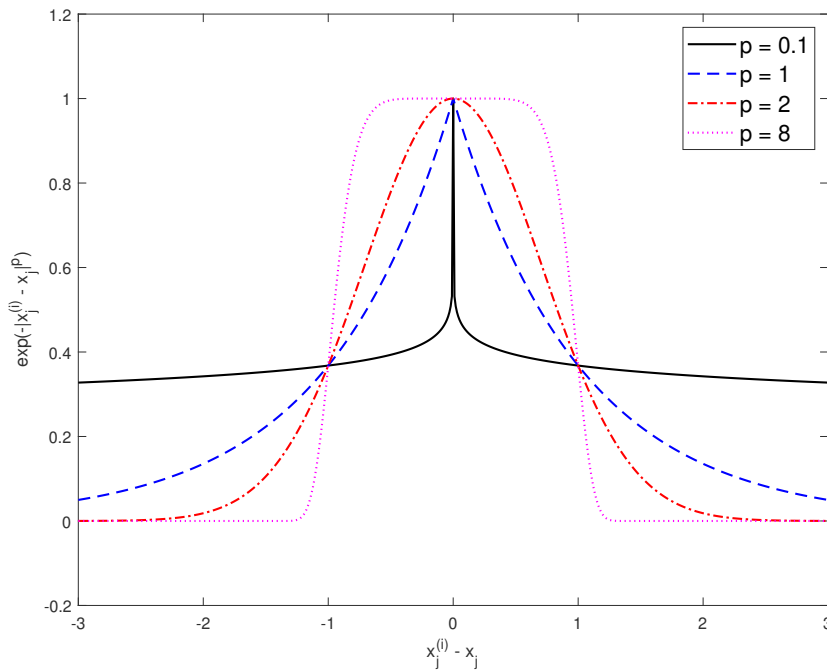


Figura 8 – Correlação com variação de \mathbf{p} .

Fonte – Adaptado de Forrester, Sobester e Keane (2008).

O parâmetro θ_k , por sua vez continua livre para assumir diferentes valores para cada uma das dimensões do problema. Essa característica faz com que o número de parâmetros no modelo de *Kriging* seja igual à dimensão do problema a ser interpolado. Ou seja, quanto maior a dimensão do problema, maior o número de variáveis no modelo de aproximação. Esse elevado número de variáveis pode tornar o processo de interpolação ligeiramente custoso. Todavia, o tempo computacional gasto durante o processo de geração do metamodelo pode vir a ser desconsiderado em função da precisão do modelo e do custo de análise da função real (FORRESTER; KEANE, 2009). Estudos relacionados a essa compensação de tempo computacional foram realizados por Costa, Pronzato e Thierry

(1999), Kim, Lee e Choi (2009) e Krishnamurthy (2005). A influência do parâmetro θ_k na correlação entre as variáveis aleatórias é ilustrada na Figura 9.

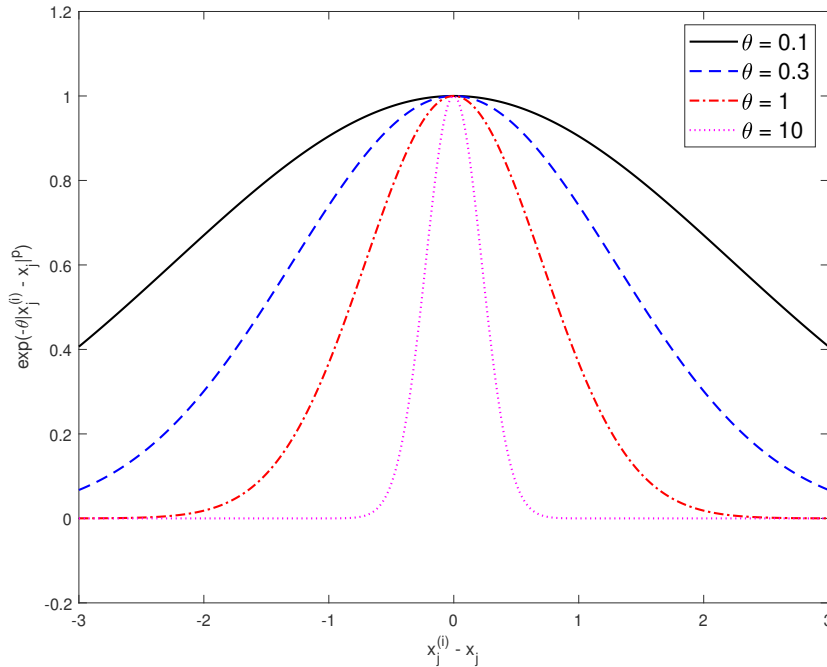


Figura 9 – Correlação com variação de θ .

Fonte – Adaptado de Forrester, Sobester e Keane (2008).

Observa-se ainda que, a Equação (12) é intuitiva e dependente das distâncias entre os pontos. Em outras palavras, quando mais próximo os pontos, ou seja, $d_k^{(i)} - d_k^{(j)} \rightarrow 0$, o valor da correlação tenderá a 1, caracterizando uma correlação muito próxima. Por outro lado, quando a distância entre os pontos, $d_k^{(i)} - d_k^{(j)} \rightarrow \infty$, a correlação tenderá a zero, ou seja, apresentando uma baixa correlação entre esses pontos.

A correlação apresenta uma medida padronizada de relação entre duas variáveis aleatórias, variando entre -1 e 1. Contudo, faz-se necessário mensurar o quanto as variáveis de projeto variam em conjunto, ou seja, o quanto uma variável varia dado que a outra variou. Esse valor é obtido através da covariância, expressa por:

$$\text{Cov}[Z(\mathbf{x}_i), Z(\mathbf{x}_j)] = \sigma^2 \Psi. \quad (14)$$

Essa correlação entre variáveis se dá devido a esperança de que a formulação matemática que descreve o problema seja contínua e suave no domínio de soluções possíveis (FORRESTER; SOBESTER; KEANE, 2008).

Como pode ser visto, os parâmetros p_k e θ_k são quem ditam as correlações entre esses pontos e, observando a Equação (11), essa correlação interfere diretamente no plano interpolado. Dessa forma, deseja-se determinar esses parâmetros de modo que a interpolação seja a mais representativa possível, minimizando o erro entre a função origi-

nal e a aproximada. Essa determinação pode ser feita através do estimador de máxima verosimilhança (*Maximum Likelihood Estimate* - MLE) (MONTGOMERY; RUNGER, 2010). Esse método possibilita estimar parâmetros de um modelo estatístico e é dado por (FORRESTER; SOBESTER; KEANE, 2008):

$$L = \frac{1}{(2\pi\sigma^2)^{n/2} |\Psi|^{1/2}} \exp \left[-\frac{(\mathbf{y} - \mathbf{1}\mu)^T \Psi^{-1} (\mathbf{y} - \mathbf{1}\mu)}{2\sigma^2} \right]. \quad (15)$$

Com o objetivo de simplificar as operações realizadas nesse estimador, costuma-se aplicar o logaritmo natural a Equação (15). O resultado final pode ser chamado de *log-likelihood* e é dado por:

$$\ln(L) = -\frac{n}{2} \ln(2\pi) - \frac{n}{2} \ln(\sigma^2) - \frac{1}{2} \ln |\Psi| - \frac{(\mathbf{y} - \mathbf{1}\mu)^T \Psi^{-1} (\mathbf{y} - \mathbf{1}\mu)}{2\sigma^2}. \quad (16)$$

Usando o critério da primeira derivada, pode-se determinar o ponto de máximo da função *log-likelihood*, dado pela derivada da Equação 16 e igualando a zero, obtendo-se assim:

$$\hat{\mu} = \frac{\mathbf{1}^T \Psi^{-1} \mathbf{y}}{\mathbf{1}^T \Psi^{-1} \mathbf{1}}, \quad (17)$$

$$\hat{\sigma}^2 = \frac{(\mathbf{y} - \mathbf{1}\mu)^T \Psi^{-1} (\mathbf{y} - \mathbf{1}\mu)}{n}. \quad (18)$$

Substituindo as Equações 17 e 18 na Equação 16 e eliminando os termos constantes, pode-se obter uma aproximação do ponto mínimo da função *likelihood* expressa como:

$$\ln(L) = -\frac{n}{2} \ln(\hat{\sigma}^2) - \frac{1}{2} \ln |\Psi|. \quad (19)$$

Essa equação pode ser chamada de função concentrada de probabilidade logarítmica. Essa expressão depende unicamente dos valores da matriz de correlações (Ψ), que por sua vez, depende dos parâmetros p e θ de cada uma das dimensões do problema.

Por se tratar de um modelo de interpolação exata nos pontos mostrais, o erro entre o modelo real e o aproximado em cada um desses pontos deve ser igual zero. Devido a possibilidade dos parâmetros \mathbf{p} e $\boldsymbol{\theta}$ se tratarem de conjuntos de variáveis com n valores cada, a determinação dos valores atribuídos a esses conjuntos que minimizem a Equação (19) não é facilmente determinada aplicando-se o critério da primeira derivada, por exemplo. Todavia, esse problema nada mais é que um problema de otimização e, tendo em vista que o custo computacional de chamada da função concentrada de probabilidade logarítmica é relativamente baixo, o problema pode ser resolvido utilizando uma abordagem de otimização estocástica. Nesse trabalho, a determinação desses valores será realizada utilizando o algoritmo *Particle Swarm Optimization* (KENNEDY; EBERHART, 1995) nativo do software MATLAB $\text{\textcircled{R}}$.

3.2.2 Previsor do *Kriging*

De posse dos parâmetro \mathbf{p} e $\boldsymbol{\theta}$ que maximizam a função concentrada de probabilidade logarítmica, pode-se então obter uma expressão matemática de baixo custo computacional que aproxime, de forma satisfatória, a função original. Agora, essa função dependente apenas dos valores das variáveis de projeto em que se deseja estimar o valor da função custo. Essa equação foi expressa por Sacks *et al.* (1989) e é dependente do vetor de \mathbf{r} que contem a correlação entre o ponto que se deseja estimar, \mathbf{x}_u , e os pontos do plano amostral inicial. Essa equação pode é dada por:

$$\hat{y}(\mathbf{x}_u) = \hat{\mu} + \mathbf{r}^T \boldsymbol{\Psi}^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu}). \quad (20)$$

Como mencionado anteriormente, o uso de uma expressão Gaussiana permite que o método de *Kriging* estime o erro no processo de previsão apresentado pela Equação 20. Sacks *et al.* (1989) calculou esse Erro Médio Quadrático (MSE) da função de previsão como sendo:

$$s^2(\mathbf{x}) = \hat{\sigma}^2 \left[1 - \mathbf{r}^T \boldsymbol{\Psi}^{-1} \mathbf{r} + \frac{(1 - \mathbf{1}^T \boldsymbol{\Psi}^{-1} \mathbf{r})^2}{\mathbf{1}^T \boldsymbol{\Psi}^{-1} \mathbf{1}} \right]. \quad (21)$$

A principal etapa que demanda maior tempo de cálculo do modelo de *kriging* é a determinação dos parâmetros de correlação citados anteriormente (NASCENTES, F. F. d. S. *et al.*, 2019). Essa maior exigência computacional se dá ao fato de que, nas equações 18 e 19, serem necessários a inversa e o determinante da matriz de correlação $\boldsymbol{\Psi}$, apresentada na Equação 13 (CARRARO, 2017). Contudo, visando o aumento da eficiência do cálculo na determinação da inversa e do determinante dessa matriz, a decomposição de Cholesky pode ser utilizada, uma vez que essa se trata de uma matriz positiva definida (FORRESTER; SOBESTER; KEANE, 2008).

Outra abordagem utilizada por diversos autores, como Sacks *et al.* (1989), Schonlau, Welch e Jones (1998), Jones, Schonlau e Welch (1998), Forrester, Sobester e Keane (2008) e Forrester e Keane (2009), entre outros, é a fixação do parâmetro $\mathbf{p} = 2$. Essa abordagem favorece uma suavização entre as variáveis do problema e, ao mesmo tempo, reduz o número de variáveis de projeto a serem otimizadas no subproblema de otimização da equação (19).

3.3 REFINAMENTO DO METAMODELO

Como mostrado na seção 3, os metamodelos são modelos aproximados de um dado problema real. Essa aproximação interpola globalmente pontos amostrais de um dado domínio. Aplicado no campo da otimização, é interessante que essa aproximação seja o mais precisa possível próximo as regiões do mínimo do problema. Para isso, pontos devem ser adicionados ao plano amostral do problema, esse método é conhecido como critério de

preenchimento, ou *Infill Criteria*. A Figura 10 apresenta um exemplo de refinamento de um problema unidimensional.

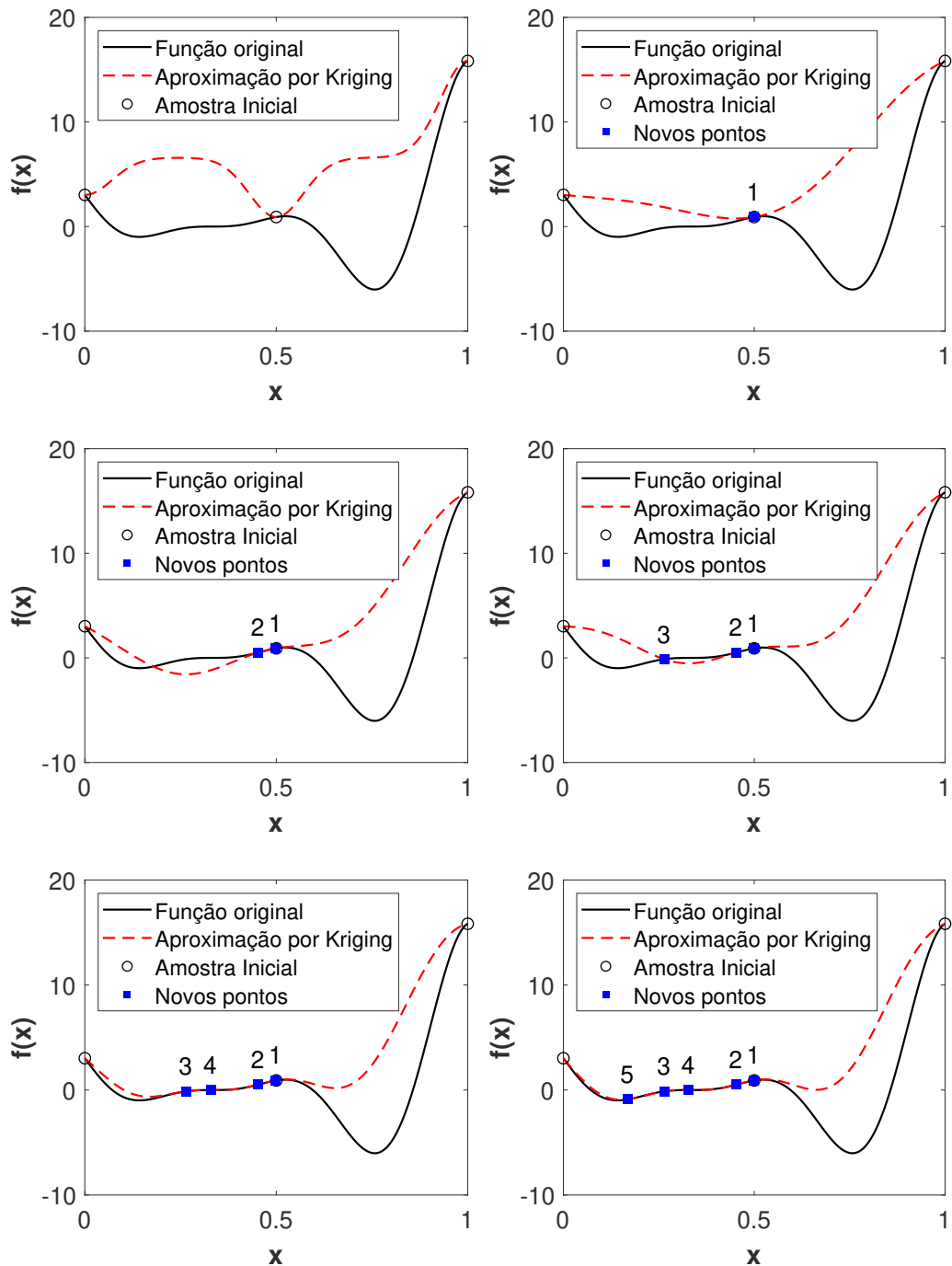


Figura 10 – Aproximação do modelo de *kriging* com refinamento através da inserção de pontos.

Fonte – Adaptado de Forrester, Sobester e Keane (2008).

A melhoria da aproximação do metamodelo depende diretamente do plano amostral que se tem. Diferentes métodos para estimar o melhor local para se inserir um novo ponto podem ser encontrados na literatura. Nesse trabalho, será utilizado o método *Expected*

Improvement, detalhado na seção 3.3.1.

3.3.1 Expected Improvement - (EI)

O método da inserção de pontos pela máxima melhoria esperada consiste em estimar a quantidade de informação que um dado ponto trará se adicionado a amostra. Essa estimativa é obtida por:

$$E[I(\mathbf{x})] = \begin{cases} (y_{min} - \hat{y}(\mathbf{x}))\Phi\left(\frac{y_{min} - \hat{y}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right) + s\phi\left(\frac{y_{min} - \hat{y}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right), & \text{if } \hat{s} > 0 \\ 0, & \text{if } \hat{s} = 0 \end{cases}. \quad (22)$$

Nessa equação, $\Phi(\cdot)$ e $\phi(\cdot)$ são distribuições normais de densidade cumulativa e probabilidade respectivamente (FORRESTER; SOBESTER; KEANE, 2008). É possível notar que, quando $\hat{s} = 0$ o valor de $E[I(\mathbf{x})] = 0$. Isso ocorre também com o $P[I(\mathbf{x})]$ e se dá porque, nessa região, o valor da aproximação é igual ao valor da função original.

A Equação 22 pode ainda ser avaliada usando a função erro, resultando em:

$$E[I(\mathbf{x})] = (y_{min} - \hat{y}(\mathbf{x})) \left[\frac{1}{2} + \frac{1}{2} \operatorname{erf} \left(\frac{y_{min} - \hat{y}(\mathbf{x})}{\hat{s}\sqrt{2}} \right) \right] + \hat{s} \frac{1}{\sqrt{2\pi}} \exp \left[\frac{-(y_{min} - \hat{y}(\mathbf{x}))^2}{2\hat{s}^2} \right]. \quad (23)$$

A Figura 11 apresenta um exemplo de avaliação de espereça de melhoria em um dado ponto do domínio. A área hachurada nesse exemplo representa a quantidade da espereça de melhoria nesse ponto. A esperança de melhoria pode ser vista na curva tracejada na parte inferior dessa mesma figura. O método *EI* consiste em adicionar um ponto amostral onde essa esperança é máxima, ou seja, onde, percorrendo todo o domínio, a área formada sob a curva é máxima.

Em geral, maximizar $E[I(\mathbf{x})]$ é a melhor forma de se encontrar o mínimo global de uma função. Contudo, se as suposições utilizadas nesse método forem falsas e não representarem o comportamento real do problema, a convergência para o ponto de mínimo global pode ser lenta ou nunca acontecer (FORRESTER; SOBESTER; KEANE, 2008). Nesse caso, uma abordagem mista, utilizando diferentes métodos de inserção de pontos, pode se mostrar mais interessante.

Nesse trabalho, foi utilizado um método híbrido de inserção de novos pontos no domínio do problema. Foi utilizado o método de máxima esperança de melhoria somada a inserção de n pontos arbitrados aleatoriamente utilizando o *Latin Hypercube Sampling*. Apesar de inserir mais dependência de números pseudoaleatórios, isso garante que pontos diferentes do domínio sejam avaliados em uma única execução do processo iterativo e evita a avaliação apenas em regiões próximas. A não utilização de uma abordagem híbrida entre o *Expected Improvement* e o *Probability of Improvement* se deu após a verificação,

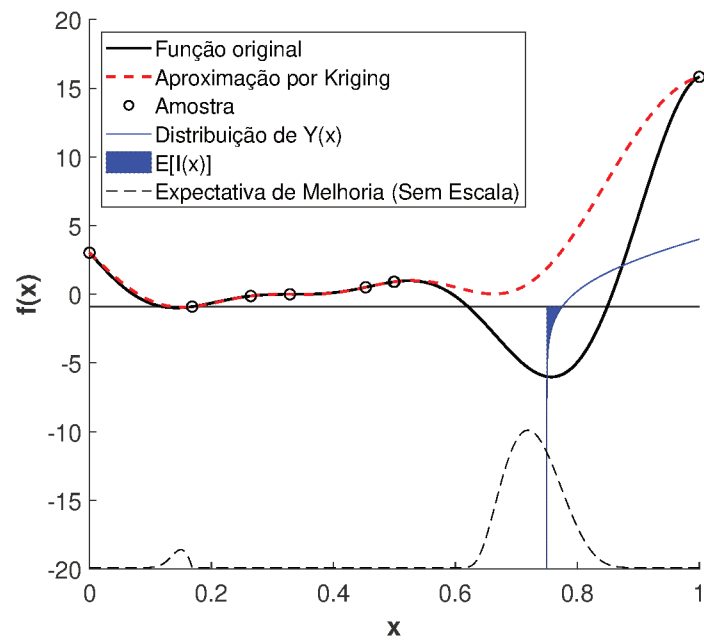


Figura 11 – Interpretação gráfica da espereça de melhoria.

em algumas execuções do processo iterativo, de uma proximidade entre os pontos de maximização obtido pelos dois métodos. Embora essa aproximação trouxesse um maior ganho de informação naquela região próxima. Nessas situações, se tinha pouco ganho de conhecimento em regiões outras do domínio que poderiam conter uma região factível, informação essa bem explorada na metodologia proposta nesse trabalho.

4 TRABALHOS SEMELHANTES

O uso de metamodelos vem sendo cada vez mais popular no campo da otimização em virtude da possibilidade de redução do custo computacional. Esse capítulo trata sobre alguns trabalhos desenvolvidos com foco em algoritmos do tipo Efficient Global Optimization (EGO), que usa uma técnica de modelagem substituta baseada em Processos Gaussianos e é usada na solução de problemas com funções caixa preta, *black-box*, de alto custo computacional, e sobre o algoritmo *Global Optimization with Surrogate Approximation of Constraints* (GOSAC), que inspirou o método proposto nesse trabalho.

4.1 CENÁRIO GERAL

Diversas metodologias que tratam de otimizações com funções computacionalmente onerosas vem sendo desenvolvidas. Esses trabalhos apoiam-se no fato de que problemas que representem de forma mais próxima a realidade resultam em modelos mais complexos e refinados, aumentando seu custo computacional de processamento e análise. Em geral, esses algoritmos são desenvolvidos para problemas onde não se pode desconsiderar o custo computacional inerente a avaliação da função objetivo. Esses algoritmos contam ainda com falta de restrições do domínio de busca, sendo restringidas somente pelos limites de valores das variáveis de projeto (MÜLLER; WOODBURY, 2017).

Na literatura, são encontrados trabalhos que utilizam diferentes técnicas para a abordagem de problemas que envolvam funções caixa preta. Em seu trabalho, Schonlau, Welch e Jones (1998) propôs uma metodóloga de melhoria esperada adaptado para problemas com função objetivo tipo caixa preta e funções de restrição conhecidas. Por sua vez, Gramacy e Polson (2011) desenvolveram um método para a solução de problemas com função objetivo e funções de restrições do tipo caixa preta através da exploração de Monte Carlo sequencial para produzir um algoritmo de projeto rápido. Williams *et al.* (2010) desenvolveram uma técnica que considera uma função objetivo do tipo *black-box* e restrições acoplados por operadores integrais. Nesse trabalho, Williams *et al.* (2010) consideraram uma inserção de pontos baseado na maximização de EI ao mesmo tempo que se tenta minimizar o erro quadrado da previsão naquele mesmo ponto. Embora esses trabalhos apresentem bons resultados, o ponto negativo deles é a dificuldade de utilização desses modelos em outros problemas. Pensando nisso, Robert B Gramacy *et al.* (2016) propuseram uma metodologia estatística baseada no método Lagrangiano Aumentado, proposto por (BERTSEKAS, 2014).

O Lagrangiano Aumentado é caracterizado como uma ferramenta de programação capaz de transformar problemas com restrições gerais em uma série de problemas sem restrições ou com restrições simples, localizando mínimos locais do problema. Todavia, em problemas com um grande número de mínimos locais, a identificação do mínimo atingido por esse método é difícil de ser determinada. Pensando nisso, Robert B Gramacy

et al. (2016) propuseram uma abordagem híbrida envolvendo os modelos de superfície de resposta, a melhoria esperada (EI) e o Lagrangiano aumentado. A ideia consiste na geração de metamodelos para a função objetivo e as funções de restrições do problema de forma separada, determinando pontos de melhoria da aproximação por um método EI adaptado para a utilização junto do Lagrangiano Aumentado. A implementação dessa metodologia esta presente no trabalho de Robert B Gramacy *et al.* (2016). Embora o modelo tenha sido empregado apenas para problemas com funções objetivo conhecidas e funções de restrições dependentes de simulações externas, o método pode ser empregado em problemas onde a função custo é também desconhecida e aproximada por um modelo substituto (GRAMACY, Robert B *et al.*, 2016).

Trabalhos mais recentes que utilizam modelos aproximados foram desenvolvidos por Müller, Shoemaker e Piché (2013) no desenvolvimento de um algoritmo de modelo aproximado para problemas *black-box* computacionalmente onerosos não lineares mistos, nomeado de *SO-MI*. Outro trabalho foi realizado por Müller, Krityakierne e Shoemaker (2014), com o desenvolvimento de uma metodologia de otimização para problemas computacionalmente onerosos, multi-modais, com alta dimensão, utilizando modelos aproximados no processo de busca pela solução do problema. Essa metodologia, denominada *SO-MODS*, essa metodologia foi é caracterizada com uma extensão do algoritmos *DY-CORS*, desenvolvido por Regis e Shoemaker (2013), e foi validado em problemas com 10, 20 e 30 variáveis de projeto e, segundo a os idealizadores do método, este pode ser empregado a problemas com no máximo 50 variáveis.

O *SOCEMO* (MÜLLER, 2017) é um algoritmo de otimização substituta de problemas multiobjetivos computacionalmente caros. Esse algoritmo possui como característica a possibilidade de utilização em problemas cuja função custo seja de formulação desconhecida e computacionalmente onerosa, em que a realização de algumas centenas de avaliações podem podem demandar uma grande quantidade de tempo. Outra característica relevante é que o *SOCEMO* não é um algoritmo evolutivo que se baseia em uma população. Em vez disso, o método utiliza uma combinação de estratégias de amostragem local e global para aprimorar localmente as melhores soluções (MÜLLER, 2017).

O *MATSuMoTo*, MATLAB Surrogate Model Toolbox, é um toolbox desenvolvido para uso no software MATLAB ® que realizada a otimização de problemas de otimização global com restrição de caixa, computacionalmente caros e de formulação desconhecida, e utiliza modelos aproximados para a representação das funções custos e de restrição. É um toolbox robusto e capaz de solucionar uma gama enorme de problemas que possuam funções custo de restrição computacionalmente onerosas. Todavia, não é uma ferramenta recomendada para uso em problemas cuja função custo possa ser avaliada em uma fração de segundos, isso porque, nesses casos, o esforço computacional necessário para construir o modelo substituto supera o tempo de avaliação da função custo original.

Problemas cuja função objetivo possui um baixo custo de avaliação sujeita a restri-

ções com elevado custo computacional, como é o caso dos problemas avaliados por Robert B Gramacy *et al.* (2016), podem ser encontrados facilmente no âmbito da engenharia. Um exemplo desse tipo de problema é a otimização das vigas de uma edificação com múltiplos pavimentos. Nesse exemplo, a função custo pode ser definida como sendo o somatório algébrico dos custos de cada elemento estrutural que compõem a edificação. Por outro lado, as condições de restrição dependem da execução de um código/software de elementos finitos para verificar o cumprimento ou não de limitações de resistência última e/ou de utilização. Técnicas que exploram o fato da função objetivo ter baixo custo computacional ainda são pouco exploradas, dado seu potencial. Abordando esse tipo de problema em específico, Müller e Woodbury (2017) apresentaram boas soluções com um algoritmo de otimização global com aproximação substitutiva de restrições, batizado de *GOSAC*. Esse algoritmo foi usado como base para essa dissertação e é apresentado na seção 4.2.

4.2 ALGORITMO DE OTIMIZAÇÃO GLOBAL COM APROXIMAÇÃO SUBSTITUTIVA DE RESTRIÇÕES - GOSAC

O algoritmo *GOSAC* baseia-se no algoritmo geral de otimização de problemas com função objetivo computacionalmente caras. O método consiste em realizar uma otimização da função custo do problema sujeita a uma função de restrição aproximada de um modelo computacionalmente caro. Dessa forma, a técnica aumenta a possibilidade de encontrar um ponto próximo do valor mínimo da função custo que atenda todas as condições de restrição. Os autores adotaram funções de base radial (RBF) para geração do metamodelo interpolado. O Algoritmo 1 apresenta uma visão geral da metodologia.

No Algoritmo 1, o passo 1 tem por finalidade criar um plano amostral capaz de fornecer uma boa aproximação da superfície de restrição. Para isso, Müller e Woodbury (2017) sugere que o número de pontos amostrais iniciais seja dado por $n_0 = 2(m+1)$, onde m é a dimensão do problema de otimização. Esses pontos podem ser gerados aleatoriamente ou serem inseridos pelo próprio usuário do algoritmo. Todavia, é comum a utilização do método hipercubo latino para uma geração espaçada dessa amostras.

Dado os pontos iniciais, deverá ser avaliado quais os valores da função custo e das funções de restrição onerosas em cada um desses. Feito isso, verifica-se a existência ou não de um ponto factível entre os pontos iniciais, utilizando-se a Fase I do método caso não haja um ponto factível. Esse processo é apresentado no Algoritmo 2 e tem como objetivo encontrar o primeiro ponto viável que aceita todas condições de restrição.

Na Fase I (Algoritmo 2), a metodologia cria uma função substituta para cada função de restrição computacionalmente onerosa do problema. Em seguida, realiza-se a busca de um conjunto de pontos com base em uma otimização multiobjetivo dos modelos aproximados criados no passo anterior. Feito isso, descarta-se os pontos amostrais muito próximos e armazena-se os pontos que não atendem as condições de restrições substitutas. O novo ponto a ser adicionado ao plano amostral do problema é então determinado como

Algoritmo 1 *GOSAC*: Otimização global com aproximação substitutiva de restrições.

- 1: Crie um plano amostral inicial (\mathbf{S}_0) com n_0 pontos. Faça a avaliação dos pontos amostrais na função custo e nas funções de restrição do problema;
 - 2: Defina $n = n_0$ (número de pontos para avaliação) e $\mathbf{S} = \mathbf{S}_0$ (conjunto de pontos amostrais);
 - 3: **se** Se não houver um ponto factível no conjunto de pontos iniciais **então**
 - 4: Fase I da otimização: encontre o primeiro ponto factível (Algoritmo 2);
 - 5: **fim se**
 - 6: **se** O critério de parada não for atingido **então**
 - 7: Fase II da otimização: melhorar o melhor ponto encontrado até então (Algoritmo 3);
 - 8: **fim se**
 - 9: Retorne a melhor solução encontrada.
-

o que tem menor valor da soma das violações do modelo aproximado. Em outras palavras, soma-se o quanto cada ponto viola cada restrição e pega-se o ponto com menor valor dessa soma. O processo é repetido até que se encontre um ponto que respeite todas as restrições.

Algoritmo 2 Fase I da otimização: encontre o primeiro ponto factível.

- 1: **enquanto** O critérios de parada não for satisfeito **faça**
 - 2: Defina um modelo RBF cubico para cada função de restrição ($s_j(\mathbf{x}) = c_j(\mathbf{x}) + \epsilon_j(\mathbf{x}), j = 1, \dots, m$);
 - 3: Resolva o problema de otimização multiobjetivo $\min_{\mathbf{x} \in \Omega} \mathbf{s}(\mathbf{x}) = [s_1(\mathbf{x}), \dots, s_m(\mathbf{x})]^T$ e obtenha o conjunto de pontos não avaliado $\mathbf{Y} = \mathbf{y}_1, \dots, \mathbf{y}_p$;
 - 4: Descarte os pontos y_i em que $\|y_i - x_k\|_2 < \Delta_{min}, \forall \mathbf{x}_k \in \mathbf{S}$ e $\mathbf{y}_i \in \mathbf{Y}$ e defina o conjunto como \mathbf{Y}' ;
 - 5: Encontre os pontos que não atendem as condições de restrição e descarte-os. Para cada ponto restante, calcule a soma das violações $\sum_{j=1}^m c_j(\mathbf{x}, 0)$. Defina o ponto com menor valor da soma como \mathbf{x}_{new} ;
 - 6: Avalie \mathbf{x}_{new} nas funções de restrição caras;
 - 7: Faça $n = n + 1, \mathbf{S} = \mathbf{S} \cup \mathbf{x}_{new}$;
 - 8: **se** Todas as restrições forem satisfeitas em \mathbf{x}_{new} **então**
 - 9: Finalize a Fase I;
 - 10: **fim se**
 - 11: **fim enquanto**
-

Encontrado ao menos um ponto factível, a segunda fase do algoritmo visa melhorar esse ponto. Nessa etapa, realiza-se a minimização da função objetivo com as restrições aproximadas geradas por um algoritmo de metamodelagem, isso permite que pontos mais próximos ao mínimos locais sejam avaliados. Para que o método não fique preso em mínimos locais, o algoritmo conta com o parâmetro Δ_{min} , usado para mensurar a distância entre dois pontos amostrais. Para isso, verifica-se se dois pontos estão muito próximo e, em caso afirmativo, um novo ponto distante é estimado. Por exemplo, dado um ponto amostral minimizado da função objetivo com as restrições aproximadas, se um novo ponto a ser avaliado estiver a uma distância menor que o estimador Δ_{min} , deduz-se que esse

ponto já esta bem representado como ponto de mínimo. Nesse caso, deve-se estimar um ponto qualquer em outra parte do domínio no intuito de se encontrar outras regiões que possam conter novos mínimos.

Outra característica desse parâmetro é o fato de possibilitar que o algoritmo faça buscas locais e globais. Valores altos para o parâmetro Δ_{min} fará com que pontos relativamente distantes sejam descartados, fazendo uma busca mais espalhada no domínio. Por outro lado, valores baixos de Δ_{min} são recomendados para problemas cujo os pontos mínimos estejam em regiões estreitas e ingrimas. Müller e Woodbury (2017) recomenda ainda adotar $\Delta_{min} = 0,001$ para problemas com variáveis contínuas e $\Delta_{min} = 1$ para casos puramente discretos. Esses valores são sugeridos pois, em problemas contínuos, as variáveis podem assumir quaisquer valor do intervalo, já em casos discretos, as variáveis só podem assumir valores separados por ao menos uma unidade de distância entre os valores discretos, seja essa distância um valor unitário, fracionário, entre outros. O processo seguido na Fase II pode ser visto no Algoritmo 3.

Algoritmo 3 Fase II da otimização: melhorar o melhor ponto encontrado.

- 1: Defina \mathbf{x}_{best} como o ponto factível com menor valor da função objetivo;
 - 2: **enquanto** O critérios de parada não for satisfeito **faça**
 - 3: Obtenha os modelos aproximados para as m funções de restrição e defina-as como $s_j(\cdot)$;
 - 4: **se** O domínio de variáveis viáveis definido por Ω e $s_j(\cdot)$ não for vazio **então**
 - 5: Minimize a função objetivo $f(\mathbf{x})$ sujeita as restrições $s_j(\mathbf{x}) \leq 0$ e defina o menor ponto como \mathbf{x}_{temp} ;
 - 6: **se** $\|x_{temp} - \mathbf{x}_k\|_2 < \Delta_{min}$, para qualquer $\mathbf{x}_k \in \mathbf{S}$ **então**
 - 7: Encontre $\mathbf{x}_{new} \in \arg \max\{\Delta(\mathbf{x}, \mathbf{S})\}$ em que $\mathbf{x} \in \Omega$;
 - 8: **senão**
 - 9: Faça $\mathbf{x}_{new} = \mathbf{x}_{temp}$;
 - 10: **fim se**
 - 11: **senão**
 - 12: Encontre $\mathbf{x}_{new} \in \arg \max\{\Delta(\mathbf{x}, \mathbf{S})\}$ em que $\mathbf{x} \in \Omega$;
 - 13: **fim se**
 - 14: Avalie o ponto \mathbf{x}_{new} nas funções de restrições caras;
 - 15: **se** $c_j(\mathbf{x}_{new}) \leq 0$ para todo $j = 1, \dots, m$ e $f(\mathbf{x}_{new}) < f_{best}$ **então**
 - 16: Faça $\mathbf{x}_{best} = \mathbf{x}_{new}$ e $f_{best} = f(\mathbf{x}_{new})$;
 - 17: **fim se**
 - 18: Atualize $n = n + 1$, $\mathbf{S} = \mathbf{S} \cup \mathbf{x}_{new}$;
 - 19: **fim enquanto**
 - 20: Retorne a melhor solução encontrada $(\mathbf{x}_{best}, f_{best})$.
-

O algoritmo *GOSAC* é definido como um método heurístico capaz de escapar de pontos de mínimos locais e tem capacidade de convergência similar ao algoritmo de busca aleatória. Estima-se que o método converge para o mínimo global quando $n \rightarrow \infty$, todavia, espera-se que a convergência ocorra com um número baixo, na ordem de algumas centenas, de pontos de análise, uma vez que cada avaliação das funções de restrição têm

custo computacional extremamente elevado.

Apesar de apresentar uma boa solução para problemas com restrições caras e de possibilitar a escapada de mínimos locais, o *GOSAC* pode apresentar resultados ligeiramente divergentes do mínimo global do problema, dependendo do comportamento desse problema. O fato de se trabalhar com a melhoria do melhor ponto factível torna o algoritmo pouco capaz de explorar regiões mais promissoras do domínio, dificultando a avaliação em regiões factíveis isoladas no domínio. O método proposto nesse trabalho difere dos demais por utilizar uma metodologia de refinamento do domínio voltada para as funções de restrição. Com o objetivo de melhorar a busca global do algoritmo *GOSAC*, nesse trabalho, propõe-se realizar primeiramente uma melhoria da representação das funções de restrição. Essa alteração possibilita que o algoritmo encontre, de forma mais fácil, regiões factíveis do domínio em pontos isolados. Diferente do *GOSAC*, a metodologia aqui proposta se utiliza de uma população pequena para realizar a busca por regiões factíveis no domínio. Essa técnica foi implementada com o objetivo de tornar mais eficiente o processo de busca por regiões promissoras.

Outra abordagem utilizada que difere o algoritmo proposto é a alteração gradual dos limites de caixa do problema. Aproveitando-se do fato da função objetivo ter baixo custo computacional, estima-se, primeiramente, onde estão localizados os mínimos locais e global dessa função, não levando em consideração quaisquer restrições, exceto os limites de caixa das variáveis de projeto. Essa estimativa permite identificar qual região merece mais atenção no que diz respeito ao refinamento das funções de restrição. Dessa forma, sempre que um ponto factível for encontrado e estiver mais próximo de um mínimo local, as restrições de caixa são atualizadas, passando a confinar o domínio de busca. Esse confinamento evita que regiões distantes do possível ponto de mínimo sejam refinadas pois a probabilidade de que esses pontos contenham um ponto de mínimo é baixa. Um detalhamento sobre o método proposto é apresentado no capítulo 5.

5 METODOLOGIA PROPOSTA - ESTRUTURA

Neste capítulo é apresentada a estruturação lógica do processo de otimização com a metodologia proposta nesse trabalho.

5.1 ESTRUTURA PRINCIPAL

Inspirada no método de otimização *GOSAC*, proposto por Müller e Woodbury (2017), a metodologia desenvolvida nesse trabalho tem por objetivo determinar o ponto ótimo de problemas de otimização que demandem um alto tempo de resolução. No modelo de inspiração, a otimização é realizada através de duas etapas, uma busca por regiões factíveis no domínio e, em seguida, o refinamento do melhor ponto encontrado. Essa abordagem demonstra boa consistência e precisão para determinar ótimos globais em problemas com poucas regiões factíveis ou bem definida. Contudo, em se tratado de problemas com múltiplas regiões factíveis, espaçadas no domínio do problema, e bem concentradas, a determinação do ótimo global tende a ser menos consistente. Em outras palavras, notou-se que o método tem uma resistência em sair de uma região factível e ir para outra, e ainda assim, essa nova região pode ser ainda pior que a inicial, desperdiçando esforço de busca.

Nesse sentido, a metodologia aqui desenvolvida tenta direcionar a busca por regiões factíveis realizada na primeira etapa do *GOSAC*. Essa nova metodologia é destinada a problemas em que a função custo tenha um tempo de avaliação muito pequeno e as restrições sejam significativamente demoradas. Entendendo-se que, um tempo de avaliação pequeno é aquele em que uma realização demanda alguns milissegundos enquanto que avaliações demoradas podem chegar a demandar minutos ou dias por realização.

Ambos os modelos foram desenvolvidos para a realização de otimização em problemas com funções denominadas caixa preta (*Black-box*). Essas funções têm como característica o retorno de um valor, dado um conjunto de parâmetros de entrada, sem retornar qualquer outra informação quanto a influência desses parâmetros no resultado (SHAN; WANG, 2010). Uma vez que cada avaliação da função de restrição demandar um elevado tempo computacional, algoritmos não pensados para problemas com esse tipo de função podem acabar demandando muito tempo para chegar a um bom resultado.

A estrutura principal da metodologia proposta (Figura 12) esta dividida em três etapas. A primeira etapa consistem na geração dos pontos iniciais e na determinação dos pontos de referência. Em seguida, na etapa iterativa, realiza-se a busca global por regiões factíveis, confinando sempre que possível a área de busca por novas regiões. Por fim, encontrada uma região promissora para a existência do mínimo global, realiza-se a otimização local, tentando refinar cada vez mais o melhor ponto. Essas etapas são descritas a seguir.

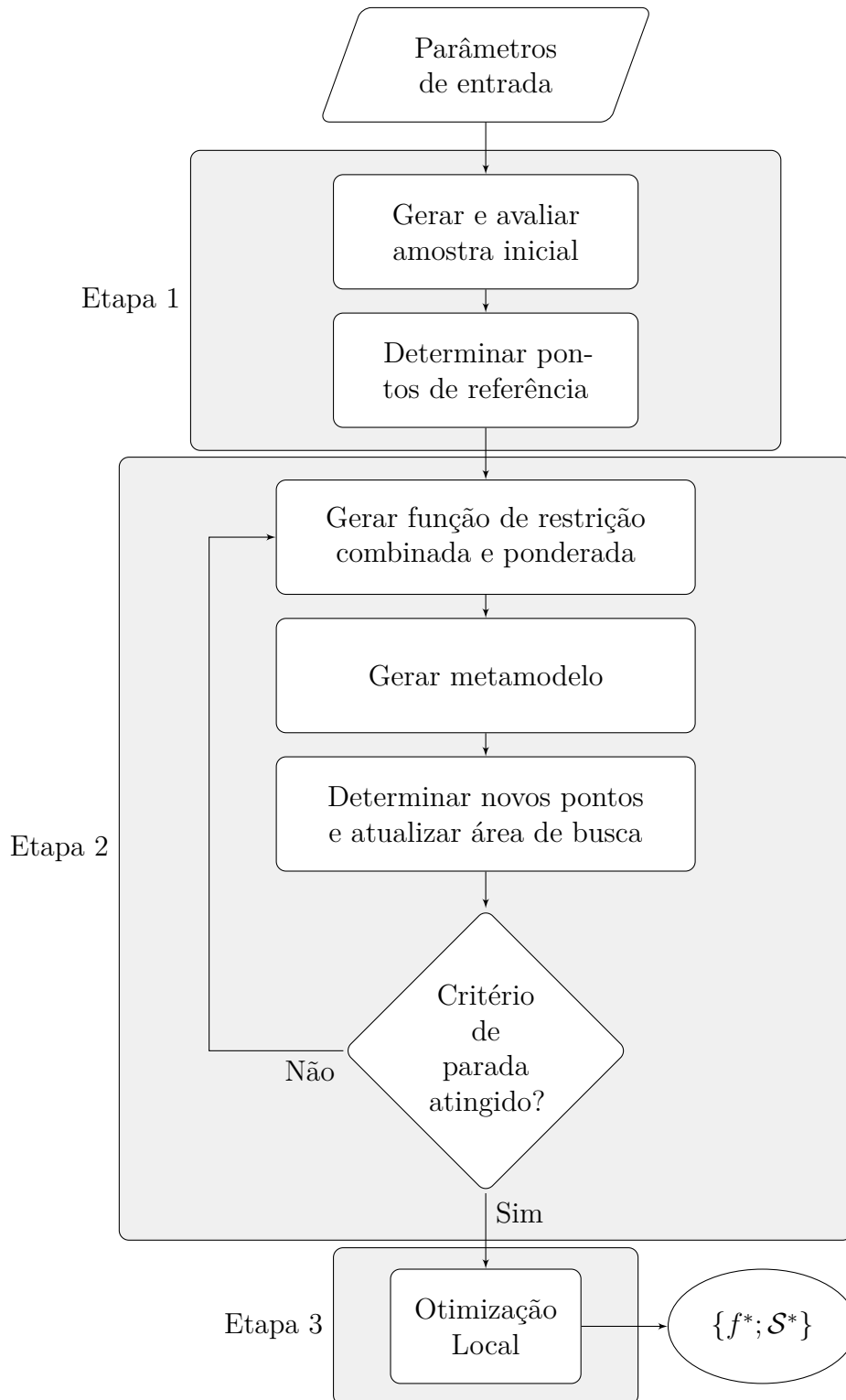


Figura 12 – Fluxograma Estrutura principal.

5.2 ETAPA 1

A etapa 1 da metodologia de otimização apresentada nesse trabalho consiste na geração da amostra inicial e determinação dos pontos de referência para a convergência a área de busca. Inicialmente, dados os parâmetros de entrada, determina-se a quantidade de

pontos amostrais que serão utilizados como partida no processo de otimização. A escolha da quantidade de pontos iniciais (n_i) é critério do usuário, todavia, seguindo o modelo de otimização que inspirou esse trabalho, recomenda-se utilizar $n_i = 2(m + 1)$. Esses pontos então, com auxílio de algum método de simulação de amostras, como, por exemplo o hipercubo latino otimizado (MORRIS; MITCHELL, 1995), devem ser distribuídos em todo o domínio do problema.

Definida a amostra inicial, deverão ser determinados ainda os pontos de referência para a determinação da área de busca por regiões factíveis. Esses pontos são definidos como sendo o mínimo global e os mínimos locais (quando houver) da função custo do problema sujeita apenas as restrições delimitadas pelos valores máximos e mínimos que cada variável de projeto esta sujeita. Uma vez que a função custo do problema tem baixo tempo de análise, qualquer método de otimização pode ser aplicado para determinar esses pontos ótimos, como *Probabilistic Restart* (LUERSEN; LE RICHE, 2004), *Genetic Algorithm* (GOLDBERG, 1989), entre outros. Cada ponto deve, então, ser avaliado quanto ao seu valor na função custo, formando o vetor \mathbf{y} , e nas funções de restrição computacionalmente onerosas, formando \mathbf{C} .

Os pontos ótimos, encontrados pelo método de otimização escolhido pelo usuário, devem ser ordenados de forma crescente em relação ao valor da função custo do problema. Essa ordem servirá para determinar o peso que cada ponto de referência terá para a inserção de novos pontos na sua respectiva região de busca. Pontos de referência com valores mais baixos na função custo terão um peso maior, indicando uma maior preferência na inserção de pontos nas regiões formadas por eles. O oposto acontece com pontos de referência cujo valor na função custo seja mais elevado. O número de pontos inseridos em cada rodada é determinado pelo usuário. Todavia, essa quantidade não contabiliza os pontos padrões que serão citados na etapa 2. Por fim, todos os pontos avaliados deverão, então, ser armazenados para as próximas etapas da metodologia. O Algoritmo 4 apresenta a estrutura sequencial lógica dessa primeira etapa.

Algoritmo 4 Etapa 1 - Pré-iteração

- 1: Defina uma amostra inicial $\mathcal{S} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{n_i}\}$
 - 2: Faça $\mathcal{S}_{Ref}(i) =$ pontos ótimos da função custo do problema sem restrições.
 - 3: Ordene $\mathcal{S}_{Ref}(i)$ de forma crescente em relação ao seu valor na função custo.
 - 4: Defina um peso, entre 0 e 1, para cada ponto \mathcal{S}_{Ref} , com 1 sendo aplicado para o ponto de referência com menor valor e 0 para o ponto com maior valor.
 - 5: Adicione os novos pontos a amostra inicial.
 - 6: Faça $\mathbf{y} = \{f(\mathcal{S})\}$ e $\mathbf{C} = \{g_1(\mathcal{S}) \dots g_{nc}(\mathcal{S})\}$ em que nc é o número de restrições
-

5.3 ETAPA 2

Nesta etapa, a busca por áreas factíveis no domínio do problema é feita de forma global, buscando identificar essas áreas para concentrar a busca pelo ponto ótimo, uma

vez que cada avaliação do problema demanda um custo alto de tempo e/ou esforço computacional. Como o comportamento geral das funções que restringem o problema não é conhecido, um modelo aproximado, gerado a partir dos pontos analisados na etapa anterior, é gerado para representar esse comportamento. Sendo assim, toda e qualquer aplicação de métodos de otimização será aplicada a essa função aproximada que possui um menor custo de análise.

Uma vez que a busca por regiões factíveis nessa etapa tem apenas o objetivo de determinar essas regiões, sem necessariamente encontrar o ponto de mínimo local nessa região, o modelo aproximado só precisa ser capaz de identificar essas regiões. Em virtude disso, e visando diminuir o tempo de análise por iteração, o modelo aproximado gerado é único para a combinação de todas as funções de restrição do problema. Dessa forma, avalia-se, para cada ponto de análise, qual o maior valor entre todas as funções de restrição. Embora essa técnica produza vários pontos de descontinuidade na função a ser aproximada, o que pode prejudicar uma boa representação do modelo aproximado, o objetivo aqui não é essa boa aproximação mas sim as regiões factíveis.

Durante o processo iterativo, algumas regiões mostram-se mais promissoras a conterem o mínimo global do problema. Em virtude disso, a busca por áreas factíveis em determinadas regiões se mostra menos interessante e até inviáveis, uma vez que analisar pontos naquelas regiões só tornaria o processo mais custoso. Sendo assim, limitar a inserção de novos pontos de análise em regiões promissoras à existência do mínimo global é uma outra técnica usada na metodologia apresentada nesse trabalho. Nessa técnica, utiliza-se a norma infinda da distância, delimitada pelas restrições de cada variável, entre o melhor ponto encontrado dentre todos os analisados e o(s) ponto(s) de referência encontrado(s) na etapa 1. Uma vez que se sabe o ponto ótimo da função custo sem restrições, determinado na etapa 1, esse será o ponto ótimo do problema, caso esteja em uma região fatível. Adquirir conhecimento do comportamento das funções de restrição nas proximidades desse ponto é uma característica que pode levar a solução do problema de forma mais acelerada.

Para tentar forçar novos pontos próximas a esse ótimo utópico, aplica-se uma função de ponderação na combinação de restrições do problema. Essa ponderação tem como objetivo acentuar as áreas factíveis mais próximas do ótimo da função custo irrestrita.

O processo iterativo tem fim quando não se consegue mais inserir pontos na região de busca (em função do seu tamanho e do número de pontos já inserido nessa área) ou quando o número de pontos analisados atinge o máximo aceitável pelo usuário. O Algoritmo 5 apresenta a estrutura lógica da etapa 2.

5.4 ETAPA 3

Assim como a etapa 2, a etapa 3 consiste em um processo iterativo. Contudo, nessa etapa é realizada uma busca refinada em procura do ponto ótimo do problema. A aproximação individual de cada função de restrição por um modelo aproximado dedicado,

Algoritmo 5 Etapa 2 - Iteração (Busca Global)

```

1: enquanto Critérios de parada não forem atingidas faça
2:   para  $i_{Ref} = 1 \leftarrow 1$  até  $N^\circ$  de  $\mathcal{S}_{Ref}$  faça
3:     Determine  $\mathbf{C}_{max} = \max(\{g_1(\mathcal{S}) \dots g_{nc}(\mathcal{S})\})$ 
4:     Faça  $\hat{y} = \mathbf{C}_{max} \cdot \left| 1 - \frac{f(\mathcal{S}) - f_{min}}{f_{max} - f_{min}} \right|$ 
5:     Gere um metamodelo  $s(\mathbf{x})$  para a função  $\hat{y}$ 
6:     se  $\exists \mathbf{x} \in \mathcal{S} \mid \{g_1(\mathbf{x}) \leq 0 \ \& \ \dots \ \& \ g_{nc}(\mathbf{x}) \leq 0\}$  então
7:       Faça  $\mathbf{xFac}_{best}$  = Melhor ponto factível em  $\mathcal{S}$ 
8:       Atualize a região de busca
9:     fim se
10:    Determine o ponto  $x_s$  que minimize a função objetivo  $f(\mathcal{S})$  sujeita a  $s(\mathcal{S}) < 0$ 
    e  $x_j^l \leq x_j \leq x_j^u, i = 1, \dots, m$ 
11:    Determine o ponto  $x_{EI}$  com máxima esperança de melhoria(EI)
12:    Determine  $x_{Points}$  pontos aleatórios adicionais a serem inseridos
13:    Faça  $x_{new} = \{x_s, x_{EI}, x_{Points}\}$  e  $n_{new}$  = número de pontos novos
14:    Remova os pontos em que  $|x_{new}(k) - X| < \Delta_{global} \cdot (x^u - x^l)$  com  $k = 1, \dots, n_{new}$ 
15:    Avalie os novos pontos quanto as restrições originais
16:    Adicione os novos pontos a  $\mathcal{S}$ 
17:    Faça  $\mathbf{xFac}_{new}$  Melhor ponto factível em  $\mathcal{S}$ 
18:    se  $\mathbf{xFac}_{best} = \mathbf{xFac}_{new}$  então
19:      Faça  $Dist(i)$  = Distância entre  $\mathcal{S}_{Ref}(i)$  e  $\mathbf{xFac}_{best}$  com  $i = 1, \dots, n_{\mathcal{S}_{Ref}}$ 
20:      Faça  $\mathcal{S}_{Ref}(i) = \mathcal{S}_{Ref}(i) + Dist(i) \cdot rand(1, m) \cdot passe$  com  $i = 1, \dots, n_{\mathcal{S}_{Ref}}$ 
21:    fim se
22:    Verifique os critérios de parada
23:  fim para
24: fim enquanto

```

ou seja, definido apenas para ela, torna a busca mais precisa. Isso implica ainda em um tempo computacional maior, uma vez que a geração de cada modelo aproximado passa por um processo de otimização único. Em virtude disso, esse é um processo com poucas iterações, visando apenas encontrar um ponto inicial e melhora-lo gradativamente se possível. Esse critério é definido pelo usuário, informando o número de iterações ($nInt_{Local}$) que ele deseja. Ainda nesse caso, um novo limite de aceitação de proximidade entre pontos pode ser definido. Esse novo limite é dado por Δ_{local} que pode ou não ser igual ao Δ_{global} a depender da precisão desejada pelo usuário.

Observe que, como, nessa etapa, busca-se encontrar uma solução mais próxima do problema real, não aplica-se nenhuma ponderação aos valores dos pontos analisados, como era feito na etapa 2. O Algoritmo 6 apresenta a estrutura lógica da etapa 3 da metodologia proposta nesse trabalho.

Algoritmo 6 Etapa 3 - Iteração (Busca Local)

-
- 1: Faça $f_{feasible} = f(\mathcal{S}) \mid \{g_1(\mathcal{S}) \leq 0 \dots g_{nc}(\mathcal{S}) \leq 0\}$
 - 2: Faça $f_{best} = \min(f_{feasible})$
 - 3: Faça $x_{best} = x \mid f(\mathcal{S}) = f_{best}$
 - 4: Defina $n = 1$
 - 5: **enquanto** $n < nInt_{Local}$ **faça**
 - 6: **para** $i = 1 \leftarrow 1$ **até** nc **faça**
 - 7: Determine $\hat{y} = g_i(\mathcal{S})$
 - 8: Gere um metamodelo $s_i(\mathcal{S})$ para a função \hat{y}
 - 9: **fim para**
 - 10: Determine o ponto x_{new} que minimize a função objetivo $f(\mathcal{S})$ sujeita a $s(\mathcal{S}) < 0$ e $x_j^l \leq x_j \leq x_j^u, i = 1, \dots, m$
 - 11: **se** $|x_{new} - X| \geq \Delta_{local} \cdot (x^u - x^l)$ **então**
 - 12: Adicione o ponto ao conjunto de pontos já analisados
 - 13: **senão**
 - 14: Determine um novo em que $|x_{new} - X| < \Delta_{local} \cdot (x^u - x^l)$
 - 15: **fim se**
 - 16: **se** $f(x_{new}) < f_{best}$ **então**
 - 17: Faça $f_{best} = f(x_{new})$
 - 18: Faça $x_{best} = x_{new}$
 - 19: **fim se**
 - 20: Faça $n = n + 1$
 - 21: **fim enquanto**
 - 22: Retorne $\{f_{best}, x_{best}\}$
-

6 METODOLOGIA PROPOSTA - EXEMPLOS COMENTADOS

Com base na metodologia apresentada com capítulo anterior, este capítulo tem por objetivo apresentar os procedimentos realizados pelo algoritmo de otimização desenvolvido e proposto neste trabalho. Para isso, tomou-se dois exemplos de problemas de otimização de duas dimensões, $m = 2$ (2 variáveis de otimização).

O primeiro exemplo tratará dos passos de execução do algoritmo em um problema mais simples, cujo o mínimo da função custo é único e bem definido. Em seguida, será apresentado um problema ligeiramente mais complexo, em que, embora as funções de restrição não sejam modificadas, a função custo será alterada para uma equação com múltiplos mínimos. Dessa forma, será possível estudar o comportamento do algoritmo em ambas as situações.

6.1 TOY PROBLEM

Proposto por Robert B. Gramacy *et al.* (2015) o *Toy Problem* é um problema de otimização analítico bidimensional com variáveis contínuas. Esse problema tem como função custo, ou função objetivo, a equação $f(\mathbf{x}) = x_1 + x_2$, com \mathbf{x} restrita no intervalo $\mathbf{d} = [0, 1]^2$. Essa função está sujeita a duas funções de restrição não lineares que representam as funções de restrição de caixa preta de um problema de otimização e são dadas por:

$$\begin{aligned} g_1(\mathbf{x}) &= \frac{3}{2} - x_1 - 2x_2 - \frac{1}{2}\sin(2\pi(x_1^2 - 2x_2)) \leq 0 \\ g_2(\mathbf{x}) &= x_1^2 + x_2^2 - \frac{3}{2} \leq 0, \end{aligned} \tag{24}$$

cujo mínimo global encontra-se no ponto $\mathbf{x}^* = \{0, 1954, 0, 4044\}^T$ com o valor de $f^* = 0, 5998$.

Neste exemplo, a função objetivo tem baixo custo computacional e pode ser avaliada ilimitadamente enquanto que as funções de restrição representam funções desconhecidas com alto custo de análise. Dessa forma, deseja-se encontrar o ponto ótimo da função custo com o mínimo número de avaliações das funções de restrição.

6.1.1 Processo de otimização

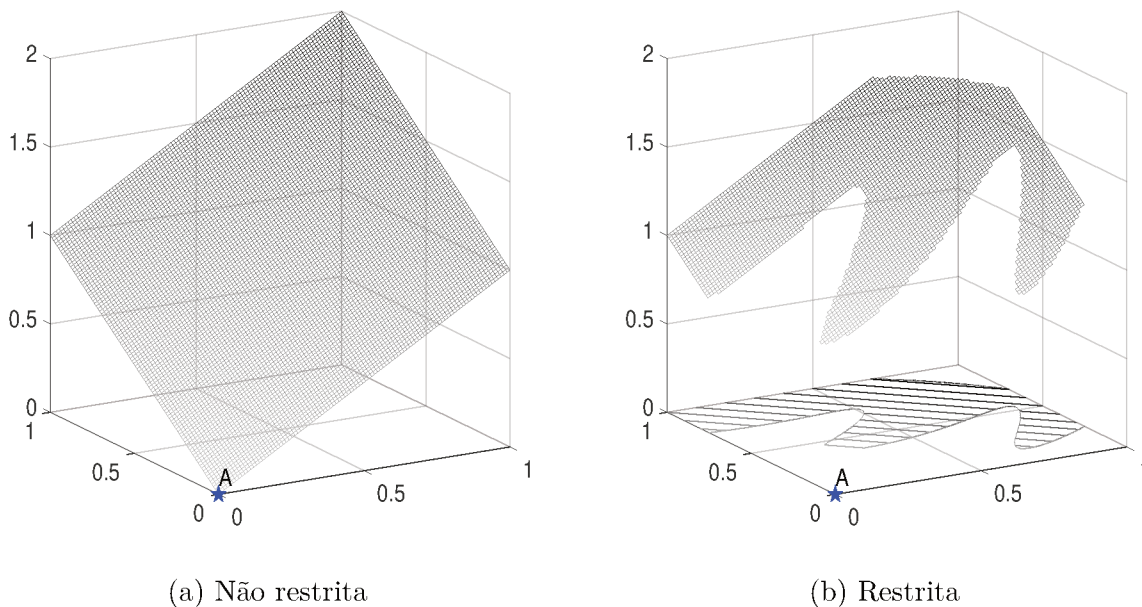
Conforme apresentado no capítulo 5, o processo de otimização será dividido em três etapas. A primeira etapa conta com passos iniciais de preparação para o processo iterativo de determinação das melhores regiões factíveis bem como com a determinação dos primeiros pontos que formaram o primeiro modelo aproximado. Na segunda etapa, busca-se refinar o metamodelo nas proximidades de onde acredita-se estar o mínimo do problema, esse processo funciona de forma similar a uma busca global. Por fim, na terceira etapa, realiza-

se uma otimização da função objetivo sujeita a restrição do modelo aproximado, processo de busca local no mínimo do problema.

6.1.1.1 Etapa 1 - Processo Pré-iterativo

Como citado anteriormente, o problema de otimização aqui estudado possui como característica uma função objetivo com baixo custo computacional e funções de restrição com alto custo de análise. Dessa forma, a avaliação de cada ponto nas funções de restrição deve ser pensada para ser o mais eficiente possível na busca pelo mínimo do problema. Em vista disso, nessa fase será determinado o chamando ponto de referência (n_{ref}). Esse ponto têm por finalidade orientar o sentido da análise da função de restrição, evitando buscas em regiões pouco propícias a não existência de um mínimo da função. Visando explorar a característica da rápida análise de um ponto na função objetivo, o ponto de referência adotado será escolhidos como sendo o ponto de mínimo dessa função.

Para determinar o ponto de mínimo da função objetivo, foi utilizado o algoritmo de otimização *Probabilistic Restart*, proposto e demonstrado por Luersen e Le Riche (2004) em seu artigo. A Figura 13 ilustra a função objetivo do *Toy Problem* irrestrita quanto as funções de restrição (Figura 13a) e a mesma função sujeita as condições de restrição (Figura 13b).



(a) Não restrita

(b) Restrita

Figura 13 – Função objetivo e mínimo local - *Toy Problem*.

O ponto A, embora não atenda todas as condições de restrição, mostra o caminho de decrescimento da função objetivo, em virtude disso, esse é o ponto adotado como referência. É válido salientar que, em grande parte dos problemas de otimização, esse ponto é considerado como utópico, uma vez que, se esse atendesse todas condições de restrição nenhum outro ponto no domínio do problema seria melhor de que ele.

Definido o ponto de referência, o passo seguinte é a geração de uma amostra em todo o domínio do problema. Essa amostra servirá para tentar compreender minimamente o comportamento das funções de restrição e auxiliará na construção do primeiro metamodelo no processo iterativo. A quantidade de pontos amostrais gerados foi determinada segundo a equação $ni = 2(m + 1)$, como sugerida por Müller e Woodbury (2017). Neste trabalho, a geração dos pontos aleatórios se deu utilizando o método do *Optimal Latin Hypercube*, proposto por Morris e Mitchell (1995) e que tem por objetivo gerar uma amostra bem distribuída para garantir uma boa representação do domínio do problema. A representação desses pontos aleatórios no *Toy Problem* pode ser vista tanto para a função objetivo irrestrita (Figura 14a) quanto para essa mesma função sujeita às restrições (Figura 14b).

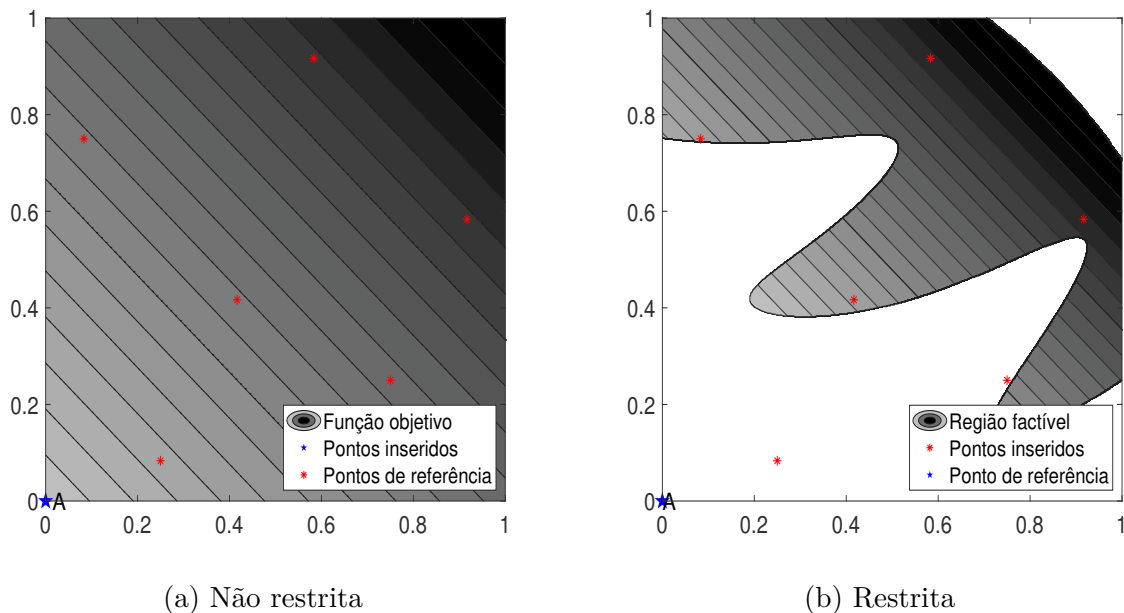


Figura 14 – Função objetivo e pontos iniciais.

O passo final nessa primeira etapa é a avaliação de todos os pontos obtidos na função objetivo e nas funções de restrição dispendiosas. Esses pontos devem ser então avaliados quanto a função objetivo e as funções de restrição de alto custo e, posteriormente, deverão ser armazenados para a próxima etapa do método.

6.1.1.2 Etapa 2 - Processo iterativo

Definidos os pontos iniciais destinados a representar o comportamento do problema, passemos então para a geração de um modelo representativo que substituirá as funções onerosas do problema. Geraremos então um modelo aproximado único para a combinação de todas as restrições. Para isso, precisaremos selecionar o maior valor de todas as restrições em cada ponto avaliado. Para exemplificar esse processo, a Figura 15a ilustra a determinação desses valores máximos para um problema unidimensional com duas restrições. Visando aumentar a probabilidade de inserção de pontos nas regiões mais

próximas ao ponto de referência, será aplicado uma ponderação na função combinada. Essa ponderação é aplicada a partir de uma função que depende do valor da função objetivo avaliado ponto e dos valores máximo e mínimo de todos os pontos no interior da região de busca avaliados na mesma equação. A função de restrições combinada com ponderação é então dada por:

$$\hat{y}(i) = C_{max}(i) \cdot \left| 1 - \frac{f(\mathbf{x}_i) - f_{min}}{f_{max} - f_{min}} \right|. \quad (25)$$

Em que:

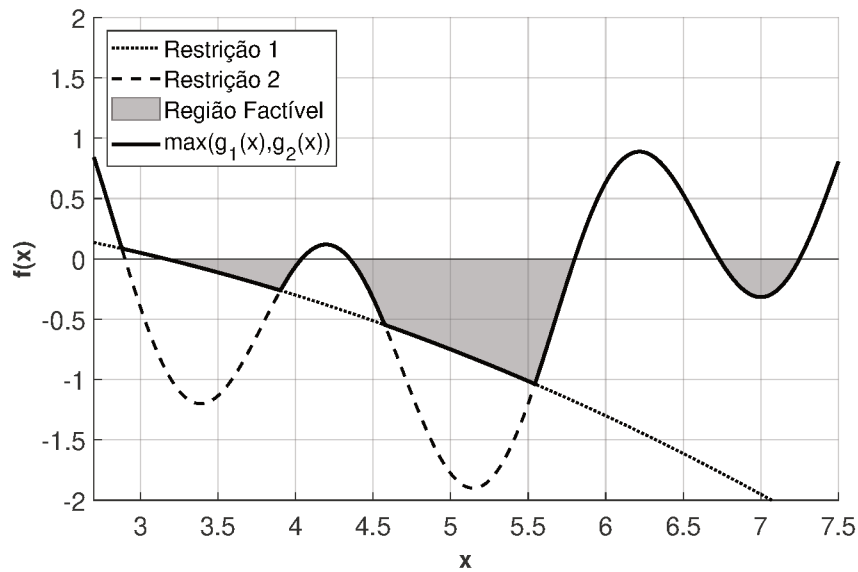
- $\hat{y}(i)$ é o valor do modelo aproximado no ponto i ;
- $C_{max}(i)$ é o maior valor em um dado ponto i comparando todas as restrições;
- $f(\mathbf{x}_i)$ é o valor da função objetivo no ponto i ;
- f_{max} é o valor máximo da função objetivo encontrado dentre todos os pontos dentro na região de busca;
- f_{min} é o valor mínimo da função objetivo encontrado dentre todos os pontos dentro na região de busca.

A função de restrição combinada ponderada pode ser vista então na Figura 15b. Nessa figura, temos a função que deverá ser representada com auxílio de um metamodelo e o comportamento da função objetivo do problema. É possível observar então o comportamento da função objetivo bem como a direção do seu ponto de mínimo. Nota-se assim que, quanto mais próximo o valor do ponto avaliado estiver do valor de mínimo global da função objetivo, menor a penalização desse ponto. O inverso acontece quando o valor esta distante do valor de mínimo geral.

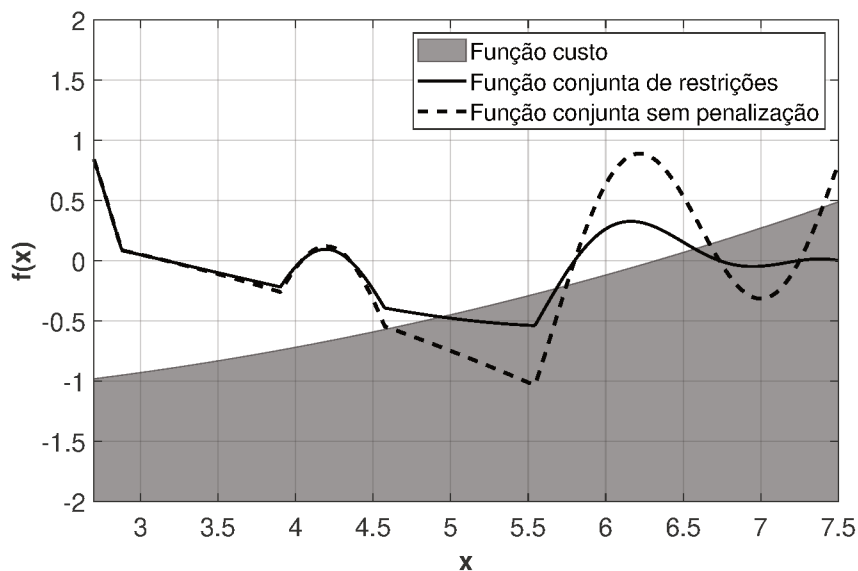
Definido a função de restrição a ser aproximada, o próximo passo é entender o comportamento do processo iterativo. Nesta etapa do processo de otimização, o espaço de busca pelo ponto ótimo poderá ser reduzido conforme se adquire conhecimento sobre o comportamento do problema. Esse espaço pode ser chamado de *região de busca*. Contudo, antes de definir uma região de busca, faz-se necessário entender o conceito de *região factível*, usado na apresentação dessa região.

Müller e Woodbury (2017) define um ponto factível como sendo um ponto do domínio que satisfaz todas as condições de restrição do problema. Expandindo isso para um conjunto de pontos temos que, uma região factível no domínio de um problema de otimização consiste em uma determinada área onde qualquer ponto inserido em seu interior atenderá as condições de restrição.

Uma região de busca, por sua vez, consiste em uma área de exploração por regiões factíveis próximas ao ponto de referência adotado pelo algoritmo. Nesse trabalho, essa



(a) Máximo do exemplo com duas restrições.



(b) Função de restrição combinada com ponderação.

Figura 15 – Exemplo unidimensional com duas restrições.

área é determinada a partir da norma infinita da distância entre o melhor ponto na região factível e o ponto de referência analisado. A região é limitada ainda pelas restrições de caixa reais do problema, evitando a busca em regiões fora do domínio do problema. Na Figura 16 vemos a representação de uma região de busca, área retangular hachurada. Nessa imagem, o ponto A representa o ponto de referência da região e o ponto B, mais extremo, é o melhor ponto encontrado até o momento em região factível. Na Figura 16a, a região de busca é delimitada apenas pela norma infinita da distancia, formando um hipercubo centrado no ponto de referência. Já na Figura 16b, a região de busca esta limitada tanto pela norma infinita da distância quanto pelos limites de caixa reais do problema, não

permitindo que sejam inseridos pontos fora do domínio real.

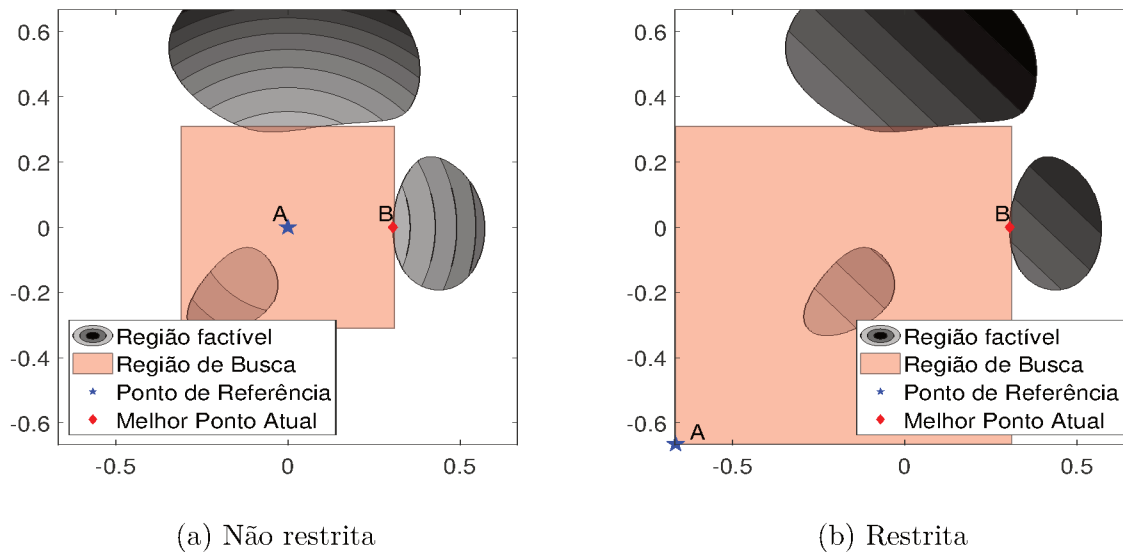


Figura 16 – Exemplo de uma região de busca.

A geração dessa região funciona como uma nova restrição de caixa para o problema, limitando o intervalo onde o algoritmo inserirá novos pontos. Uma quantidade qualquer de pontos pode ser inserida nessa região a fim de se coletar informações sobre o comportamento das restrições nessa área e assim gerar um modelo aproximado dessas funções de restrição.

A inserção de pontos nessa região gerará um crescimento gradativo de conhecimento do comportamento do problema. Todavia, por se tratar de um problema onde cada ponto terá um elevado custo de análise, a inserção apenas de pontos aleatórios nessa região pode não ser uma estratégia muito eficiente. Nesse sentido, devemos utilizar estratégias que visem trazer o máximo de ganho de informação para cada ponto inserido.

Pensando nisso, serão utilizadas três técnicas de inserção de pontos na região de busca. A primeira técnica consiste na utilização do método da máxima esperança de melhoria (*Expected Improvement*). Proposto por Jones, Schonlau e Welch (1998), o *EI*, abreviação de *Expected Improvement*, é uma estratégia muito usada, em virtude da sua facilidade de implementação e performance razoável. Essa metodologia visa encontrar o ponto que oferece maior quantidade de melhoria em relação aos pontos atualmente amostrados (QIN; KLABJAN; RUSSO, 2017). A próxima técnica utilizada consiste em encontrar o ponto ótimo da função objetivo sujeita ao metamodelo gerado com os pontos analisados anteriormente. Essa estratégia tem por objetivo garantir que o menor ponto de uma região de factível foi analisado. Esse ponto pode ser encontrado então por um algoritmo de otimização (Ex.: *Genetic Algorithm* (GOLDBERG, 1989), *Simulated Annealing* (KIRKPATRICK; GELATT; VECCHI, 1983), *Particle Swarm Optimization* (KENNEDY; EBERHART, 1995)). Por fim, a terceira técnica consiste na inserção de pontos por meio

de uma geração aleatória.

Voltando ao *Toy Problem*, nesse exemplo serão inseridos 5 pontos a cada iteração do algoritmo. Desses pontos, um será determinado através do *EI*. Um segundo ponto será determinado pela otimização da função objetivo, sujeita as restrições proporcionadas pelo metamodelo da combinação das funções de restrição. Esse ponto será obtido com o auxílio do *Particle Swarm Optimization (PSO)*. Inspirado na movimentação de voo de aves em revoada, o *PSO* parte de um conjunto de partículas que percorrem o domínio do problema em busca de pontos ótimos. Nesse modelo, cada partícula apresenta um comportamento aleatório em relação ao seu movimento, todavia, sendo influenciado pelas demais partículas da vizinhança, movendo-se globalmente para o melhor ponto encontrado por qualquer partícula.

Os demais pontos serão determinados de forma aleatória com auxílio do *Optimal Latin Hypercube*. Já o metamodelo será gerado a partir da função de restrição combinada ponderada. A Figura 17 apresenta a primeira rodada do processo iterativo. Na primeira imagem (Figura 17a), podemos ver os pontos em que já temos conhecimento dos valores. Vemos ainda os novos pontos inseridos na região de busca do problema, delimitado pela região retangular hachurada. Temos ainda o melhor ponto factível e o ponto de referência, usados para a geração da região de busca. Por fim, podemos notar também a região factível, formada pelo metamodelo gerado pela função conjunta das restrições. Note que, quando comparada a região real (Figura 17b), essa região nos parece pouco representativa, todavia, isso reflete o grau de conhecimento que temos até essa iteração. Nessa chamada iterativa, iremos inserir 5 pontos na região de busca. Esses pontos precisam então ser avaliados quanto as restrições onerosas do nosso problema.

A inserção dos pontos esta sujeita ainda a uma condição. Como pontos muito próximos podem não trazer um ganho efetivo de informação, e que cada ponto inserido tem um elevado custo de análise, qualquer ponto inserido aqui deve respeitar uma distância da vizinhança. Essa distancia mínima é determinada a partir de de um fator Δ_{global} multiplicado pelo intervalo que cara variável de projeto pode ter, denominado *range*. No nosso exemplo, $\Delta_{global} = 0.1$ e $range = [1, 1]$ uma vez que cada variável de projeto pode variar de 0 a 1. Note que $0 < \Delta_{global} < 1$, uma vez que esse parâmetro representa o percentual de distância em relação ao domínio total do problema.

Avaliados os novos pontos no domínio, devemos então gerar um novo modelo de aproximação das funções de restrição do problema. Esse modelo é gerado a partir dos pontos encontrados no interior da região de busca. A eliminação dos pontos fora dessa região tem por finalidade minimizar o tempo de geração do metamodelo. É válido observar que o objetivo principal nessa etapa refinar a restrições em uma região mais próxima ao ponto de referência, tentando encontrar regiões factíveis no entorno desse ponto. Logo, deseja-se, com esse processo iterativo, diminuir a região de busca por novas áreas. Note que, essa redução de área de busca, atrelada a geração de um único modelo para diversas

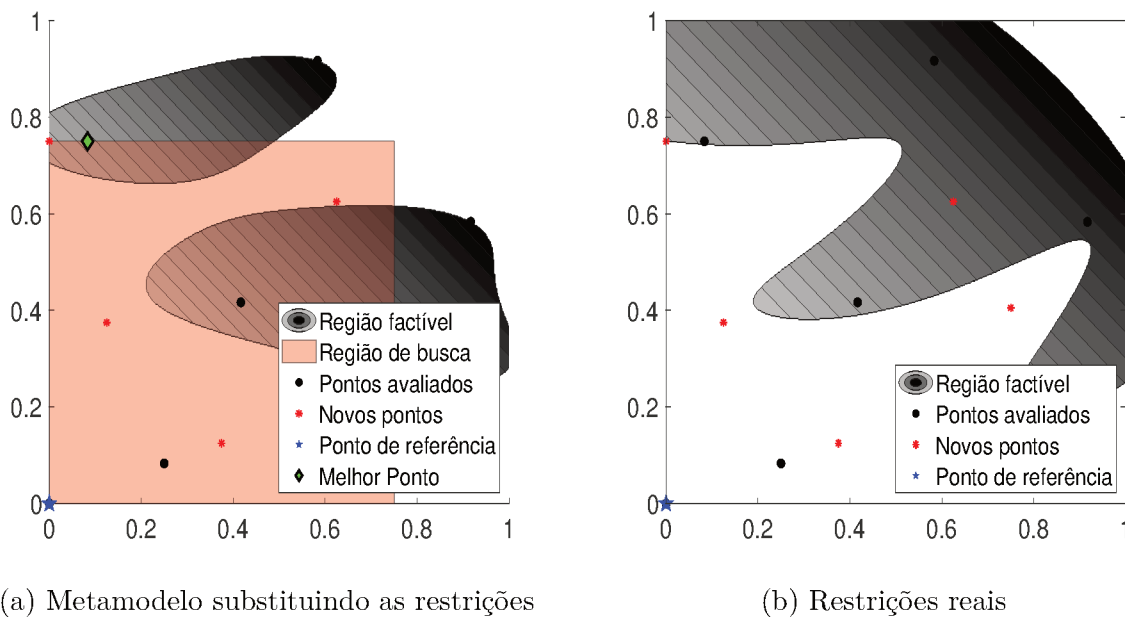


Figura 17 – Iteração 1 - Representação com os diferentes modelos de restrição

restrições, pode resultar na determinação de mínimos locais do problema. Todavia, os mínimos locais encontrados nessa etapa servirão para identificar onde estão as regiões factíveis enquanto que a determinação do mínimo global só será feita na etapa 3.

A próxima iteração, então, será realizada com um novo modelo aproximado das funções de restrição do problema real (Figura 18). A região de busca, nesse passo, foi reduzida em virtude do encontro de um novo ponto factível mais próximo ao ponto de referência. Esse processo se repete até que a condição de pausa do processo iterativo seja atingida.

Apesar de fornecer uma melhor concentração de busca pelo ponto ótimo do problema, a diminuição da região de busca só será feita se for encontrado um novo ponto de valor menor mais próximo ao ponto de referência. Em problemas de otimização cujo as regiões factíveis encontram-se próximas ao mínimo global (ponto de referência) da função custo essa característica se mostra interessante, pois estamos convergindo para próximo desse ponto. Todavia, problemas que tenham regiões factíveis completamente opostas a esse mínimo podem acabar não convergindo da forma desejada, pois, nesses casos, a metodologia tentará sempre buscar um ponto mais próximo ao ponto de referência. Para contornar esse problema, podemos pensar em um deslocamento da posição do ponto de referência em direção do melhor mínimo factível encontrado até então. Esse passo (*passo*) pode ser feito de forma aleatória, contudo, com uma distância máxima permitida, para que o ponto de referência não pule instantaneamente para o melhor ponto e interrompa o processo iterativo.

A realização desse passo ocorrerá sempre que o ponto de mínimo não mude entre iterações, ou seja, dada duas iterações, se o ponto de mínimo permanecer o mesmo, é

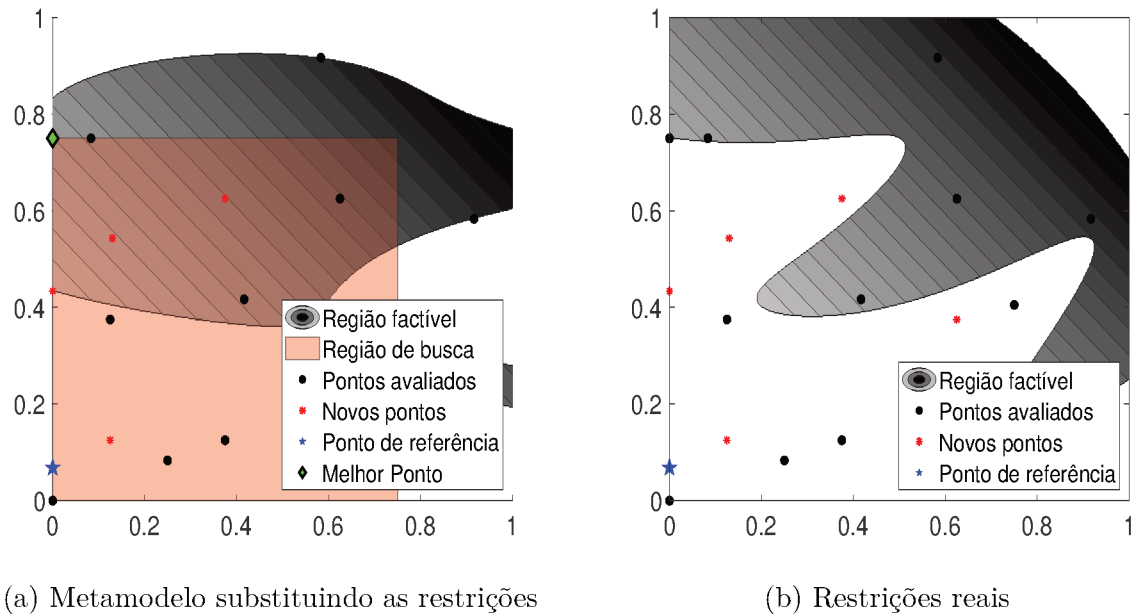


Figura 18 – Iteração 2 - Representação com os diferentes modelos de restrição

realizado um passo na direção desse ponto mínimo. Esse processo tende a fazer a região de busca diminuir na direção da melhor região factível sempre que não houver mudança de ponto factível encontrado. Note que, o *passee* modifica apenas a posição do ponto de referência e não o seu valor, ou seja, essa nova posição não é avaliada como um ponto de análise computacionalmente caro para a solução do problema. Para esse exemplo, foi utilizado um *passee* aleatório com no máximo de 10% da distância entre o ponto de referência e o melhor ponto de mínimo.

A interrupção do processo iterativo caracteriza também o fim da etapa 2 do algoritmo. Essa interrupção pode acontecer por dois motivos. O primeiro motivo está ligado à área da região de busca. Quanto menor a área da região de busca menor a chance de adicionar pontos espaçados o suficiente para se obter um ganho significativo de informação. Em outras palavras, adicionar um ponto que trata informação de uma região factível inexplorada se torna cada vez mais difícil, pois outros pontos já podem ter sido adicionados para explorar aquela região. O segundo motivo de pausa do processo iterativo está relacionado à quantidade de pontos avaliados. Uma vez que o problema conte com restrições cuja a análise seja demorada, é necessário definir um limite máximo de pontos a serem avaliados. Caso contrário, a otimização poderia continuar inserindo pontos infinitamente em busca de um melhor valor.

Nesse exemplo, será utilizado como primeiro critério de parada a área da região de busca dependente do número de pontos dentro dessa região. Para isso, será determinado um valor máximo permitido que um ponto esteja próximo a outro. A área mínima permitida então é dada por $nPoints_{região} \cdot \Delta_{global} \cdot area_x \cdot 2^m$. Com $nPoints_{região}$ sendo o número de pontos na região de busca, Δ_{global} o fator de distância mínima permitida entre dois

pontos, $area_x$ área formada pelo intervalo de cada variável de projeto e 2^m o multiplicador para determinar a área que esse ponto representa. Como segundo critério, adotou-se que não seria permitido um número maior que $100m$ pontos de avaliação, ou seja, 200 pontos avaliados.

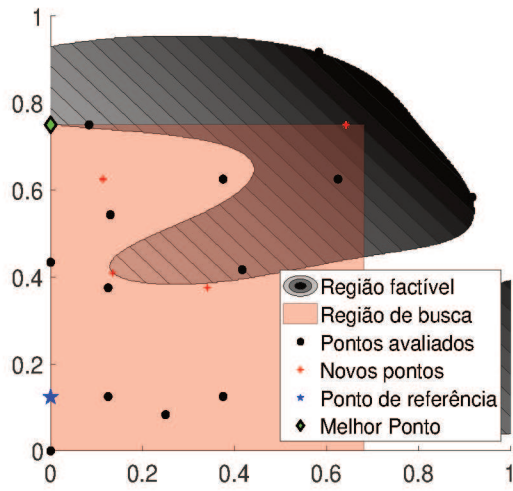
A Figura 19 apresenta os pontos de análise obtidos em algumas iterações desse exemplo, bem como o comportamento do metamodelo que aproxima a função de restrição. Nesse problema, o critério de parada atingido se deu pelo tamanho da região de busca, com 32 pontos avaliados no total. Finalizada então a etapa de busca por regiões factíveis, passemos então para a terceira etapa do processo, a otimização local.

6.1.1.3 Etapa 3 - Otimização local

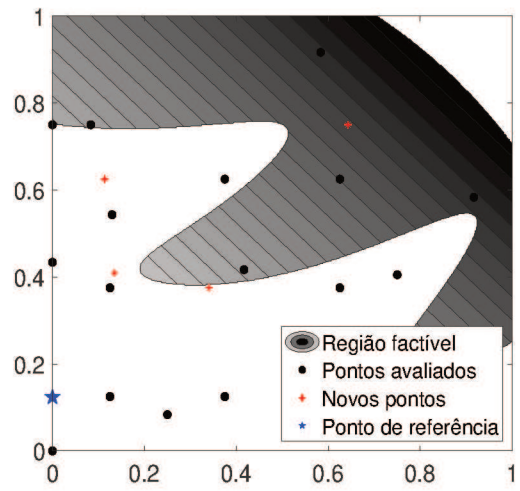
Nessa etapa, já temos algumas das melhores regiões factíveis e as melhores candidatas a conterem o mínimo global do problema, podemos então iniciar o processo de busca por esse ponto ótimo. Para isso, a partir de todos os pontos avaliados até a etapa anterior, vamos gerar um metamodelo para cada uma das funções de restrição reais do problema. Gerar modelos individuais para cada uma dessas funções garante um metamodelo mais fiel o comportamento real do problema, uma vez que diminui a chance de variações bruscas em cruzamento de funções. Para entender melhor esse comportamento tomemos a Figura 15b. Nesse exemplo temos duas funções de restrição distintas que se cruzam em determinados pontos do domínio. O modelo aproximado único a ser aproximado, citado na etapa 2, é representado pela linha contínua. Observe que, nesse caso, no domínio do problema, o modelo sofre algumas descontinuidades. Essas descontinuidades dificultam o processo de aproximação do metamodelo.

Gerados os modelos aproximados para cada uma das funções de restrição, podemos então iniciar o processo de otimização local. Assim como a etapa anterior, esse é um processo iterativo em que será realizada a determinação do melhor ponto obtido a partir da otimização da função objetivo sujeita aos metamodelos gerados para representar as funções de restrição. Para essa otimização, usaremos novamente o algoritmo de otimização *PSO*. A Figura 20 apresenta a primeira iteração dessa etapa. A esquerda (Figura 20a), temos os pontos já conhecidos, o ponto ótimo obtido com a otimização com o *PSO* e a região factível gerada com relação aos metamodelos das funções de restrição. A direita (Figura 20b), temos as mesmas informações, agora, com a região delimitada pelas restrições reais do problema. Note que, o modelo aproximado possui precisão maior na região onde temos maior conhecimento, ou seja, maior quantidade de pontos. As regiões mais distantes desses pontos, por sua vez, possuem menos precisão na aproximação. Esse é um comportamento desejado pois mostra que estamos fazendo poucas análises em regiões com baixa chance de ocorrência do mínimo global e, conseqüentemente, diminuindo o tempo que gastaríamos para otimizar esse problema.

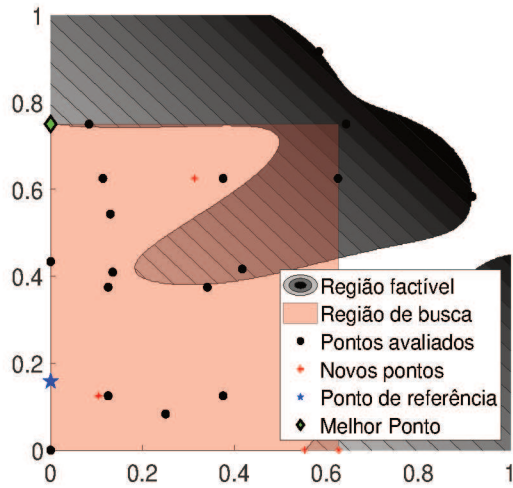
Assim como fizemos na etapa 2, aqui podemos definir uma distancia mínima que



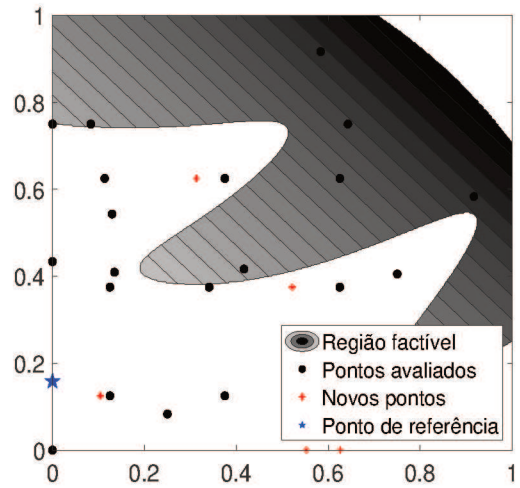
(a) Iteração 4 (Aproximado)



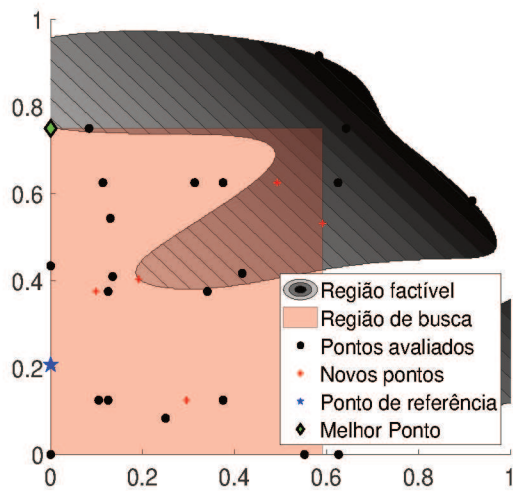
(b) Iteração 4 (Real)



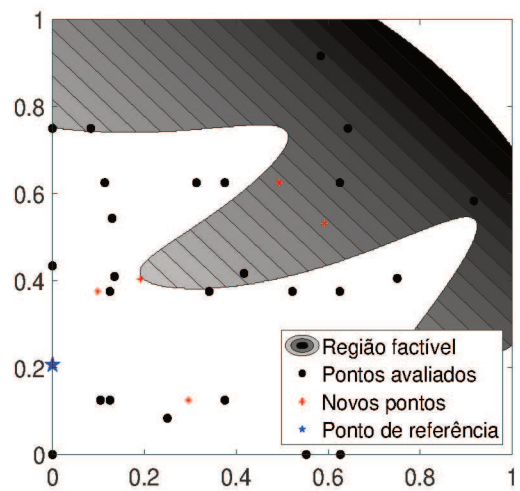
(c) Iteração 5



(d) Iteração 5



(e) Iteração 6



(f) Iteração 6

Figura 19 – Iterações da etapa 2 para o *Toy Problem*.

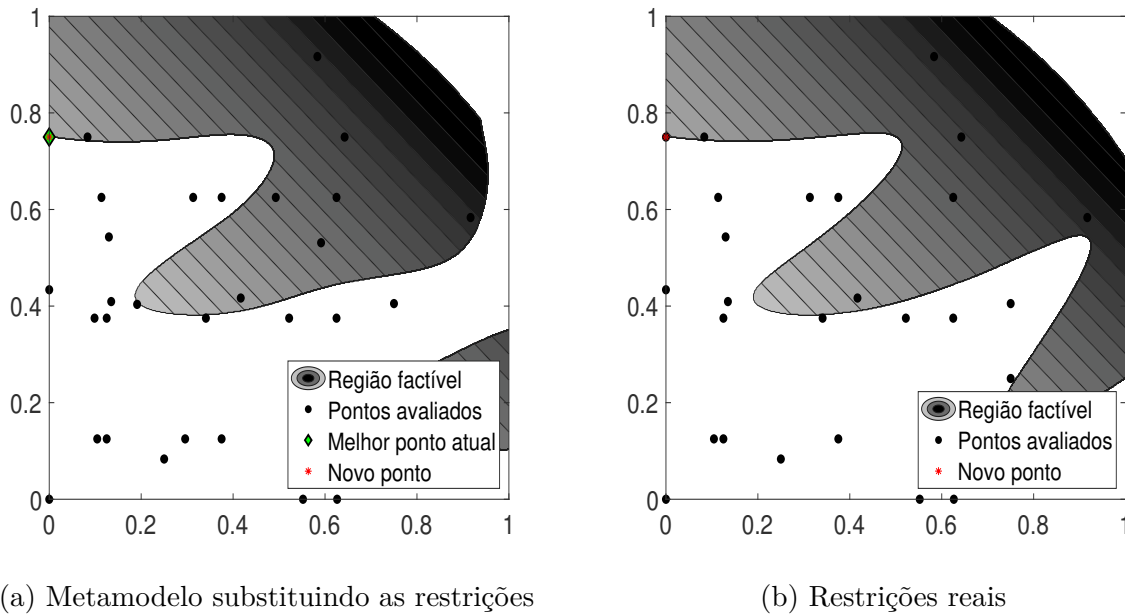
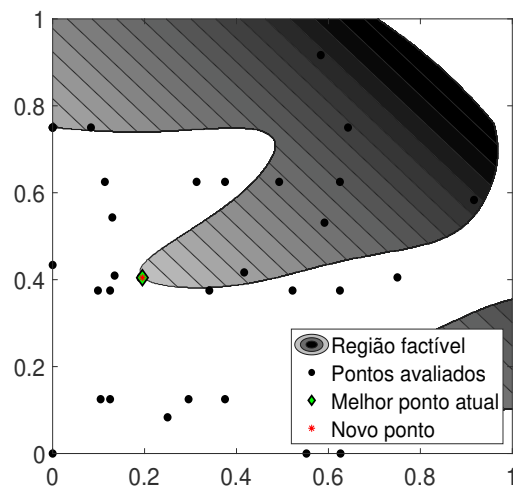


Figura 20 – Iteração 7 - Otimização local

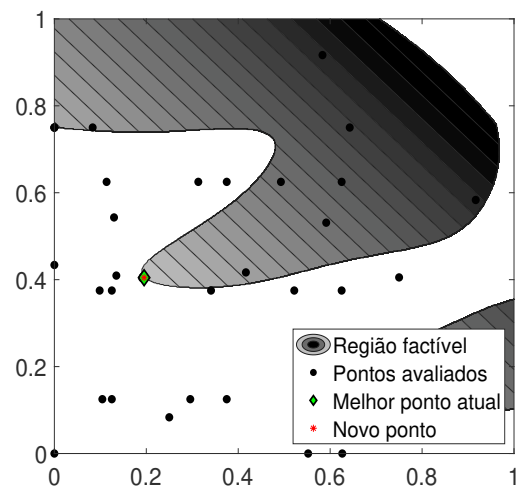
um ponto pode estar do outro, isso vai depender do grau de precisão que aceitamos para o problema. Nesse problema, não definimos distância mínima, permitindo que o algoritmo tente encontrar o melhor ponto que ele conseguir. O critério de parada nessa etapa está ligado apenas ao número de pontos que queremos inserir nessa etapa. Nesse exemplo, foram permitidos a inserção de 10 pontos para refinamento do valor ótimo. A Figura 21 ilustra as últimas 6 iterações do processo de otimização.

Finalizado o processo de otimização, o valor ótimo encontrado pela metodologia sugerida nesse trabalho foi $f(\mathbf{x}) \approx 0.5998$, sendo $\mathbf{x} \approx [0.1954, 0.4044]$. Como mencionado inicialmente, Robert B. Gramacy *et al.* (2015) cita o ponto ótimo do problema como sendo $f(\mathbf{x}) \approx 0.5998$, com $\mathbf{x} \approx [0.1954, 0.4044]$. Logo, nota-se que o algoritmo foi capaz de encontrar esse ponto. Para a resolução desse problema foram realizados 72 pontos de análise.

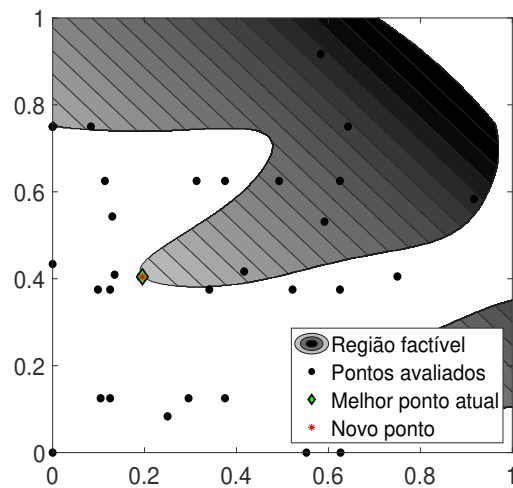
O *Toy Problem* conta com uma característica interessante de ser um problema com alguns mínimos locais cujo os valores são relativamente próximos. Essa característica permite observar o comportamento de um algoritmo de otimização, avaliando como ele está trabalhando para evitar esses mínimos locais do problema. Todavia, nesse problema, a função objetivo possui apenas um ponto de mínimo bem definido ($f(\mathbf{0}) = 0$). Nesse trabalho o ponto de mínimo da função objetivo influencia no comportamento do processo de otimização, pois esse é o ponto de referência que auxilia na convergência do resultado de forma mais eficiente. Precisamos então avaliar o comportamento desse algoritmo em problemas com múltiplos mínimos locais na função objetivo. Pensando nisso, precisaremos de um novo problema de estudo.



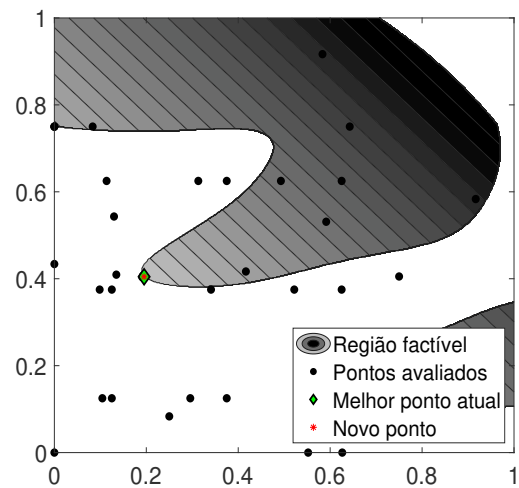
(a) Iteração 11



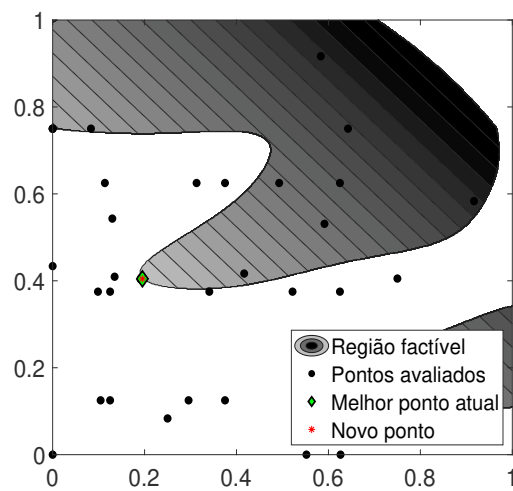
(b) Iteração 12



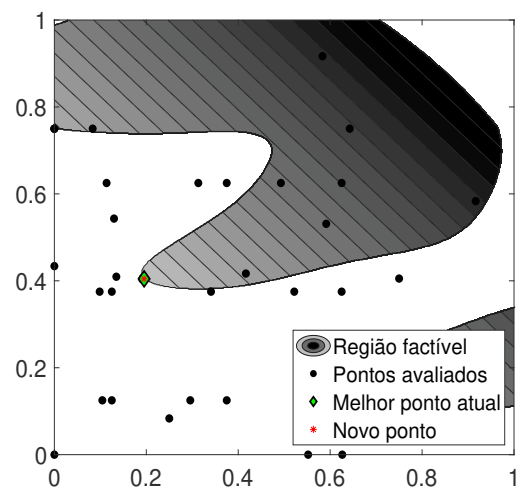
(c) Iteração 13



(d) Iteração 14



(e) Iteração 15



(f) Iteração 16

Figura 21 – Iterações da etapa 3 para o *Toy Problem*.

6.2 TOY PROBLEM MODIFICADO

O problema que estudaremos agora é uma modificação do *Toy Problem* original. Nesse exemplo, substituiremos a função objetivo por:

$$f(\mathbf{x}) = 0.15 + \left(-(w_2 + 47) \sin \left(\sqrt{\left| w_2 + \frac{w_1}{2} + 47 \right|} \right) - w_1 \sin \left(\sqrt{|w_1 - (w_2 + 47)|} \right) \right) / 400, \quad (26)$$

em que:

$$\begin{aligned} w_1 &= 200(-0.5 + x_1) \\ w_2 &= 200(1.5 + x_2). \end{aligned} \quad (27)$$

Com isso, nossa nova função objetivo pode ser vista na Figura 22. Essa função foi criada a partir da função *EggHolder* (WHITLEY *et al.*, 1996), modificada para apresentar um bom comportamento no intervalo de variáveis $0 \leq x_1, x_2 \leq 1$.

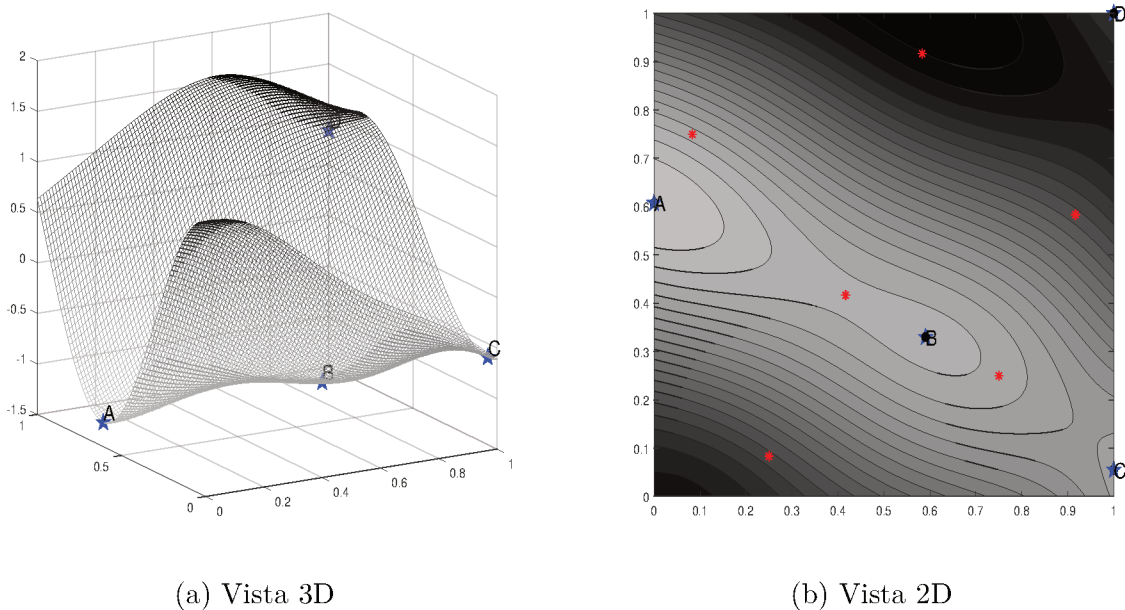


Figura 22 – Função objetivo e mínimo local - *Toy Problem* Modificado.

6.2.0.1 Etapa 1 - Processo Pré-iterativo

Como podemos ver, a função em questão agora possui 4 pontos de mínimo, encontrados, assim como no primeiro exemplo, pelo algoritmo *Probabilistic Restart*. Tal qual esse exemplo ainda, a primeira etapa do processo de otimização continua sendo realizada de forma similar, contudo, com a diferença de encontrarmos múltiplos pontos de referência.

Outra diferença do que foi realizado na primeira etapa do algoritmo, descrito no Capítulo 6.1.1.1, é a comparação e análise desses pontos de referência, uma vez que no exemplo anterior tínhamos apenas 1 ponto de referência. Primeiramente ordenaremos os pontos em ordem crescente em relação aos valores desses na função objetivo. Feito isso, atribuiremos pesos a esses pontos, quanto a magnitude desse valor. Esses pesos serão utilizados no processo iterativo. Por fim, avaliemos ainda esses pontos quanto as restrições, observando se algum desses encontra-se em uma região factível. Observando de forma ordenada pelo valor da função objetivo, se algum desses pontos estiver em uma região factível, podemos descartar, da lista de pontos de referência, os demais pontos com valores superior a ele. Isso será feito pois esses outros pontos já são mínimos locais e não poderiam trazer valores menores que o ponto encontrado na região factível.

Os valores ordenados da função objetivo em cada ponto de referência podem ser vistos na Tabela 1. Nesse exemplo, não nenhum ponto encontra-se em uma região factível, logo, todos os pontos serão utilizados como referência.

Tabela 1 – Pontos de referência para o exemplo *Toy Problem Modificado*.

n_{ref}	\mathbf{x}	$f(\mathbf{x})$	Região factível
A	[6.9515e-08, 0.6074]	-1.2606	não
B	[0.5905, 0.3298]	-0.9133	não
C	[1, 0.0550]	-0.6364	não
D	[1, 1]	0.8425	não

Uma vez definidos as referências, continuemos o processo definindo $n_i = 2(m + 1)$ pontos espalhados pelo domínio utilizando o *Optimal Latin Hypercube*, assim como feito para no exemplo 1 e então passemos então para a próxima etapa.

6.2.0.2 Etapa 2 - Processo iterativo

Como vimos, nessa etapa inserimos pontos em uma região de busca delimitada pelo ponto de referência, o melhor ponto factível encontrado e as restrições limitantes de cada variável do problema. No exemplo anterior, a otimização da função custo sem restrições encontrou apenas 1 ponto de mínimo, usado como ponto de referência e gerando uma única região de busca. Nesse exemplo, por outro lado, a otimização encontrou 4 pontos de mínimo (locais e global), ou seja, 4 pontos de referência foram encontrados, consequentemente, teremos então 4 regiões de buscas. Apesar de serem independentes umas das outras quanto ao seu tamanho, essas regiões podem ter intersecções entre áreas. Em virtude disso, a inserção de um ponto de análise em uma área de intersecção irá influenciar em ambas regiões de busca que compartilhem essa área.

Partindo então para o próximo passo, vamos definir o número de pontos a serem inseridos em cada iteração. No primeiro exemplo havíamos definido o essa quantidade como sendo 5 pontos. Dois desses pontos foram inseridos por meio do *EI* e do mínimo da

função objetivo sujeita ao metamodelo da função de restrição combinada com ponderação. Já os demais pontos foram inseridos por meio de uma amostra gerada pelo *Optimal Latin Hypercube*. Essa estratégia pode continuar sendo utilizada para problemas com múltiplos pontos de referência, entretanto, ao fazermos isso, poderemos estar inserindo como pontos de referência, indivíduos que apenas tornem o processo mais demorado. Isso porque, nesses casos, alguns pontos de referência que estejam fora de uma região factível podem ter valores maiores, em relação a função custo, quando comparados a ótimos locais que estejam dentro de uma região factível.

Para evitar esse esforço computacional, podemos então fazer a inserção com base na ponderação do valor da função objetivo. Dessa forma, regiões delimitadas por pontos de referência com valores de mínimos não muito expressivos terão menos pontos inseridos.

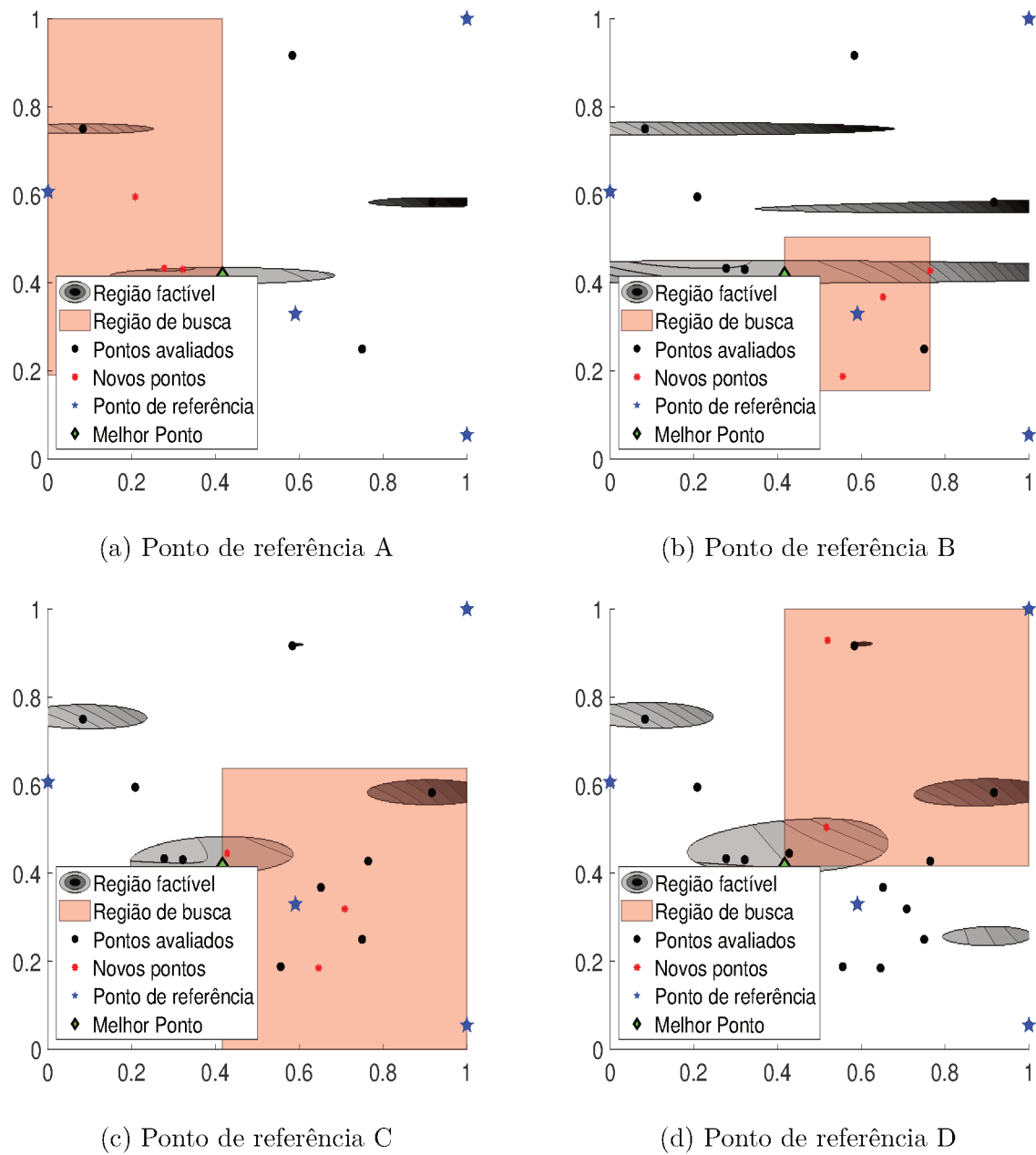
Voltando ao nosso exemplo, adotemos o número de pontos a serem inseridos como sendo $n_{defPoints} = 3$. Esse valor deverá ser multiplicado então pelo peso de cada região de busca e depois arredondado para se determinar a quantidade de pontos inseridos. Na Tabela 2 temos essas quantidades de pontos que serão inseridos em cada região formada pelos pontos de referência. Note que, independente do peso atribuído, ao menos dois pontos estão susceptíveis a ser inseridos em cada região, apresentados entre parênteses. Esses pontos são os obtidos pelo mínimo da função objetivo sujeita ao metamodelo da função de restrição combinada com ponderação e o ponto arbitrado pelo *EI*. Isso não garante que esses pontos obrigatoriamente serão inseridos, pois cada novo ponto deve obedecer o distanciamento mínimo assim como descrito no primeiro exemplo.

Tabela 2 – Número de pontos inseridos em cada iteração - *Toy Problem* Modificado.

N_{ref}	\mathbf{x}	$f(\mathbf{x})$	Peso	$nPoints$
A	[6.9515e-08, 0.6074]	-1.2606	0.394	1 + (2)
B	[0.5905, 0.3298]	-0.9133	0.329	1 + (2)
C	[1, 0.0550]	-0.6364	0.277	1 + (2)
D	[1, 1]	0.8425	0.000	0 + (2)

A Figura 23 apresenta a primeira execução do processo iterativo para cada uma das regiões de busca. Observe que, o processo foi realizado de forma sequencial, da região com menor mínimo para a região com maior mínimo. Dessa forma, os pontos inseridos na região de busca com o menor mínimo interferiu nos modelos aproximados gerados para as próximas regiões.

A cada nova iteração, o melhor valor encontrado é comparado com os valores dos pontos de referência. Uma vez encontrado um ponto melhor que algum ponto de referência, a região de busca gerada por esse ponto será eliminada do processo iterativo. O objetivo disso é eliminar regiões de buscas pouco promissoras. Podemos notar isso na iteração 2 desse exemplo (Figura 24). Nesse caso, duas regiões de busca foram eliminadas do processo pois o melhor ponto encontrado na execução 1 é melhor que os valores dos pontos de referência que formam essas regiões.

Figura 23 – Iteração 1 - *Toy Problem Modificado*.

Esse processo continua até serem atingidos os mesmos critérios de parada definidos no exemplo 1. Para este caso, com vários pontos de referência, vale ressaltar que o critério do tamanho da área de busca encerra o processo iterativo todo o processo iterativo. Ou seja, se ao menos uma das regiões de busca for pequena o suficiente para atingir esse critério, o processo de iteração é pausado para todas as outras.

Nesse exemplo, considerou-se que os pontos de referência não sofrem alterações, logo, $pass = 0$.

Finalizado o processo iterativo da etapa 2, passemos então para a etapa de otimização local. Nessa etapa, os procedimentos realizados são exatamente os mesmos mostrados no exemplo 1. A Figura 25 apresenta o resultado após a última iteração do *Toy Problem*

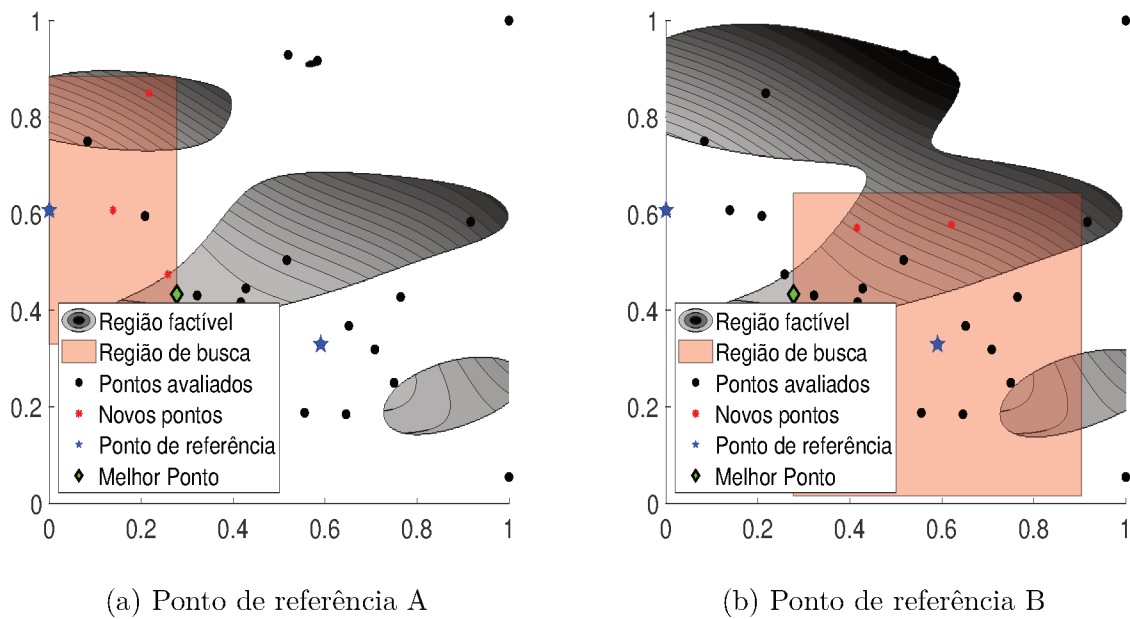


Figura 24 – Iteração 2 - *Toy Problem* Modificado.

Modificado.

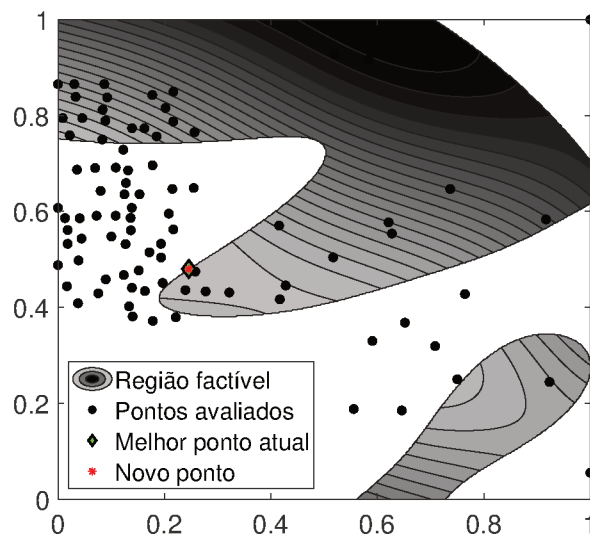


Figura 25 – Iteração 38 - *Toy Problem* Modificado.

O valor ótimo encontrado pela metodologia sugerida nesse trabalho foi $f(\mathbf{x}) \approx -0.9868$, em $\mathbf{x} \approx [0.2456, 0.4801]$. Para esse exemplo foram realizadas 100 avaliações do problema. Para validação desse resultado, foi realizada uma busca exaustiva pelo ponto de mínimo, com 3000 otimizações independentes utilizando o algoritmo de otimização *PSO*. O ponto ótimo obtido com essas otimizações foi $f(\mathbf{x}) \approx -0.9868$, em $\mathbf{x} \approx [0.2457, 0.4801]$.

7 EXPERIMENTOS NUMÉRICOS

Nesse capítulo são apresentados os problemas numéricos utilizados para validar a metodologia apresentada nesse trabalho. Os resultados serão então comparados a valores fornecidos por algoritmos clássicos e consolidados na literatura.

Os problemas 2, 3, 4, 5, 6 e 7 foram desenvolvidos pelo autor desse trabalho e a determinação do melhor ponto para cada um desses problemas foi obtida através de uma busca exaustiva por cada ponto ótimo. Para isso, foram realizadas 5000 otimizações independentes com os algoritmos *AG*, *PSO* e *SGA*. Foram realizadas ainda 1000 otimizações com as metodologias proposta nesse trabalho e o algoritmo *MATSuMoTo*. Os dados utilizados nessa determinação não foram utilizados para a comparação entre os métodos.

Para melhor compreensão e organização, esse capítulo foi dividido em quatro seções: Implementação, problemas numéricos, problemas clássicos da engenharia e problemas de otimização estrutural. Cada seção conta com a descrição dos problemas estudados, comentários e resultados obtidos.

Na seção 7.1 são descritas algumas informações acerca dos métodos e ferramentas utilizadas para implementação dos códigos das metodologias utilizadas nesse trabalho. São apresentadas ainda algumas informações sobre a exposição dos resultados obtidos e o que foi comparado entre as metodologias. Na seção 7.2 são apresentados problemas matemáticos que, em virtude do seu baixo custo de avaliação, possibilitam uma análise rápida do desempenho do algoritmo implementado e da metodologia como um todo. Os problemas apresentados na seção 7.3 foram escolhidos a partir de problemas clássicos da engenharia implementados por diferentes autores, permitindo assim uma validação dos resultados obtidos. Por fim, na seção 7.4, são apresentados dois problemas de otimização estrutural cujo a avaliação de resistência apresente um custo de análise significativo, inviabilizando a execução de varias chamadas no processo de otimização.

7.1 IMPLEMENTAÇÃO

Para a implementação do método optou-se pela linguagem de programação MATLAB[®], também chamada de M-código. Essa nada mais é do que uma linguagem interpretada de alto nível baseada em C++ e programada no software MATLAB[®] (MATLAB, 2018). Optou-se por essa linguagem devido sua facilidade de manipulação e pelo conhecimento anterior do autor e do orientador desse trabalho. Nos problemas estruturais, foi implementada uma rotina de análise estrutural por meio do *MEF*, retornando apenas o descolamento global de cada nó da estrutura e os esforços solicitantes em cada elemento.

O número de execuções independentes realizadas pode ser visto na apresentação de cada problema. O objetivo dessas realizações é de apresentar informações estatísticas para os resultados obtidos para as metodologias aqui utilizadas. Para tentar garantir uma boa consistência no processo de comparação, nas metodologias implementadas pelo autor,

foram utilizadas as mesmas sementes entre as metodologias.

Uma semente (*seed*) nada mais é de que um estado inicial utilizado por um algoritmo matemático de geração de números aleatórios. A definição desse estado inicial permite a obtenção da mesma sequência de sequência numérica sempre que se iniciar a geração dos números aleatórios, logo, os algoritmos de geração são, na verdade, pseudoaleatórios (VIEIRA; SOUZA; RIBEIRO, 2004). Nos exemplos aqui estudados, o controle dessa geração pseudoaleatórios permite uma melhor comparação entre os métodos, pois esses partiram da mesma origem.

7.1.1 Algoritmos utilizados

Diferentes metodologias de otimização foram empregadas com o objetivo de validar o método proposto nesse trabalho. Dentre os algoritmos utilizados estão:

- *Genetic Algorithm* - GA (GOLDBERG, 1989);
- *Particle Swarm Optimization* - PSO (KENNEDY; EBERHART, 1995)
- *Search Group Algorithm* - SGA (GONÇALVES; LOPEZ; MIGUEL, 2015)
- *Global Optimization with Surrogate Approximation of Constraints* - GOSAC (MÜLLER; WOODBURY, 2017)
- MATSuMoTo (MUELLER, 2014);
- *MATLAB Surrogate Model Toolbox* - MATSuMoTo (MUELLER, 2014).

7.1.2 Apresentação dos resultados

Durante o processo de otimização com as metodologias aqui trabalhadas, alguns dados são gerados a partir de amostras aleatórias não controladas pelo usuário. Sendo assim, os resultados finais podem variar para cada execução dos algoritmos de otimização. Portanto, a validação dos resultados não pode ser verificada realizando uma única otimização para cada problema. Em virtude disso, a apresentação dos resultados é feita de forma estatística através de uma amostra de execuções de cada metodologia, como sugerido por Gomes *et al.* (2018).

Nesse trabalho, optou-se por apresentações em tabelas e gráficos tipo violino com gráficos de caixa. Nas tabelas, os melhores resultados são apresentados em negrito, para facilitar na identificação desses. Os gráficos violinos, por sua vez, foram escolhidos por, além de apresentarem os mínimos e máximos encontrados, ilustrarem uma função de densidade do conjunto de dados. Aliando-os com gráficos de caixa, outras informações, como posição da mediana e percentis, podem ficar mais claras para o leitor quando apresentadas nesse formato de gráfico.

A Figura 26 apresenta uma demonstração de apresentação de um conjunto de dados por meio de um gráfico violino com gráfico de caixa. Nesse exemplo, apresentado na verticalmente e espelhado em relação ao eixo vertical a função de densidade ilustra a distribuição do conjunto de dados obtido. No centro desse espelhamento, é possível observar o gráfico de caixa. Nesse gráfico, os percentis são apresentados como vértices superior e inferior da caixa. A caixa então é formada pelos dados que estão dentro do intervalo formado pelos percentis desejados. Para os exemplos aqui apresentados, foi utilizado os percentis de 10% (inferior) e 90% (superior). Por fim, o seguimento de reta horizontal no interior da caixa representa a mediana do conjunto de dados apresentado.

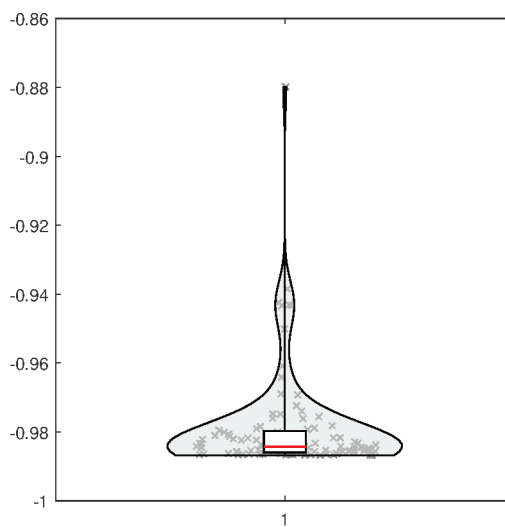


Figura 26 – Exemplo gráfico violino

7.2 PROBLEMAS MATEMÁTICOS

Para os problemas dessa seção foram realizadas 100 otimizações independentes com o objetivo de se obter uma distribuição dos resultados encontrados. Além da metodologia proposta nesse estudo, foram utilizados nessa análise os algoritmos GA, PSO, SGA, GOSAC.

Para essa análise, utilizou-se o *Genetic Algorithm* nativo do software MATLAB, sendo adotado como parâmetros de entrada:

- População inicial: 10 indivíduos;
- Número de gerações: 29.

Os demais parâmetros do algoritmo não foram alterados, utilizando os valores padrões fornecidos pelo software.

Assim como no algoritmo genético, o algoritmo *Particle Swarm Optimization* utilizado foi o nativo do software MATLAB. Como parâmetros de entrada, foram adotados:

- População inicial: 10 indivíduos;
- Número de iterações: 29.

No *Search Group Algorithm* utilizou-se o algoritmo implementado e fornecido pelo próprio autor Gonçalves, Lopez e Miguel (2015). Foram adotados com parâmetros de entrada:

- Mínimo valor que alfa pode assumir: 0.01;
- Alfa inicial: 2;
- Número máximo de iterações: 36;
- porcentagem dedicado ao esquema de seleção de fase global: 0.3;
- Tamanho da população: 10;
- Percentual da população que forma o grupo de busca: 0.2;
- Número de indivíduos com mutação: 0.

Para o algoritmo *Global Optimization with Surrogate Approximation of Constraints*, a implementação foi feita seguindo os pseudo-códigos fornecidos por Müller e Woodbury (2017) em seu trabalho de apresentação do método. Os parâmetros iniciais nesse caso foram:

- Número de pontos iniciais: $2(m + 1)$;
- Delta mínimo: 0.01;
- Geração de metamodelos: Método de *Kriging*:
 - Limite superior para máxima verossimilhança: 4;
 - Limite inferior para máxima verossimilhança: -3.
- Número máximo de chamada das funções de restrição: 150;

Os códigos da ferramenta *MATLAB Surrogate Model Toolbox* utilizados foram os fornecidos pela autora Mueller (2014) em sua página¹ do centro de ciências e engenharia computacional. Como parâmetro de entrada, foi utilizado apenas:

- Número máximo de chamada das funções de restrição: 150;

¹ <https://ccse.lbl.gov/people/julianem/>

Com relação a metodologia proposta nesse estudo, os parâmetros de entrada para os problemas matemáticos foram:

- Número de pontos iniciais: $2(m + 1)$;
- Delta Global: 0.02;
- Delta Local: 0.01;
- Número de pontos adicionais inseridos por iteração: 3;
- *passe*: 0;
- Número de iterações na fase 3: 10;
- Geração de metamodelos: Método de *Kriging*:
 - Limite superior para máxima verossimilhança: 4;
 - Limite inferior para máxima verossimilhança: -3.
- Número máximo de chamada das funções de restrição: 150;

7.2.1 Toy problem

Como mencionado anteriormente, esse problema foi proposto por Robert B. Gramacy *et al.* (2015) e trata-se de um problema de otimização analítico bidimensional com variáveis contínuas. Esse problema tem como função custo a equação:

$$f(\mathbf{x}) = x_1 + x_2.$$

As variáveis de projeto x_1 e x_2 são limitadas no intervalo de $[0, 1]$. A função esta sujeita ainda à duas funções não lineares, que representam as funções de restrição de caixa preta do problema. Essas funções são dadas por:

$$g_1(\mathbf{x}) = \frac{3}{2} - x_1 - 2x_2 - \frac{1}{2}\sin(2\pi(x_1^2 - 2x_2)) \leq 0$$

$$g_2(\mathbf{x}) = x_1^2 + x_2^2 - \frac{3}{2} \leq 0,$$

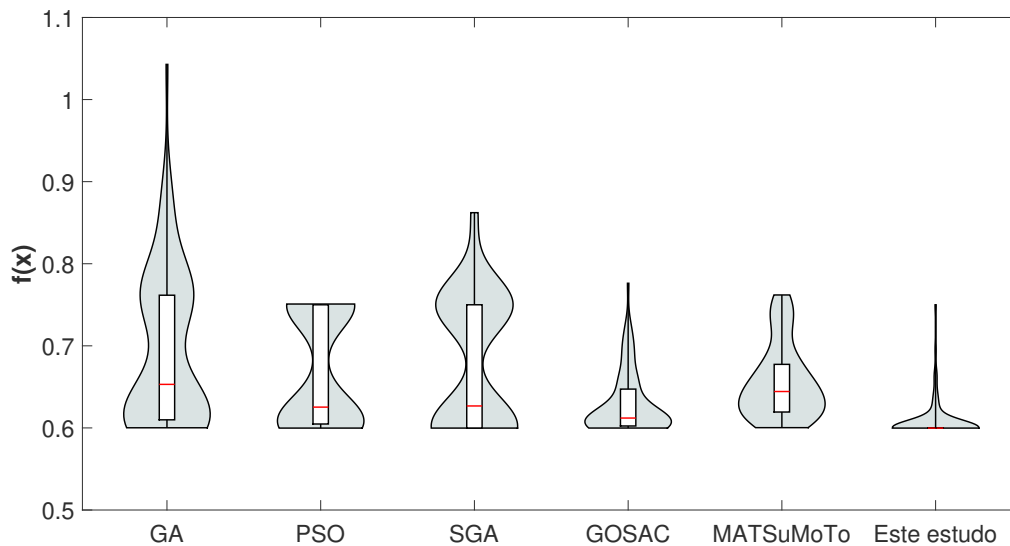
Os resultados obtidos estatísticos encontrados são apresentados na Tabela 3. Por sua vez, a Tabela 4 apresenta o número médio de avaliações das funções de restrição do problema. O número máximo de avaliações permitidas nos algoritmos *GA*, *PSO* e *SGA* foram dobrados, pois, nesses casos, a metodologia não foi projetada para a solução de problemas com poucas avaliações. Os melhores valores nessa tabela foram destacados em

Tabela 3 – Resultados estatísticos do *Toy Problem*.

Método	Menor Valor	Mediana	Maior Valor	Desvio Padrão	Percentil 10	Percentil 90
GA	0,6002	0,6530	1,0429	0,0942	0,6023	0,8026
PSO	0,5998	0,6253	0,7509	0,0690	0,6021	0,7501
SGA	0,5998	0,6268	0,8622	0,0817	0,5998	0,7500
GOSAC	0,5998	0,6120	0,7762	0,0367	0,6004	0,6887
MATSuMoTo	0,6003	0,6444	0,7620	0,0448	0,6072	0,7346
Este estudo	0,5998	0,6073	0,7500	0,0218	0,5998	0,6265
Diferença percentual	0,00%	1,25%	25,04%	-	-	-

Tabela 4 – Número de análise das restrições do *Toy Problem*.

Método	Nº médio de avaliações
GA	300
PSO	300
SGA	299
GOSAC	150
MATSuMoTo	150
Este estudo	112

Figura 27 – Distribuição dos resultados obtidos para o *Toy Problem*

negrito. A Figura 27 apresenta a distribuição dos valores ótimos encontrados em cada método.

O otimizador, definido por Robert B. Gramacy *et al.* (2015), encontra-se no ponto $\mathbf{x}^* = \{0,1954, 0,4044\}^T$ com valor na função custo de $f^* = 0,5998$.

7.2.2 Problema com restrição por *Eggholder* Modificada

O segundo problema analisado nessa sessão, proposto pelo autor desse trabalho, tem como função custo $f(\mathbf{x}) = 2 - 4x_1 - 4x_2 + 4x_1^2 + 4x_2^2$ com duas variáveis restritas a valores contínuos no intervalo entre 0 e 1. Como restrição, foi usada uma função adaptada da função de *Eggholder* (VANARET, 2020), que é dada por:

$$g_1(\mathbf{x}) = 0.15 + \frac{-(x'_2 + 47)\sin\left(\sqrt{|x'_2 + \frac{x'_1}{2} + 47|}\right) - x'_1\sin\left(\sqrt{|x'_1 - (x'_2 + 47)|}\right)}{400},$$

em que:

$$\begin{aligned}x'_1 &= (-0.5 + x_1) \cdot 600; \\x'_2 &= (-0.5 + x_2) \cdot 600.\end{aligned}$$

A representação das regiões factíveis do problema, bem como seu ponto ótimo, podem ser vistos na Figura 28. Nessa figura, pode-se observar que o problema conta com diversos mínimos locais encontrados em várias regiões no domínio. A avaliação desse exemplo tem como objetivo testar a capacidade do algoritmo em solucionar problemas de otimização cujo a função de restrição é complexa e irregular.

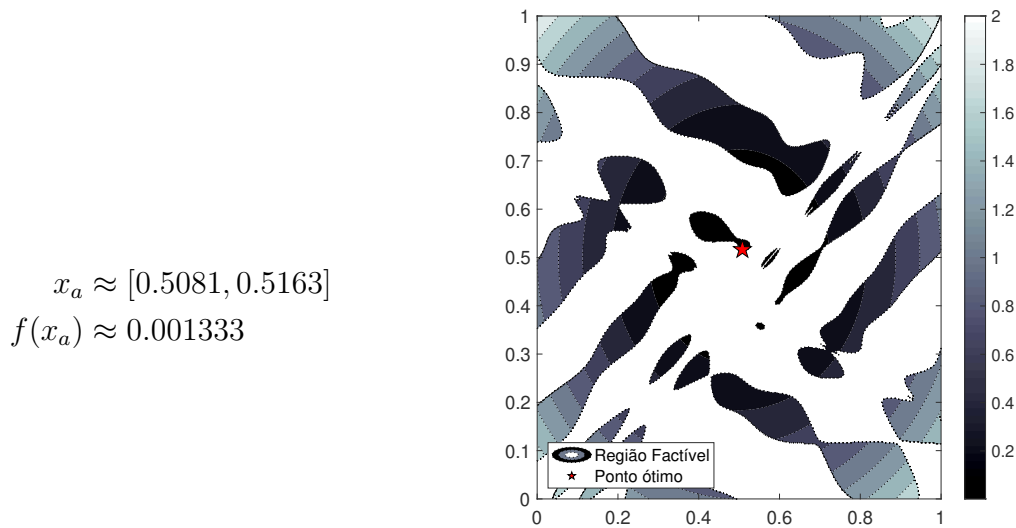


Figura 28 – Problema com restrição *Eggholder* modificada.

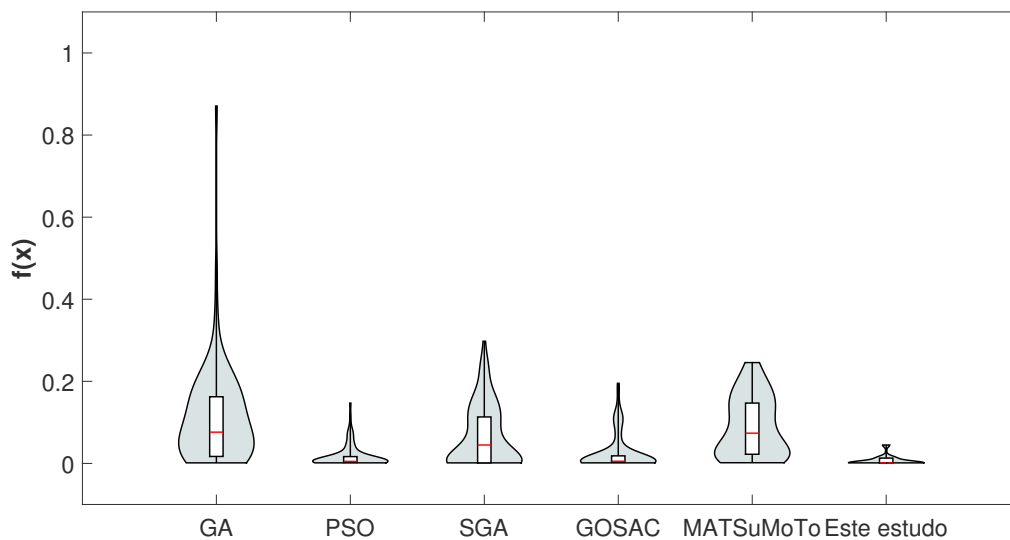
Por se tratar de uma função desenvolvida nesse trabalho, como citado anteriormente, o ponto de ótimo do problema foi adotado a partir de uma busca extensiva. Os resultados obtidos com as diferentes metodologias podem ser vistos na Tabela 5. Esses resultados são apresentados ainda na Figura 29. A Tabela 6 apresenta o número de avaliações da função de restrição.

Tabela 5 – Resultados estatísticos para o problema com restrição *Eggholder* modificada.

Método	Menor Valor	Mediana	Maior Valor	Desvio Padrão	Percentil 10	Percentil 90
GA	0,0013	0,0762	0,8708	0,1187	0,0050	0,2231
PSO	0,0014	0,0050	0,1471	0,0262	0,0017	0,0513
SGA	0,0013	0,0449	0,2978	0,0748	0,0013	0,1624
GOSAC	0,0014	0,0032	0,2682	0,0449	0,0016	0,0550
MATSuMoTo	0,0017	0,0737	0,2457	0,0737	0,0050	0,1914
Este estudo	0,0013	0,0013	0,0449	0,0118	0,0013	0,0159

Tabela 6 – Número de análise das restrições para o problema com restrição *Eggholder* modificada.

Método	Nº médio de avaliações
GA	300
PSO	300
SGA	299
GOSAC	200
MATSuMoTo	150
Este estudo	41,75

Figura 29 – Distribuição dos resultados obtidos para o problema com restrição *Eggholder* modificada.

7.2.3 Problema com restrição por *Rastrigin* Modificada

O próximo exemplo avaliado é o Problema com restrição por *Rastrigin* modificada, proposto pelo autor desse trabalho. esse problema tem como função custo $f(\mathbf{x}) = x_1^3 + x_2^3$ com duas variáveis contínuas restritas no intervalo entre 0 e 1. A função de restrição foi

baseada na função *Rastrigin* (POHLHEIM, 2007) e é dada por:

$$g_1(\mathbf{x}) = 1 - \frac{20 + \sum_{i=1}^2 [x_i^2 - 10\cos(2\pi x_i)]}{40}.$$

A Figura 30 mostra uma representação das regiões factíveis desse problema bem como seu ponto de mínimo.

$$x_a \approx [0.0912, 0.0912]$$

$$f(x_a) \approx 0.001515$$

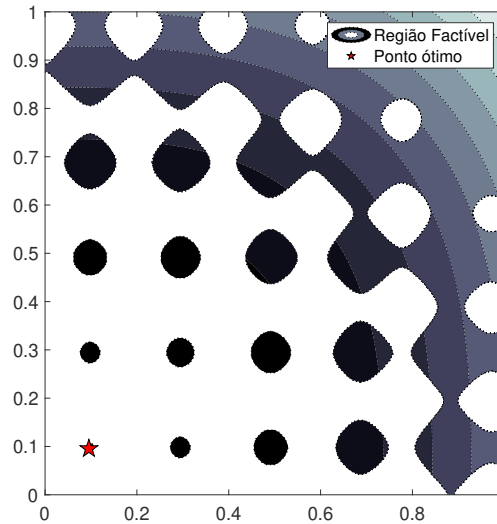


Figura 30 – Problema com restrição *Rastrigin* modificada.

Da mesma forma que o exemplo anterior, o ponto de ótimo do problema foi adotado a partir de uma busca extensiva, uma vez que o problema foi proposto pelo próprio autor desse trabalho. Os resultados obtidos com as diferentes metodologias podem ser vistos na Tabela 7. Esses resultados são apresentados ainda na Figura 31. A Tabela 8 apresenta o número de avaliações da função de restrição.

Tabela 7 – Resultados estatísticos para o problema com restrição *Rastrigin* modificada.

Método	Menor Valor	Mediana	Maior Valor	Desvio Padrão	Percentil 10	Percentil 90
GA	0,0213	0,1176	0,6247	0,1339	0,0283	0,3337
PSO	0,0017	0,0441	0,5309	0,0607	0,0216	0,1145
SGA	0,0210	0,1134	0,3455	0,0955	0,0211	0,2711
GOSAC	0,0015	0,0217	0,2789	0,0521	0,0017	0,1124
MATSuMoTo	0,0021	0,1091	0,3815	0,0641	0,0416	0,1655
Este estudo	0,0015	0,0015	0,0268	0,0035	0,0015	0,0015

7.2.4 Problema com restrição por *Cross-in-Tray* Modificada

O quarto problema dessa seção é possui como função custo a mesma do problema anterior, $f(\mathbf{x}) = x_1^3 + x_2^3$. Como função de restrição, foi utilizada uma adaptação da

Tabela 8 – Número de análise das restrições para o problema com restrição *Rastrigin* modificada.

Método	Nº médio de avaliações
GA	300
PSO	300
SGA	299
GOSAC	150
MATSuMoTo	150
Este estudo	105,37

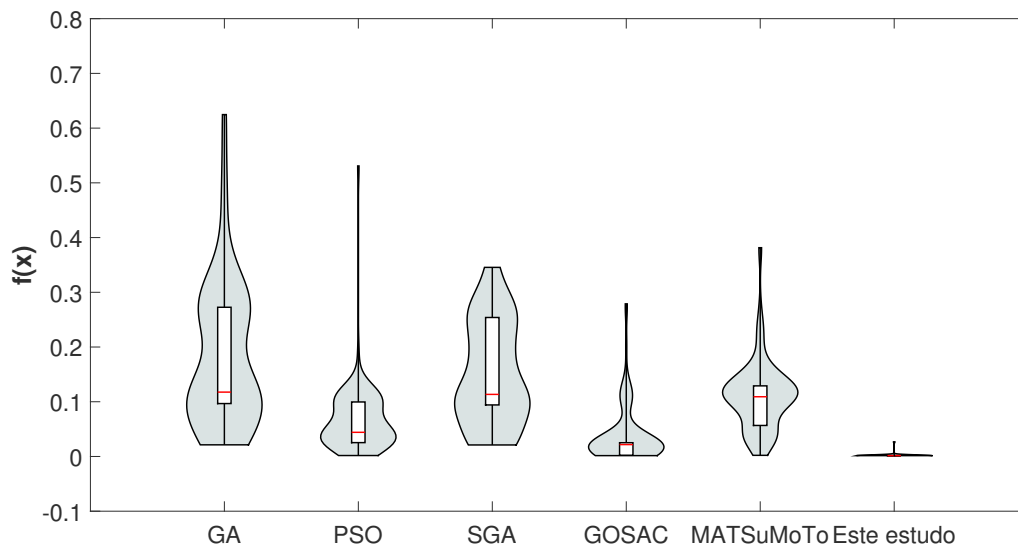


Figura 31 – Distribuição dos resultados obtidos para o problema com restrição *Rastrigin* modificada.

função *Cross-in-Tray*, obtida no trabalho de Jamil e Yang (2013). As variáveis de projeto do problema estão contidas no intervalo $[0, 1]$. A Figura 32 apresenta uma representação gráfica do problema estudado, bem como o seu ponto ótimo.

Os resultados obtidos com as diferentes metodologias são demonstrados na Tabela 9 e na Figura 33. A Tabela 10 apresenta o número de avaliações da função de restrição.

Nesse exemplo, alguns algoritmos, em determinadas execuções do processo de otimização, não encontraram nenhum ponto de mínimo viável. Dentre esses, o *GA* e o *PSO* foram os que obtiveram maiores casos de não convergência, com 85% e 84%, respectivamente. Os não convergentes foram removidos para a realização da análise estatística. A Tabela 11 apresenta o número realizações que obtiveram sucesso em encontrar ao menos um mínimo.

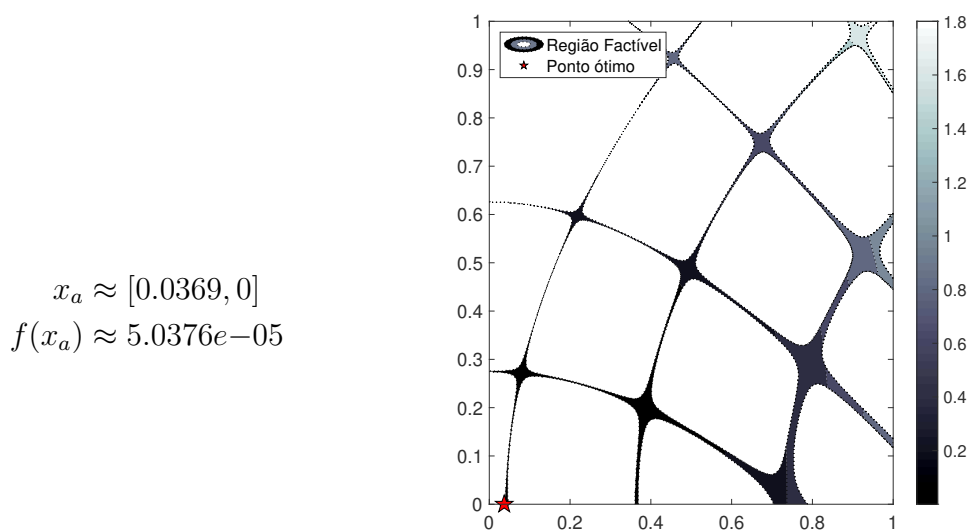


Figura 32 – Problema com restrição *Cross-in-Tray* modificada.

Tabela 9 – Resultados estatísticos para o problema com restrição *Cross-in-Tray* modificada.

Método	Menor Valor	Mediana	Maior Valor	Desvio Padrão	Percentil 10	Percentil 90
GA	5,08e-05	0,0757	1,6242	0,2567	5,57e-04	0,5220
PSO	5,04e-05	5,62e-05	0,2366	0,0433	5,04e-05	0,0510
SGA	5,04e-05	0,0074	0,8193	0,1723	5,04e-05	0,3071
GOSAC	5,07e-05	0,0169	0,3643	0,0682	7,48e-05	0,0544
MATSuMoTo	5,26e-05	0,0435	0,3626	0,0960	9,42e-05	0,2311
Este estudo	5,04e-05	5,82e-04	0,0484	0,0114	5,04e-05	0,0060

Tabela 10 – Número de análise das restrições para o problema com restrição *Cross-in-Tray* modificada.

Método	Nº médio de avaliações
GA	300
PSO	300
SGA	299
GOSAC	150
MATSuMoTo	150
Este estudo	78,86

7.2.5 Problema 12 *paper GOSAC*

Esse problema foi apresentado por Müller e Woodbury (2017) em seu artigo de demonstração da metodologia *Global Optimization with Surrogate Approximation of Constraints*. Esse consiste em um problema com dimensão 3 e consiste em minimizar a função:

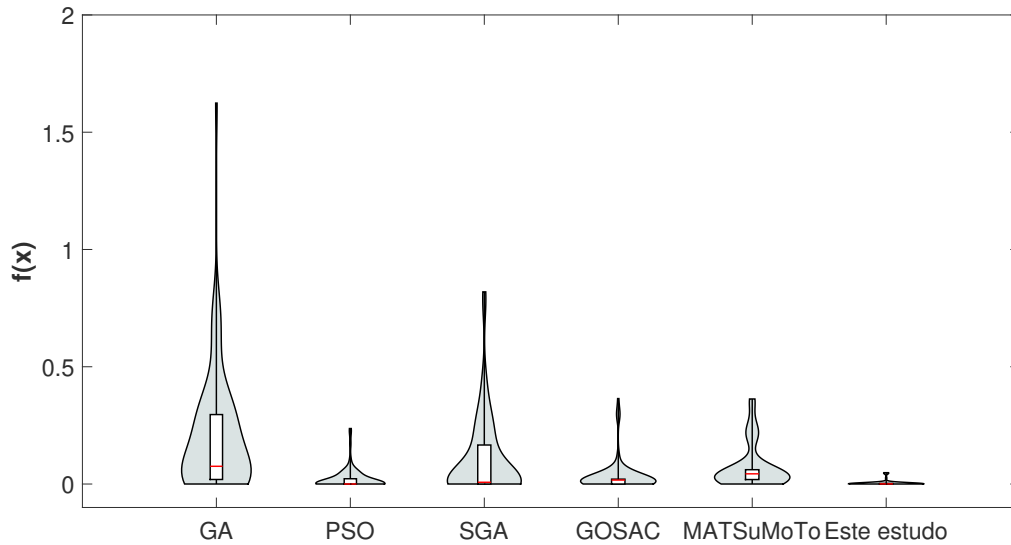


Figura 33 – Distribuição dos resultados obtidos para o problema com restrição *Cross-in-Tray* modificada.

Tabela 11 – Número realizações convergentes utilizadas na análise estatística para o problema com restrição *Cross-in-Tray* modificada.

Método	N convergências
GA	85
PSO	84
SGA	96
GOSAC	100
MATSuMoTo	100
Este estudo	100

$$f(\mathbf{x}) = -0,7x_1 + 5(x_2 - 0,2)^2 + 0,8,$$

sujeita as restrições

$$g_1(\mathbf{x}) = -\exp(x_2 - 0,2) - x_3 \leq 0,$$

$$g_2(\mathbf{x}) = 1,1x_1 + x_3 + 1 \leq 0,$$

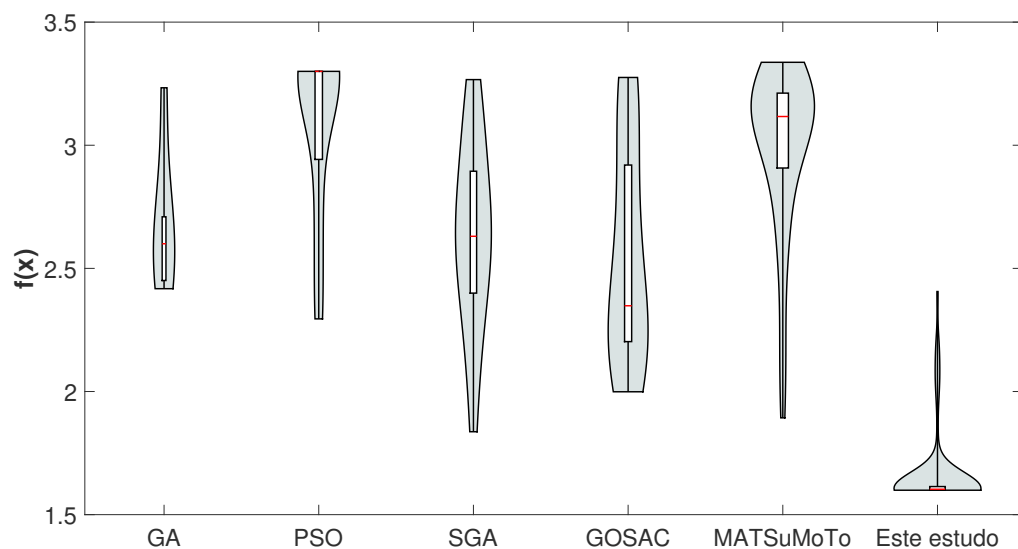
$$g_3(\mathbf{x}) = -1,2x_1 + x_2 \leq 0.$$

As variáveis de projeto são contínuas e estão contidas no intervalo $\mathcal{X} = [0; 1] \times [0, 2; 1] \times [-2, 22554; -1]$. O valor mínimo do problema, apresentado pela autora, é $y^* = 1,5991$ no ponto $\mathbf{x} = \{0,5752; 0,6903; -1,6327\}^T$. A Tabela 12 apresenta os resultados obtidos para o problema com as diversas metodologias. A distribuição dos resultados obtidos por cada metodologia são apresentados na Figura 34.

Assim como no problema anterior, algumas metodologias apresentaram dificuldades para encontrar um ponto de mínimo em algumas realizações, a Tabela 13 apresenta o

Tabela 12 – Resultados estatísticos para o problema 12 *paper GOSAC*.

Método	Menor Valor	Mediana	Maior Valor	Desvio Padrão	Percentil 10	Percentil 90
GA	2,4175	2,6000	3,2333	0,2971	2,4175	3,2333
PSO	2,2949	3,3000	3,3000	0,3484	2,4088	3,3000
SGA	1,8367	2,6372	3,2665	0,3793	2,1493	3,1523
GOSAC	1,9989	2,4157	3,2752	0,4441	2,0620	3,2357
MATSuMoTo	1,8931	3,1192	3,3368	0,3403	2,4682	3,2870
Este estudo	1,5992	1,6031	2,4060	0,1553	1,5995	1,7934
Diferença percentual	0,01 %	0,02%	50,47%	-	-	-

Figura 34 – Distribuição dos resultados obtidos para o problema 12 do *paper GOSAC*.

número de realizações convergentes utilizadas na análise estatística. Por fim, a Tabela 14 apresenta o número médio de chamadas necessárias para a solução do problema utilizando cada metodologia.

Tabela 13 – Número realizações convergentes utilizadas na análise estatística para o problema 12 do *paper GOSAC*.

Método	N convergências
GA	6
PSO	23
SGA	17
GOSAC	21
MATSuMoTo	53
Este estudo	100

Tabela 14 – Número de análise das restrições para o problema 12 do *paper GOSAC*.

Método	Nº médio de avaliações
GA	300
PSO	300
SGA	299
MATSuMoTo	150
Este estudo	164

7.2.6 Toy Problem modificado

O último problema analisado nessa seção é o *Toy Problem* modificado, apresentado na seção 6.2. Como visto anteriormente, esse problema possui como característica a presença de múltiplos pontos de mínimo local na função custo. A função custo desse problema é dada por:

$$f(\mathbf{x}) = 0.15 + \left(-(w_2 + 47) \sin \left(\sqrt{\left| w_2 + \frac{w_1}{2} + 47 \right|} \right) - w_1 \sin \left(\sqrt{|w_1 - (w_2 + 47)|} \right) \right) / 400,$$

em que:

$$w_1 = 200(-0.5 + x_1)$$

$$w_2 = 200(1.5 + x_2).$$

Essa função custo esta sujeita as restrições:

$$g_1(\mathbf{x}) = \frac{3}{2} - x_1 - 2x_2 - \frac{1}{2} \sin(2\pi(x_1^2 - 2x_2)) \leq 0$$

$$g_2(\mathbf{x}) = x_1^2 + x_2^2 - \frac{3}{2} \leq 0.$$

As variáveis de projeto desse problema são contínuas e estão restritas no intervalo $\mathcal{X} = [0; 1] \times [0; 1]$. A Tabela 15, bem como a Figura 35, apresentam os resultados obtidos para o presente problema. O número de avaliações das funções de restrição podem ser vistos na Tabela 16.

Tabela 15 – Resultados estatísticos para o *Toy Problem* Modificado.

Método	Menor Valor	Mediana	Maior Valor	Desvio Padrão	Percentil 10	Percentil 90
GA	-0,9867	-0,9399	-0,3974	0,0970	-0,9851	-0,7796
PSO	-0,9867	-0,9843	-0,8798	0,0150	-0,9866	-0,9659
SGA	-0,9868	-0,9862	-0,7961	0,0563	-0,9868	-0,8886
GOSAC	-0,9824	-0,9490	-0,8865	0,0263	-0,9691	-0,9000
MATSuMoTo	-0,9861	-0,9445	-0,8811	0,0283	-0,9750	-0,8976
Este estudo	-0,9868	-0,9868	-0,8881	0,0172	-0,9868	-0,9665

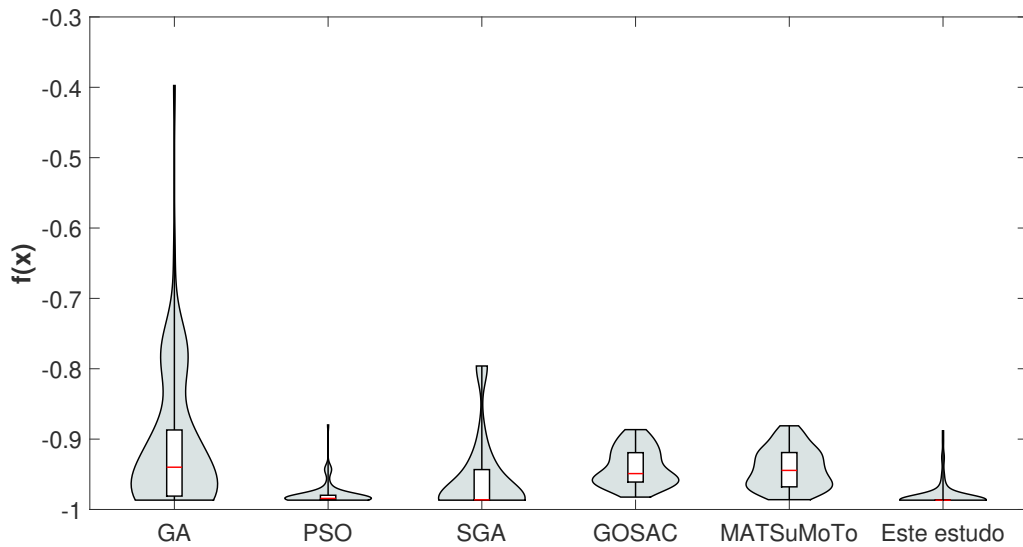


Figura 35 – Distribuição dos resultados obtidos para o *Toy Problem Modificado*.

Tabela 16 – Número de análise das restrições para o *Toy Problem Modificado*.

Método	Nº médio de avaliações
GA	300
PSO	300
SGA	299
GOSAC	150
MATSuMoTo	150
Este estudo	112,01

Assim como nos demais problemas desenvolvidos pelo autor desse estudo, o ponto de mínimo foi encontrado a partir de busca excessiva pelo ponto ótimo. Para esse problema, foi encontrado como ponto ótimo o valor $y^* = -0,9868$ com o minimizador $\mathbf{x} = [0.2457, 0.4801]$.

7.2.7 Discussão dos problemas matemáticos

Com os problemas matemáticos foi possível observar o comportamento dos resultados encontrados pela metodologia proposta nesse trabalho e compara-los aos resultados obtidos por outras metodologias. Observou-se que o método estudado nesse trabalho apresenta uma boa capacidade de captura do ponto de mínimo de um problema de otimização quando comparado com outras metodologias já validadas no meio acadêmico. Através da análise do número de avaliações das funções de restrição dos problemas, observou-se ainda que essa nova metodologia apresenta como característica a capacidade de obtenção do ponto de mínimo com um número de análises baixo. Essa é uma das características mais buscada nesse trabalho, uma vez que o método aqui proposto é sugerido para problemas

em que o custo de análise das restrições desse são demasiadamente altas.

7.3 PROBLEMAS DA ENGENHARIA

7.3.1 Mola sob tração/compressão

O primeiro problema de otimização clássico da engenharia analisado nessa seção consiste na minimização do volume de uma mola sujeita a tração/compressão, dado por:

$$V = (x_3 + 2)x_2x_1^2.$$

O problema foi proposto por Mezura-Montes, Coello e Landa-Becerra (2003) e sua representação esta ilustrada na Figura 36. Como variáveis de projeto estão diâmetro do fio ($d = x_1 \in [0,05; 2]$), diâmetro de cada volta ($D = x_2 \in [0,25; 1,3]$) e o número de espirais ativos ($N = x_3 \in [2; 15]$).

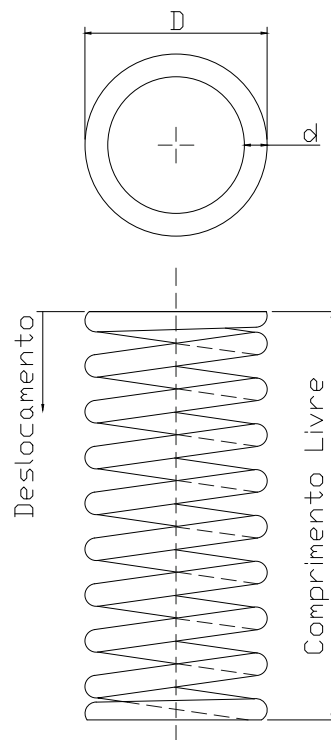


Figura 36 – Representação mola sob tração/compressão.

A mola esta sujeita a restrições mecânicas dadas por:

$$\begin{aligned}
g_1(\mathbf{x}) &= 1 - x_2^3 x_3 \leq 0 \\
g_2(\mathbf{x}) &= \frac{4x_2^2 - x_1 x_2}{12566(x_2 x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \leq 0 \\
g_3(\mathbf{x}) &= 1 - \frac{140,45x_1}{x_2^2 x_3} \leq 0 \\
g_4(\mathbf{x}) &= \frac{x_2 + x_1}{1,5} - 1 \leq 0.
\end{aligned}$$

Para essa otimização, foram executadas 10 realizações com a metodologia desenvolvida nesse estudo. Como parâmetros de entrada, foram utilizados:

- Número de pontos iniciais: $2(m + 1)$;
- Delta Global: 0,02;
- Delta Local: 0,01;
- Número de pontos adicionais inseridos por iteração: 3;
- *passo*: 0,2;
- Número de iterações na fase 3: 10;
- Geração de metamodelos: Método de *Kriging*:
 - Limite superior para máxima verossimilhança: 4;
 - Limite inferior para máxima verossimilhança: -3.
- Número máximo de chamada das funções de restrição: 350;

Os resultados obtidos são apresentados na Tabela 17. A Tabela 18 apresenta uma comparação entre os resultados obtidos com a metodologia proposta nesse trabalho e o encontrado por diferentes metodologias que utilizam otimização com metamodelos.

Tabela 17 – Resultados estatísticos para o problema de Mola sob tração/compressão.

Menor Valor	Mediana	Maior Valor	Desvio Padrão	Percentil 10	Percentil 90	Nº médio de avaliações
0,0126	0,0141	0,0414	0,0094	0,0127	0,0346	250

Tabela 18 – Comparação dos resultados do problema de Mola sob tração/compressão com diferentes autores.

Metodologia	d	D	N	V_{best}	V_{mean}
Carvalho (2014)(Heurístico)	0,05406	0,41655	8,48436	0,01266	0,01392
GOSAC	0,0541	0,3784	12,960	0,0166	0,0211
MATSuMoTo	0,0555	0,4456	7,7471	0,0134	0,1102
Este estudo	0,0512	0,3441	12,0208	0,0127	0,0187

7.3.2 Viga soldada

O segundo problema dessa seção trata-se da minimização do custo (C) de uma viga soldada Mezura-Montes, Coello e Landa-Becerra (2003). Nesse exemplo, as variáveis de projeto são dadas por $h = x_1 \in [0, 125; 2, 0]$, $l = x_2 \in [0, 1; 10, 0]$, $t = x_3 \in [0, 1; 10, 0]$ e $b = x_4 \in [0, 1; 2, 0]$. A Figura 37 ilustra o problema. A função custo é dada por:

$$C(\mathbf{x}) = 1.10471x_1^2x_2 + 0,04811x_3x_4(14,0 + x_2)$$

sujeita à:

$$g_1(\mathbf{x}) = \tau(\mathbf{x}) - \tau_{max} \leq 0,$$

$$g_2(\mathbf{x}) = \sigma(\mathbf{x}) - \sigma_{max} \leq 0,$$

$$g_3(\mathbf{x}) = x_1 - x_4 \leq 0,$$

$$g_4(\mathbf{x}) = 0,10471x_1^2 + 0,04811x_3x_4(14,0 + x_2) - 5,0 \leq 0,$$

$$g_5(\mathbf{x}) = \delta(\mathbf{x}) - \delta_{max} \leq 0,$$

$$g_6(\mathbf{x}) = P - P_c(\mathbf{x}) \leq 0,$$

em que:

$$\begin{aligned}
\tau(\mathbf{x}) &= \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}, \\
\tau' &= \frac{P}{\sqrt{2}x_1x_2}, \\
\tau'' &= \frac{MR}{J}, \\
M &= P\left(L + \frac{x_2}{2}\right), \\
R &= \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1x_3}{2}\right)^2}, \\
J &= 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\}, \\
\sigma(\mathbf{x}) &= \frac{6PL}{x_4x_3^2}, \\
\delta(\mathbf{x}) &= \frac{4PL^3}{Ex_3^3x_4}, \\
P_c(\mathbf{x}) &= \frac{4,013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right), \\
P &= 6000lb, \\
L &= 14in, \\
E &= 30 \times 10^6psi, \\
G &= 12 \times 10^6psi, \\
\tau_{max} &= 13600psi, \\
\sigma_{max} &= 30000psi, \\
\delta_{max} &= 0,25in.
\end{aligned}$$

Assim como no exemplo anterior, foram feitas 10 realizações independentes com a metodologia desenvolvida nesse estudo. Os parâmetros de entrada, forma utilizados:

- Número de pontos iniciais: $2(m + 1)$;
- Delta Global: 0,02;
- Delta Local: 0,01;
- Número de pontos adicionais inseridos por iteração: 3;
- *passe*: 0,2;
- Número de iterações na fase 3: 10;
- Geração de metamodelos: Método de *Kriging*:
 - Limite superior para máxima verossimilhança: 4;

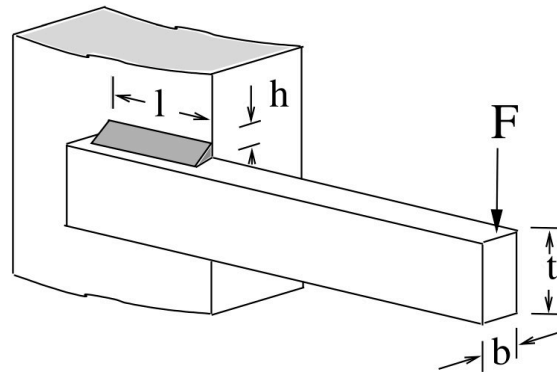


Figura 37 – Representação Viga soldada.

Fonte: Carvalho (2014)

- Limite inferior para máxima verossimilhança: -3.
- Número máximo de chamada das funções de restrição: 400;

Os resultados encontrados na solução desse problema são mostrados na Tabela 19. A Tabela 20 apresenta a comparação dos resultados obtidos entre a metodologia apresentada nesse trabalho e os resultados encontrados por outros autores e outras metodologias que utilizam modelos aproximados.

Tabela 19 – Resultados estatísticos para o problema de Viga soldada.

Menor Valor	Mediana	Maior Valor	Desvio Padrão	Percentil 10	Percentil 90	Nº médio de avaliações
2,6296	4,2089	5,0138	0,7851	2,8890	4,8526	238,60

Tabela 20 – Comparação dos resultados do problema de Viga soldada com diferentes metodologias.

Metodologia	Execuções convergentes	V_{best}	V_{mean}
Carvalho (2014)(Heurístico)	-	2,3811	2,6710
MATSuMoTo	5/10	2,3509	3,8833
Este estudo	10/10	2,6296	4,2089

7.3.3 Discussões sobre os problemas clássicos da engenharia

A avaliação da metodologia proposta em problemas clássicos da engenharia permitiu uma comparação entre os resultados obtidos por essa e os apresentados na literatura. No problema da mola sob tração/compressão, observou-se que a aproximação do ponto

mínimo foi bem satisfatória, com um diferença percentual de 0,01%. Todavia, ao observar o problema da viga soldada, a diferença entre o mínimo encontrado e o melhor mínimo presente na literatura foi ligeiramente divergente, com diferença de aproximadamente 11,86% do menor valor encontrado. Observando os dados gerados durante o processo de otimização, notou-se uma má representação do metamodelo em relação as restrições do problema. Outro ponto que acabou causando essa diferença foi a adoção de deltas muito grandes, que impediram a criação de pontos muito próximos. Isso pode ter feito com que pontos melhores e factíveis fossem desconsiderados, uma vez que já havia outro ponto na região.

É válido observar, entretanto, que a metodologia apresentou resultados satisfatórios com um número baixo de avaliações. No problema da mola sob tração/compressão, o número médio de avaliações realizadas pela metodologia GOSAC e MATSuMoTo foi de 350 avaliações. Na metodologia utilizada pelo Carvalho (2014), foram realizadas uma quantidade de 36000 de avaliações. Já o método proposto neste trabalho conseguiu encontrar pontos otimizadores com uma média de 250 avaliações.

No problema da viga soldada, essa diferença se repete, com a metodologia aqui proposta encontrando pontos ótimos com uma média de 238,6 avaliações, contra uma média de 400 avaliações para o MATSuMoTo e 360000 para os resultados do Carvalho (2014).

7.4 PROBLEMAS DE OTIMIZAÇÃO ESTRUTURAL

7.4.1 Treliça de 25 barras - Determinística e contínua

Esse problema consiste em minimizar o peso (P) da treliça de 25 barras apresentada na Figura 38. As áreas das seções transversais (a_k) são as variáveis e são do tipo contínuas. A massa específica do material (ρ) foi definida como única para todos os perfis da estrutura.

Para manter a simetria da estrutura, os elementos que formam a torre foram agrupados em 8 grupos. Nesse caso, o número de variáveis de projeto é, também, igual a 8, uma vez que o objetivo é encontrar $a_k = A_1, \dots, A_i$ que minimize o peso da torre.

Como condições de restrição, foram adotadas as tensões máximas atuantes em cada perfil, que deve estar em um intervalo de $[-40, 40]$ *ksi* e os deslocamentos globais máximos nos nós 1 e 2, que não pode ser superior a 0,35 *in* em quaisquer direção. A densidade do material que compõe as barras da estrutura é de 0.1 *lb/in³*. Já o módulo de elasticidade $E = 10^4$ *ksi*.

A Tabela 21 apresenta os grupos que foram o agrupamento dos elementos da torre e na Tabela 22 são apresentadas as forças atuantes na estrutura. Foram executadas 30 realizações independentes para esse problema cujo os parâmetros de entrada foram:

- Número de pontos iniciais: $2(m + 1)$;

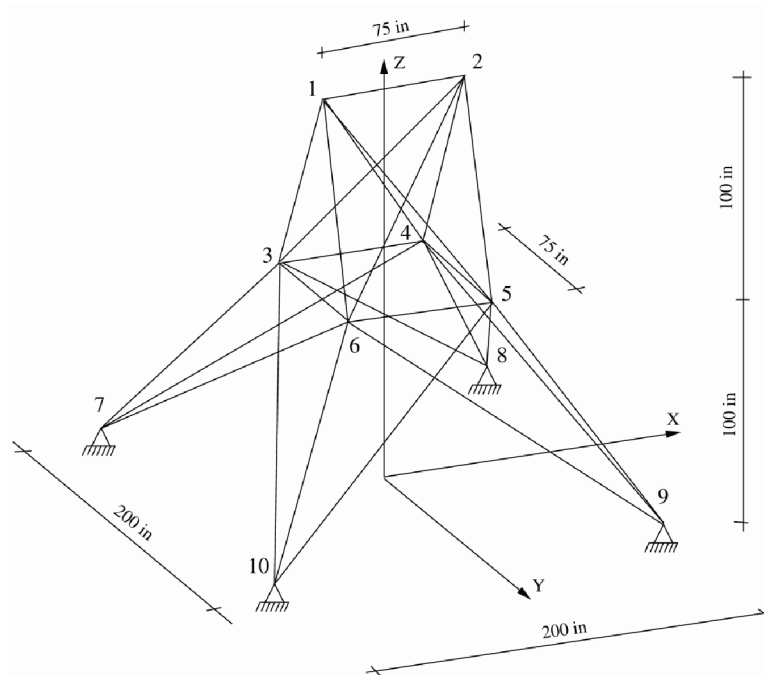


Figura 38 – Representação Torre 25 barras.

Fonte: Carvalho (2014)

- Delta Global: 0,02;
- Delta Local: 0,01;
- Número de pontos adicionais inseridos por iteração: 3;
- *passo*: 0,2;
- Número de iterações na fase 3: 10;
- Geração de metamodelos: Método de *Kriging*:
 - Limite superior para máxima verossimilhança: 4;
 - Limite inferior para máxima verossimilhança: -3.
- Número máximo de chamada das funções de restrição: 900;

A Tabela 23 apresenta os resultados obtidos com a metodologia aqui apresentada para o problema Treliça de 25 barras. Na Tabela 24 os resultados obtidos são comparados aos resultados fornecidos por outros autores da literatura.

7.4.2 Treliça 25 barras - Determinística e discreta

O segundo problema nessa seção de otimizações estruturais trata a mesma otimização realizada anteriormente, todavia, as seções transversais devem ser escolhidas a partir

Tabela 21 – Grupos de elementos para a Treliça de 25 barras.

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

Tabela 22 – Esforços atuantes na Treliça de 25 barras (em *kips*).

Nó	F_x	F_y	F_z
1	1,0	-10,0	-10,0
2	0,0	-10,0	-10,0
3	0,5	0,0	0,0
6	0,5	0,0	0,0

Tabela 23 – Resultados estatísticos para o problema Treliça de 25 barras - Determinística e contínua.

Menor Valor	Mediana	Maior Valor	Desvio Padrão	Percentil 10	Percentil 90	Nº médio de avaliações
527,3157	596,0663	687,2837	34,4722	553,1802	631,0266	183,1667

Tabela 24 – Comparação dos resultados do problema Treliça de 25 barras - Determinística e contínua.

Metodologia	Execuções convergentes	P_{best}	P_{mean}
MATSuMoTo	30/30	528,44	595,662
Este estudo	30/30	535,69	591,559
Diferença percentual	-	1,37%	- %

de um conjunto de seções. As áreas (em in^2) desse conjunto são: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.8, 3.0, 3.2, 3.4. Os demais dados são os mesmo do problema anterior, bem como os parâmetros de entrada do algoritmo.

A Tabela 25 apresenta os resultados obtidos com a metodologia proposta nesse trabalho para esse problema. Esses resultados são comparados aos obtidos na literatura, como visto na Tabela 26.

Tabela 25 – Resultados estatísticos para o problema Treliça de 25 barras - Determinística e variáveis discretas.

Menor Valor	Mediana	Maior Valor	Desvio Padrão	Percentil 10	Percentil 90	Nº médio de avaliações
543,269	603,073	759,130	48,67	669,894	561,517	145,53

Tabela 26 – Comparação dos resultados do problema Treliça de 25 barras - Determinística e variáveis discretas.

Metodologia	Nº de execuções convergentes	P_{best}	P_{mean}
MATSuMoTo	30	541,620	601,773
Este estudo	30	543,269	611,434
Diferença percentual	-	0,304%	-

7.4.3 Discussões sobre os problemas de otimização estrutural

A aplicação da metodologia em problemas de otimização estrutural permitiu analisar o comportamento do modelo em situações com várias variáveis de projeto, sejam elas contínuas ou discretas. Comparando o resultado obtido com os fornecidos por outras metodologias que utilizam metamodelos, pôde-se avaliar o comportamento da metodologia aqui desenvolvida com outras já consolidadas.

Ambos os métodos conseguiram encontrar valores ótimos com um número baixo de avaliações das funções de restrição. O MATSuMoTo foi capaz de encontrar o ponto ótimo com uma média de 250 avaliações em ambos os problemas. Por outro lado, a metodologia aqui proposta foi capaz de encontrar os pontos ótimos com uma média de de 183,167 para a estrutura com variáveis de projeto contínuas e 145,53 para o problema com variáveis discretas. Essa característica permite validar um dos objetivos iniciais, que visa o desenvolvimento de uma metodologia capaz de solucionar problemas de otimização cujas funções de restrição possuam um altíssimo custo computacional.

8 CONCLUSÃO E TRABALHOS FUTUROS

8.1 CONCLUSÃO

O presente trabalho tratou do desenvolvimento de uma metodologia de otimização de problemas *black-box* com restrições computacionalmente onerosas e função custo podendo ser calculada em fração de segundos. O algoritmo proposto tem como característica o uso do método de *Kriging* para substituir as funções com alto custo computacional por um metamodelo com menor custo. O método foi escolhido, nesse trabalho, devido sua qualidade nos resultados, rapidez e precisão. Considerou-se também a facilidade de implementação do método bem como a ampla gama de estudos encontrados na literatura e que puderam servir como base para o estudo e desenvolvimento dessa nova metodologia de otimização.

Como diferencial dos métodos encontrados na literatura, a metodologia apresenta algumas características importantes, sendo elas: o refinamento da função de restrição, a alteração gradual dos limites do domínio do problema, a utilização de uma população no processo iterativo e a possibilidade de convergência na direção de vários pontos de ótimo local. A rotina proposta foi implementada no software MATLAB® e avaliada em três conjuntos de problemas distintos: problemas matemáticos, problemas clássicos da engenharia encontrados na literatura, e, por fim, problemas de otimização do campo da engenharia estrutural. Os resultados foram comparados à valores obtidos na literatura e/ou resultados fornecidos por outras metodologias já consolidadas, como o *toolbox MATSuMoTo*, o algoritmo *GOSAC*, entre outro.

Os resultados obtidos se mostraram satisfatórios, obtendo-se, muitas vezes, valores bem próximos quando comparados a outras metodologias. O algoritmo se mostrou ainda eficiente em transpor áreas restritas do domínio para encontrar regiões isoladas no domínio. O método cumpriu ainda o quesito de tempo computacional, fornecendo a resolução dos problemas com um baixo número de execuções das funções onerosas. É interessante enfatizar a eficiência do método na solução do problema *Eggholder* Modificada, capítulo 7.2.2, em que foi possível encontrar o ponto de mínimo com uma média de apenas 41,75 execuções das funções onerosas. Nesse problema foi possível ver ainda que, mesmo atingindo o número máximo de interações, o modelo tende a encontrar o melhor ponto.

8.2 TRABALHOS FUTUROS

Como sugestão para trabalhos futuros propõe-se os seguintes pontos:

- Aperfeiçoar o método de modificação dos limites de caixa, utilizando métricas que dependem de modelos estatísticos. A adoção de um método que utilize funções de distribuição, centradas nos pontos de referência, para a população inserida à cada

execução no processo iterativo, pode tornar a metodologia ainda mais eficiente e robusta.

- Aplicar a metodologia à projetos de otimização baseados em confiabilidade (*RBDO*).
- Estender o método para problemas de altas dimensões, utilizando métricas de krigagem apropriadas para esses tipos de problemas.
- Verificar, através de uma análise paramétrica, os valores ótimos para os parâmetros do algoritmo e verificar a sensibilidade do método quanto a esses parâmetros de entrada;

REFERÊNCIAS

- ARORA, Jasbir S. **Introduction to optimum design**. Edição: Elsevier. 3. ed. Iowa City: Elsevier BV, 2012. P. 880.
- BERTSEKAS, Dimitri P. **Constrained optimization and Lagrange multiplier methods**. [S.l.]: Academic press, 2014.
- BJÖRKMAN, Mattias; HOLMSTRÖM, Kenneth. Global optimization of costly nonconvex functions using radial basis functions. **Optimization and Engineering**, Springer, v. 1, n. 4, p. 373–397, 2000.
- BURDEN, Richard L; FAIRES, Douglas J. Numerical analysis. PWS publishing company, 1985.
- CAMARGO, Eduardo Celso Gerbi. Desenvolvimento, Implementação E Teste De Procedimentos Geoestatísticos (Krigagem) No Sistema De Processamento De Informações Georreferenciadas (SPRING), 1997.
- CARRARO, Felipe. A stochastic kriging approach for the minimization of integrals, 2017.
- CARVALHO, ECR. Solução de problemas de otimização com restrições usando estratégias de penalização adaptativa e um algoritmo do tipo PSO. **Mestrado, PPGMC, Programa de Pós-graduação em Modelagem Computacional, UFJF**, 2014.
- COSTA, Jean-Pierre; PRONZATO, Luc; THIERRY, Eric. A comparison between Kriging and radial basis function networks for nonlinear prediction. *In: NSIP*. [S.l.: s.n.], 1999. P. 726–730.
- CRESSIE, Noel. Statistics for spatial data. **Terra Nova**, Wiley Online Library, v. 4, n. 5, p. 613–617, 1992.
- EBERHART, Russell; KENNEDY, James. Particle swarm optimization. *In: CITESEER. PROCEEDINGS of the IEEE international conference on neural networks*. [S.l.: s.n.], 1995. P. 1942–1948.
- FLETCHER, R. **Practical methods of optimization**. 2. ed. [S.l.]: Wiley, 1987.
- FORRESTER, Alexander IJ; KEANE, Andy J. Recent advances in surrogate-based optimization. **Progress in aerospace sciences**, Elsevier, v. 45, n. 1-3, p. 50–79, 2009.
- FORRESTER, Alexander; SOBESTER, Andras; KEANE, Andy. **Engineering design via surrogate modelling: a practical guide**. [S.l.]: John Wiley & Sons, 2008.

- GOLDBERG, David E. Genetic algorithms in search. **Optimization, and Machine Learning**, Addison Wesley Publishing Co. Inc., 1989.
- GOLDBERG, David E; HOLLAND, John H. Genetic algorithms and machine learning. **Machine learning**, Springer, v. 3, n. 2, p. 95–99, 1988.
- GOMES, Wellison JS *et al.* A probabilistic metric for comparing metaheuristic optimization algorithms. **Structural Safety**, Elsevier, v. 70, p. 59–70, 2018.
- GONÇALVES, Matheus Silva; LOPEZ, Rafael Holdorf; MIGUEL, Leandro Fleck Fadel. Search group algorithm: a new metaheuristic method for the optimization of truss structures. **Computers & Structures**, Elsevier, v. 153, p. 165–184, 2015.
- GRAMACY, Robert B. *et al.* **Modeling an Augmented Lagrangian for Blackbox Constrained Optimization**. [*S.l.*: *s.n.*], 2015. arXiv: 1403.4890 [stat.ME].
- GRAMACY, Robert B; POLSON, Nicholas G. Particle learning of Gaussian process models for sequential design and optimization. **Journal of Computational and Graphical Statistics**, Taylor & Francis, v. 20, n. 1, p. 102–118, 2011.
- GRAMACY, Robert B *et al.* Modeling an augmented Lagrangian for blackbox constrained optimization. **Technometrics**, Taylor & Francis, v. 58, n. 1, p. 1–11, 2016.
- HASOFER, Abraham M; LIND, Niels C. Exact and invariant second-moment code format. **Journal of the Engineering Mechanics division**, ASCE, v. 100, n. 1, p. 111–121, 1974.
- HAYKIN, Simon. **Redes neurais: princípios e prática**. [*S.l.*]: Bookman Editora, 2007.
- HESTENES, Magnus R; STIEFEL, Eduard *et al.* Methods of conjugate gradients for solving linear systems. **Journal of research of the National Bureau of Standards**, v. 49, n. 6, p. 409–436, 1952.
- HIRIART-URRUTY, J.B.; LEMARECHAL, C. **Convex Analysis and Minimization Algorithms I: Fundamentals**. [*S.l.*]: Springer Berlin Heidelberg, 2013. (Grundlehren der mathematischen Wissenschaften). ISBN 9783662027967.
- HOYLE, Nicola. **Automated multi-stage geometry parameterization of internal fluid flow applications**. 2006. Tese (Doutorado) – University of Southampton.
- JAMIL, Momin; YANG, Xin-She. A literature survey of benchmark functions for global optimisation problems. **International Journal of Mathematical Modelling and Numerical Optimisation**, Inderscience Publishers Ltd, v. 4, n. 2, p. 150–194, 2013.

JOHNSON, Mark E; MOORE, Leslie M; YLVISAKER, Donald. Minimax and maximin distance designs. **Journal of statistical planning and inference**, Elsevier, v. 26, n. 2, p. 131–148, 1990.

JONES, Donald R; SCHONLAU, Matthias; WELCH, William J. Efficient global optimization of expensive black-box functions. **Journal of Global optimization**, Springer, v. 13, n. 4, p. 455–492, 1998.

JOURNEL, Andre G; HUIJBREGTS, Charles J. **Mining geostatistics**. [S.l.]: Academic press London, 1978. v. 600.

KENNEDY, James; EBERHART, Russell. Particle swarm optimization. *In: IEEE. PROCEEDINGS of ICNN'95-International Conference on Neural Networks*. [S.l.: s.n.], 1995. P. 1942–1948.

KIEFER, Jack. Sequential minimax search for a maximum. **Proceedings of the American mathematical society**, JSTOR, v. 4, n. 3, p. 502–506, 1953.

KIM, Byeong-Soo; LEE, Yong-Bin; CHOI, Dong-Hoon. Comparison study on the accuracy of metamodeling technique for non-convex functions. **Journal of Mechanical Science and Technology**, Springer, v. 23, n. 4, p. 1175–1181, 2009.

KIRKPATRICK, Scott; GELATT, C Daniel; VECCHI, Mario P. Optimization by simulated annealing. **science**, American association for the advancement of science, v. 220, n. 4598, p. 671–680, 1983.

KRIGE, Daniel G. A statistical approach to some basic mine valuation problems on the Witwatersrand. **Journal of the Southern African Institute of Mining and Metallurgy**, Southern African Institute of Mining e Metallurgy, v. 52, n. 6, p. 119–139, 1951.

KRISHNAMURTHY, Thiagarajan. Comparison of response surface construction methods for derivative estimation using moving least squares, kriging and radial basis functions. *In: 46TH AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*. [S.l.: s.n.], 2005. P. 1821.

LIEM, Rhea P; MADER, Charles A; MARTINS, Joaquim RRA. Surrogate models and mixtures of experts in aerodynamic performance prediction for aircraft mission analysis. **Aerospace Science and Technology**, Elsevier, v. 43, p. 126–151, 2015.

LIU, Boyang; HAFTKA, Raphael T; AKGÜN, Mehmet A. Two-level composite wing structural optimization using response surfaces. **Structural and Multidisciplinary Optimization**, Springer, v. 20, n. 2, p. 87–96, 2000.

LORENZ, Edward N. Deterministic Nonperiodic Flow *Journal of the Atmospheric Sciences* Vol. 20, 1963.

- LUERSEN, Marco A; LE RICHE, Rodolphe. Globalized Nelder–Mead method for engineering optimization. **Computers & structures**, Elsevier, v. 82, n. 23-26, p. 2251–2260, 2004.
- MATHERON, Georges. Principles of geostatistics. **Economic geology**, Society of Economic Geologists, v. 58, n. 8, p. 1246–1266, 1963.
- MATLAB. **version 9.5.0 (R2018b)**. Natick, Massachusetts: The MathWorks Inc., 2018.
- MEZURA-MONTES, Efrén; COELLO, CA Coello; LANDA-BECERRA, Ricardo. Engineering optimization using simple evolutionary algorithm. *In: IEEE. PROCEEDINGS. 15th IEEE international conference on tools with artificial intelligence. [S.l.: s.n.], 2003. P. 149–156.*
- MONTGOMERY, Douglas C; RUNGER, George C. **Applied statistics and probability for engineers**. [S.l.]: John Wiley & Sons, 2010.
- MORRIS, Max D; MITCHELL, Toby J. Exploratory designs for computational experiments. **Journal of statistical planning and inference**, Elsevier, v. 43, n. 3, p. 381–402, 1995.
- MUELLER, Juliane. MATSumoto: the MATLAB surrogate model toolbox for computationally expensive black-box global optimization problems. **arXiv preprint arXiv:1404.4261**, 2014.
- MÜLLER, Juliane. SOCEMO: surrogate optimization of computationally expensive multiobjective problems. **INFORMS Journal on Computing**, INFORMS, v. 29, n. 4, p. 581–596, 2017.
- MÜLLER, Juliane; KRITYAKIERNE, Tipaluck; SHOEMAKER, Christine A. SO-MODS: Optimization for high dimensional computationally expensive multi-modal functions with surrogate search. *In: IEEE. 2014 IEEE Congress on Evolutionary Computation (CEC). [S.l.: s.n.], 2014. P. 1092–1099.*
- MÜLLER, Juliane; SHOEMAKER, Christine A; PICHÉ, Robert. SO-MI: A surrogate model algorithm for computationally expensive nonlinear mixed-integer black-box global optimization problems. **Computers & Operations Research**, Elsevier, v. 40, n. 5, p. 1383–1400, 2013.
- MÜLLER, Juliane; WOODBURY, Joshua D. GOSAC: global optimization with surrogate approximation of constraints. **Journal of Global Optimization**, v. 69, n. 1, p. 117–136, set. 2017. ISSN 1573-2916.
- MYERS, Donald E. Matrix formulation of co-kriging. **Journal of the International Association for Mathematical Geology**, Springer, v. 14, n. 3, p. 249–257, 1982.

NASCENTES, Fábio Felipe dos Santos *et al.* Contribuições à eficiência da otimização global estocástica adaptativa, 2019.

NASCENTES, Fábio; LOPEZ, Rafael; FANCELLO, Eduardo. **Contribuições à eficiência da otimização global estocástica adaptativa**. [*S.l.: s.n.*], abr. 2019. DOI: 10.13140/RG.2.2.17217.81766.

NASCENTES, Fábio *et al.* Stochastic Tunneling for Improving the Efficiency of Stochastic Efficient Global Optimization. *In: SPRINGER. WORLD Congress on Global Optimization*. [*S.l.: s.n.*], 2019. P. 238–246.

NORONHA, Gustavo Dela Bruna. AKMCS e suas variações: procedimento de confiabilidade utilizando Kriging e simulação de Monte Carlo, 2018.

OLEA, Ricardo A. **Geostatistical glossary and multilingual dictionary**. [*S.l.*]: Oxford University Press on Demand, 1991.

POHLHEIM, Hartmut. Examples of objective functions. **Retrieved**, v. 4, n. 10, p. 2012, 2007.

QIN, Chao; KLABJAN, Diego; RUSSO, Daniel. Improving the expected improvement algorithm. *In: ADVANCES in Neural Information Processing Systems*. [*S.l.: s.n.*], 2017. P. 5381–5391.

RAMOS, Karen Patricia Guevara; PACHECO, Marco Aurelio Cavalcanti. Propagação de incertezas via expansão por caos polinomial em simulação de reservatórios de petróleo. **Mestrado, Departamento de Engenharia Elétrica, PUC-Rio**, 2014.

REGIS, Rommel G; SHOEMAKER, Christine A. Combining radial basis function surrogates and dynamic coordinate search in high-dimensional expensive black-box optimization. **Engineering Optimization**, Taylor & Francis, v. 45, n. 5, p. 529–555, 2013.

RENARD, Philippe; ALCOLEA, Andres; GINGSBOURGER, D. Stochastic versus deterministic approaches. *In: ENVIRONMENTAL Modelling: Finding Simplicity in Complexity, Second Edition* (eds J. Wainwright and M. Mulligan). [*S.l.*]: Wiley Online Library, 2013. P. 133–149.

RIBEIRO, Celso da Cruz Carneiro. **Heurísticas para o Problema das p-Mediana Conectadas**. 2006. Tese (Doutorado) – PUC-Rio.

RONCHI, Carlos Henrique Venturi. Estudo matemático do reconhecimento de caracteres.

SACKS, Jerome *et al.* Design and analysis of computer experiments. **Statistical science**, JSTOR, p. 409–423, 1989.

SANTANA GOMES, Wellison José de. Structural Reliability Analysis Using Adaptive Artificial Neural Networks. **ASCE-ASME J Risk and Uncert in Engrg Sys Part B Mech Engrg**, v. 5, n. 4, set. 2019. 041004. ISSN 2332-9017. DOI: 10.1115/1.4044040. eprint: https://asmedigitalcollection.asme.org/risk/article-pdf/5/4/041004/5873757/risk_005_04_041004.pdf. Disponível em: <https://doi.org/10.1115/1.4044040>.

SASENA, Michael James. **Flexibility and efficiency enhancements for constrained global design optimization with kriging approximations**. 2002. Tese (Doutorado) – University of Michigan Ann Arbor.

SCHONLAU, Matthias; WELCH, William J; JONES, Donald R. Global versus local search in constrained optimization of computer models. **Lecture Notes-Monograph Series**, JSTOR, p. 11–25, 1998.

SHAN, Songqing; WANG, G Gary. Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. **Structural and multidisciplinary optimization**, Springer, v. 41, n. 2, p. 219–241, 2010.

SIMPSON, Timothy W *et al.* Kriging models for global approximation in simulation-based multidisciplinary design optimization. **AIAA journal**, v. 39, n. 12, p. 2233–2241, 2001.

TALBI, El-Ghazali. **Metaheuristics: from design to implementation**. [S.l.]: John Wiley & Sons, 2009. v. 74.

TORII, André Jacomel; LOPEZ, Rafael Holdorf. Reliability analysis of water distribution networks using the adaptive response surface approach. **Journal of Hydraulic Engineering**, American Society of Civil Engineers, v. 138, n. 3, p. 227–236, 2012.

TRAVESSA, Sheila Santisi *et al.* Uso de redes neurais artificiais como metamodelo na otimização por algoritmo PSO (particle swarm optimization’) em problemas de mapeamento eletromagnético de ambientes, 2017.

VANARET, Charlie. Hybridization of interval methods and evolutionary algorithms for solving difficult optimization problems. **arXiv preprint arXiv:2001.11465**, 2020.

VIEIRA, Carlos Eduardo Costa; SOUZA, Reinaldo de Castro e; RIBEIRO, Celso Carneiro. **Um estudo comparativo entre três geradores de números aleatórios**. [S.l.]: PUC, 2004.

WHITLEY, Darrell *et al.* Evaluating evolutionary algorithms. **Artificial intelligence**, Elsevier, v. 85, n. 1-2, p. 245–276, 1996.

WILLIAMS, Brian J *et al.* Sequential design of computer experiments for constrained optimization. *In: STATISTICAL Modelling and Regression Structures*. [S.l.]: Springer, 2010. P. 449–472.

YANG, Xin-She. A new metaheuristic bat-inspired algorithm. *In: NATURE inspired cooperative strategies for optimization (NICSO 2010)*. [S.l.]: Springer, 2010. P. 65–74.

_____. Firefly algorithms for multimodal optimization. *In: SPRINGER*.

INTERNATIONAL symposium on stochastic algorithms. [S.l.: s.n.], 2009. P. 169–178.

ZABINSKY, Z.B. **Stochastic Adaptive Search for Global Optimization**. [S.l.]: Springer US, 2013. (Nonconvex Optimization and Its Applications). ISBN 9781441991829. Disponível em: <https://books.google.com.br/books?id=rDjaBwAAQBAJ>.

ZEVIANI, Walmes Marques; JÚNIOR, PJR; BONAT, Wagner Hugo. Modelos de regressão não linear. **Laboratório de Estatística e Geoinformação. Departamento de Estatística. UFPR**, 2013.