UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTOS ACADÊMICOS DE ELETRÔNICA E MECÂNICA
CURSO SUPERIOR DE TECNOLOGIA EM MECATRÔNICA INDUSTRIAL

ADRIANO GONÇALVES DOS PASSOS

# DESENVOLVIMENTO E PUBLICAÇÃO DE UM PACOTE COMPUTACIONAL DE OTIMIZAÇÃO MULTIOBJETIVO BASEADO NO METAMODELO DE KRIGING

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA
2017

ADRIANO GONÇALVES DOS PASSOS

# DESENVOLVIMENTO E PUBLICAÇÃO DE UM PACOTE COMPUTACIONAL DE OTIMIZAÇÃO MULTIOBJETIVO BASEADO NO METAMODELO DE KRIGING

Trabalho de Conclusão de Curso apresentada à disciplina de Trabalho de Conclusão de Curso do curso de Tecnologia Mecatrônica Industrial da Universidade Tecnológica Federal do Paraná como requisito parcial à aprovação da disciplina.

Orientador: Prof. Dr. Marco Antônio Luersen

CURITIBA
2017

**TERMO DE APROVAÇÃO**

ADRIANO GONÇALVES DOS PASSOS

# DESENVOLVIMENTO E PUBLICAÇÃO DE UM PACOTE COMPUTACIONAL DE OTIMIZAÇÃO MULTIOBJETIVO BASEADO NO METAMODELO DE KRIGING

Este trabalho de conclusão de curso foi apresentado no dia 24 de novembro de, como requisito parcial para obtenção do título de Tecnólogo em Mecatrônica Industrial, outorgado pela Universidade Tecnológica Federal do Paraná. O aluno foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof. Dr. Milton Luiz Polli
Coordenador de Curso
Departamento Acadêmico de Mecânica

_____
Prof. M.Sc. Sérgio Moribe
Responsável pela Atividade de Trabalho de Conclusão de Curso
Departamento Acadêmico de Eletrônica

**BANCA EXAMINADORA**

_____
Profa. Dra. Patricia Sánez Pacheco
UTFPR

_____
Prof. Dr. Leandro Magatão
UTFPR

_____
Prof. Dr. Marco Antônio Luersen
Orientador - UTFPR

“A Folha de Aprovação assinada encontra-se na Coordenação do Curso”

# RESUMO

PASSOS, Adriano Gonçalves dos. **Desenvolvimento e publicação de um pacote computacional de otimização multiobjetivo baseado no metamodelo de kriging**. 95 f. Trabalho de Conclusão de Curso (Curso Superior de Tecnologia em Mecatrônica Industrial), Departamentos Acadêmicos de Eletrônica e Mecânica, Universidade Tecnológica Federal do Paraná. Curitiba, 2017.

A busca constante por projetos cada vez melhores torna a otimização uma ferramenta usada amplamente em diferentes ramos da engenharia. Com o crescente aumento da complexidade das demandas, múltiplos e conflitantes critérios de desempenho devem ser otimizados. Além disso, restrições críticas como de segurança, devem ser respeitadas. Nesse contexto, a otimização multiobjetivo apresenta abordagens sistemáticas para tais desafios de engenharia. Com início aproximadamente no começo dos anos 2000, inúmeras heurísticas vem sendo desenvolvidas para atender essa demanda. Por se basearem principalmente em métodos de evolução artificial, usualmente tais algoritmos necessitam de dezenas de milhares de cálculos das funções objetivo e de restrições. Comumente, em problemas reais de engenharia, as funções objetivos e restrições são calculadas com uso de ferramentas computacionais complexas como elementos finitos ou dinâmica dos fluidos computacional, onde cada avaliação pode durar uma quantidade significativa de tempo. Em problemas de otimização mono-objetivo, uma solução padrão adotada por projetistas para lidar com funções objetivos de alto custo computacional é a otimização global eficiente (EGO, do inglês *efficient global optimization*). Esse algoritmo baseia-se na construção e melhoria iterativa de um metamodelo (quase sempre o metamodelo de Kriging). Nesse contexto, os metamodelos tem como função descrever, de maneira aproximada, a relação entre o espaço de variáveis de projetos e o espaço dos objetivos. Como os metamodelos são, em geral, de baixo custo computacional, algoritmos menos eficientes de busca podem ser utilizados para encontrar projetos otimizados. Por maior sucesso que o EGO apresente na solução de problemas mono-objetivo, não há um algoritmo padrão que se destaque na otimização multiobjetivo de alto custo computacional. Os primeiros algoritmos, propostos a partir do ano 2005, são simples adaptações do EGO para o ambiente multiobjetivo. Publicações mais recentes (2011 – 2017) apresentam indicadores de otimalidade mais robustos e algoritmos mais eficientes. Contudo, a partir da revisão bibliográfica realizada, algumas oportunidades foram observadas. Os resultados apresentados neste projeto de conclusão de curso se mostram inovadores e robustos. Um novo algoritmo de otimização multiobjetivo proposto, baseado na minimização sequencial da variância da frente de Pareto predita, apresenta resultados superiores a técnicas convencionais nos problemas testados. Por final, o pacote computacional publicado tem boa recepção tendo média de 120 downloads mensais.

**Palavras-chave:** Otimização Multiobjetivo, Kriging, Pacote Computacional, R.

# LISTA DE SIGLAS

| | |
|---|---|
| DM | Projetista – *Decision Maker* |
| SPEA2 | *Strength Pareto Evolutionary Algorithm 2* |
| PESA-II | *Pareto Envelope Based Selection Algorithm II* |
| NSGA-II | *nondominating sorting genetic algorithm II* |
| SMS-EMOA | *S-metric Selection Evolutionary Multiobjective Optimization Algorithm* |
| GP | Processo Gaussiano – *Gaussian Process* |
| EI | Melhoria Esperada – *Expected Improvement* |
| PI | Probabilidade de Melhoria – *Probability of Improvement* |
| MOO | Otimização Multiobjetivo – *multiobjective optimization* |
| MOEA | Algoritmos Evolucionários de Otimização Multiobjetivo – *Multiobjective Evolutionary Algorithms* |
| KKTPM | métrica de proximidade KKT – *Karush-Kuhn-Tucker Proximity Measure* |
| EGO | Otimização Global Eficiente – *Efficient Global Optimization* |
| EHI | Melhoria esperada do hiper-volume – *Expected Hypervolume Improvement* |
| ASF | *Achievement Scalarization Function* |
| ZDT3 | Função de teste: Zitzler–Deb–Thiele's N. 3 |
| DTLZ4 | Função de teste: Deb–Thiele-Laumanns-Zitzler's -N. 4 |

# LISTA DE FIGURAS

# SUMÁRIO

# 1 INTRODUÇÃO

## 1.1 NOTAÇÕES UTILIZADAS

Usualmente utiliza-se $f(x)$ para designar uma função, e em diferentes áreas do conhecimento essa notação pode possuir significados intrínsecos diferentes. No presente texto isso é especialmente relevante pois, no ramo da otimização, $f(x)$ é comumente utilizado para representar as funções objetivo, ou de interesse do estudo. Contudo, no estudo das probabilidades e de variáveis aleatórias, é comum a utilização de $f_X(x)$ ou apenas $f(x)$ para expressar a função de densidade de probabilidades de uma variável aleatória $X$. Além disso, é comum representar a função de distribuição da variável aleatória $X$ como $F_X(x)$ ou apenas $F(x)$.

Outro conflito de notações observado é o uso de letras maiúsculas. Em otimização, é comum o uso de letras maiúsculas para representação de matrizes ou parâmetros, enquanto em probabilidade é comum o uso de letras maiúsculas para representar variáveis aleatórias.

Nesse sentido, o presente trabalho utiliza a convenção apresentada na Tabela 1.1.

Tabela 1.1 – Notação utilizada no presente trabalho – Desambiguação.

| Símbolo | Descrição |
|---|---|
| $x$ | escalar |
| $\mathbf{x}$ | vetor |
| $\mathbf{X}$ | matriz |
| $\mathcal{X}$ | variável aleatória |
| $\boldsymbol{\mathcal{X}}$ | vetor aleatório |
| $\mathcal{X}(x)$ | processo estocástico |
| $\boldsymbol{\mathcal{X}}(x)$ | processo estocástico vetorial |
| $f(x)$ | função escalar |
| $\mathbf{f}(x)$ | função vetorial |
| $\phi_{\mathcal{X}}(x)$ | função densidade de probabilidade da variável aleatória $\mathcal{X}$ |
| $\Phi_{\mathcal{X}}(x)$ | função de distribuição da variável aleatória $\mathcal{X}$ |

## 1.2 ORGANIZAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO

No Capítulo 1, é exposto um breve panorama e os principais conceitos necessários para o entendimento do texto. Contudo, com o objetivo de tornar a leitura mais eficiente, detalhes e deduções matemáticas são indicados em literatura citada. O Capítulo 2 inclui uma revisão bibliográfica para investigar o estado da arte nos temas de

otimização multiobjetivo com base no modelo de Kriging. No Capítulo 3 é apresentada a metodologia adotada para a construção do pacote computacional, e o Capítulo 4 traz os resultados bem como as considerações finais.

No Apêndice A, pode ser vista a documentação completa do pacote e no Apêndice B a vinheta explicativa do mesmo. Os Apêndices de C, D e E, contém artigos publicados pelo autor relacionados ao presente trabalho.

## 1.3 OBJETIVOS E JUSTIFICATIVA DA PESQUISA

Dentro do contexto apresentado, ***o objetivo do projeto de conclusão de curso é desenvolver e publicar um pacote de otimização multiobjetivo com base no metamodelo de Kriging que seja eficiente, robusto e que possa ser empregado na otimização de problemas reais de engenharia***.

A importância da utilização da otimização multiobjetivo em projetos de engenharia é clara. Em um mercado competitivo como o atual e com o crescente aumento da demanda por eficiência, deseja-se projetos/produtos que possuam um conjunto ótimo de características. O interesse nesse ramo de pesquisa torna-se evidente pelo aumento do número de publicações de OM em relação ao número de pesquisas de otimização como um todo (Figura 1.1).

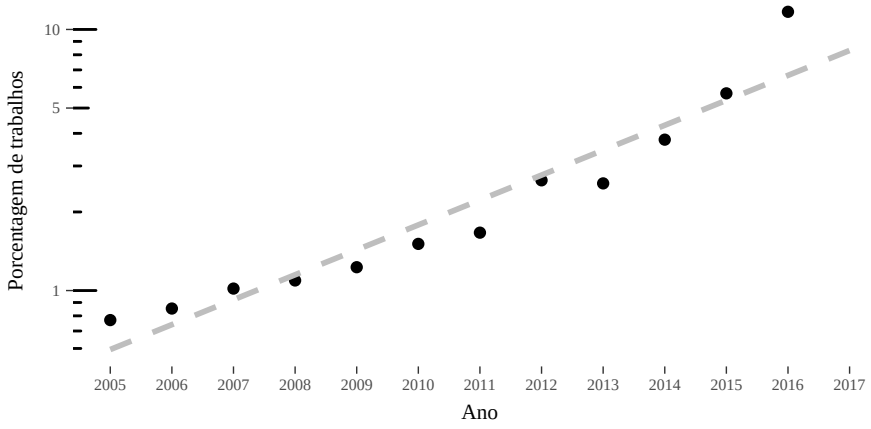



Figura 1.1 – Porcentagem de publicações com as com as palavras "*multi-objective optimization*" em relação aquelas com a palavra "*optimization*".
(**fonte:** https://scholar.google.com.br/)

Devido ao aumento constante no uso de modelos computacionais sofisticados, empregados para simulação de como produtos ou sistemas se comportarão. Por exemplo, na indústria automotiva, é possível simular o grau de danos que passageiros sofreriam em uma colisão, o nível de vibração e ruído decorrente do tráfego em um determinado tipo de estrada, e até mesmo previsões confiáveis do ciclo de vida do veículo. O resultado desse avanço é o aumento na flexibilidade e velocidade de

desenvolvimento do projeto. Contudo, um dos maiores obstáculos para o uso de tais ferramentas com algoritmos clássicos de otimização é o elevado tempo computacional (e em algumas vezes a falta de uma função gradiente). Nesse contexto, o uso de metamodelos é de extrema relevância para a redução do custo computacional, tornando o processo de otimização possível em uma quantidade de tempo realista. Além disso, o uso de metamodelos oferece grande flexibilidade de exploração de projetos alternativos enquanto mantém baixo o custo e tempo de desenvolvimento.

## 1.4 OTIMIZAÇÃO MULTIOBJETIVO

Problemas complexos de engenharia quase sempre estão relacionados a um balanço de múltiplos critérios de desempenho que, em geral, são conflitantes entre si. Por exemplo, requisitos comumente relacionados a projetos aeronáuticos são: peso, custo, robustez, performance, envelope de voo, comprimento de decolagem, agressão ao meio ambiente, manobrabilidade, entre outros. Esses objetivos são claramente conflitantes e uma solução que melhore um dos critérios provavelmente causará impacto negativo em outros. Em geral, o que o projetista (na literatura de otimização multiobjetivo: *decision maker* (DM)) deve fazer é balancear os objetivos conflitantes e decidir qual solução atende da melhor maneira possível a combinação desses (TORENBEEK, 2013). Contudo, a escolha de uma solução que atenda de maneira satisfatória todos os objetivos não é simples. O mapeamento entre o espaço de decisão (variáveis de entrada) e o espaço dos objetivos (respostas) é, em geral, altamente não linear. Sendo assim, encontrar as variáveis de entrada que levem à uma combinação ótima de respostas é, quando possível, de elevada dificuldade (FORRESTER et al., 2008).

Em um problema de otimização multiobjetivo não-trivial, não existe um único projeto que maximize simultaneamente todos os critérios. Nesse caso, diz-se que o problema possui objetivos conflitantes e existe um conjunto (possivelmente infinito) de soluções ótimas. Esse conjunto é comumente chamado de conjunto (ou frente) de Pareto. Assim, cada um dos projetos do conjunto é ótimo em algum sentido, sem levar em consideração a relevância de cada objetivo. Segundo Forrester et al. (2008), de maneira mais formal, uma frente ou conjunto de Pareto contém apenas projetos que são suficientemente otimizados tal que: para melhorar o desempenho de qualquer projeto, em qualquer critério, o desempenho em pelo menos um dos demais objetivos é reduzido. Em outras palavras, os projetos contidos em um conjunto de Pareto são considerados *não dominados*, de modo que não há nenhum projeto que seja melhor em mais de um objetivo quando comparado a outro.

Para dois ou três objetivos, um conjunto de Pareto pode ser visualizado em um gráfico de dispersão (*scatter plot*) que tenha em seus eixos a resposta para cada obje-

tivo (objetivo A *versus* objetivo B, por exemplo). A Figura 1.2 mostra o exemplo de uma frente de Pareto de um problema de minimização dos objetivos A e B. Nela, os pontos que fazem parte do conjunto de Pareto (pontos não dominados) são representados por pontos cheios e os que não fazem parte (pontos dominados) por pontos vazios. A linha cheia representa a frente de Pareto e a região hachurada com linhas cheias mostra a parte do espaço objetivo onde possíveis projetos aumentariam o conjunto de Pareto sem excluir (dominar) nenhum ponto já existente (i.e., aumentariam o número de projetos da frente de Pareto). Já a região hachurada em tracejado representa a parte do espaço que um possível projeto dominaria pelo menos um ponto da frente atual (i.e., obteria melhor desempenho em ambos os critérios). Nessa região, pode ou não existirem projetos factíveis. No caso de não existirem, os pontos não dominados são ditos fazerem parte da frente real de Pareto.

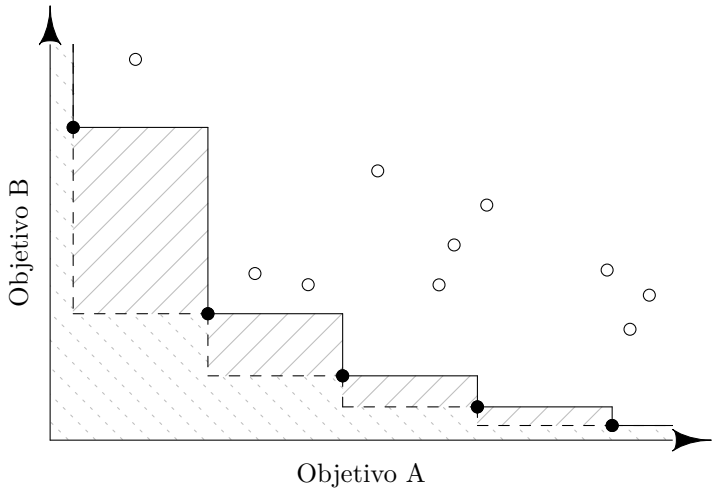



Figura 1.2 – Amostras em uma otimização multiobjetivo com um conjunto de Pareto de cinco pontos. A linha cheia representa a frente de Pareto.

Com o aumento do número de objetivos, a visualização do conjunto de Pareto se torna problemática. Fieldsend e Everson (2013) trazem uma revisão de técnicas de visualização que se baseiam em projeções e correlações com *scatterplots* bidimensionais. Já em Blasco et al. (2008), é mostrada uma nova técnica de visualização baseada em curvas de níveis, que possui a vantagem de sincronização do espaço decisão com o espaço de projeto.

O estudo de problemas com múltiplos objetivos pode ser visto em duas principais perspectivas: construção ou utilização/visualização de frentes de Pareto. No presente trabalho, o enfoque é dado unicamente no processo de construção dos conjuntos de Pareto.

O problema de otimização multiobjetivo pode ser denotado em uma forma

padrão como

$$\begin{aligned} &\min\left(f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})\right) \\ &\text{tal que } \mathbf{x} \in D, \end{aligned} \tag{1.1}$$

onde $k \geq 2$ é o número de objetivos e $D$ o espaço decisão ou conjunto de projetos factíveis, definido por

$$D = \begin{cases} \mathbf{x} \mid h_j(\mathbf{x}) = 0, & (j = 1, \ldots, p); \\ \mathbf{x} \mid g_i(\mathbf{x}) \leq 0, & (i = 1, \ldots, m). \end{cases} \tag{1.2}$$

De maneira alternativa, os objetivos a serem minimizados podem ser definidos como uma função vetorial

$$\mathbf{f} : D \to \mathbb{R}^k, \mathbf{f}(\mathbf{x}) = \left(f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})\right)^\top. \tag{1.3}$$

Como dito anteriormente, em geral não há solução única (trivial) para o problema multiobjetivo. Em termos matemáticos, um projeto $\mathbf{x}^{(1)}$ diz-se dominante em relação outro projeto $\mathbf{x}^{(2)}$, se

1. $\forall\, i \in \{1, 2, \ldots, k\}, f_i\left(\mathbf{x}^{(1)}\right) \leq f_i\left(\mathbf{x}^{(2)}\right)$ e
2. $\exists\, j \in \{1, 2, \ldots, k\}, f_j\left(\mathbf{x}^{(1)}\right) < f_j\left(\mathbf{x}^{(2)}\right)$.

Para auxiliar no ranqueamento de soluções e/ou conjuntos de soluções, vários critérios foram propostos, entre as mais populares destacando-se o hiper-volume dominado e o indicador épsilon (EMMERICH et al., 2011; SVENSON, 2011). Ambos os indicadores podem ser vistos na Figura 1.3, para um projeto hipotético adicionado ao mesmo exemplo da Figura 1.2 (representado por $\boxplus$). A melhoria do hiper-volume dominado (Fig. 1.3 a) é o incremento de hiper-volume contido entre a frente de Pareto e um ponto de referência[1] (indicado por $\oplus$) no espaço dos objetivos, quando um novo ponto não-dominado (indicado por $\boxplus$) é adicionado. O hiper-volume dominado original é indicado pela área hachurada com linhas cheias e o incremento no hiper-volume dominado pela área hachurada com linhas tracejadas. O funcionamento do indicador épsilon pode ser visto na Figura 1.3 b). O indicador possui valor igual ao menor escalar que deve ser adicionado aos componentes de um novo projeto (no espaço dos objetivos) de modo que esse passe a ser dominado pela frente atual.

Diversas heurísticas foram desenvolvidas para resolver problemas com múltiplos objetivos. Entre aquelas de maior sucesso podem ser destacadas: SPEA2[2] (ZITZ-

[1] O ponto de referência pode ser definido arbitrariamente desde que dentro da região dominada. Contudo, usualmente escolhe-se o maior para cada objetivo obtido considerando todos os projetos amostrados.

[2] *Strength Pareto evolutionary algorithm 2.*

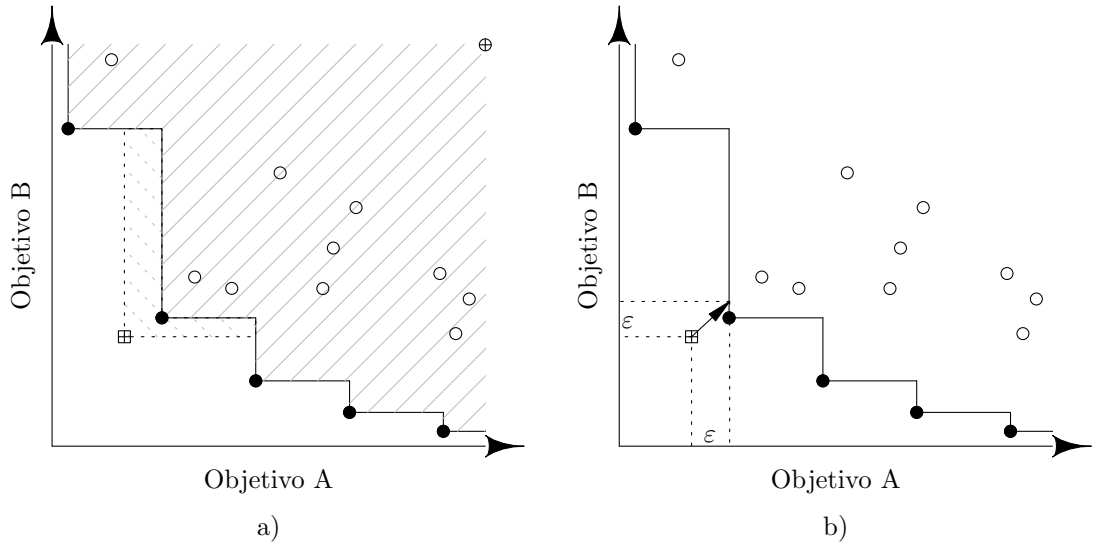



Figura 1.3 – Comparação entre dois indicadores de otimalidade de conjuntos de Pareto: hiper-volume dominado (a) e indicador épsilon (b)

LER et al., 2001), PESA-II[3] (CORNE et al., 2001), NSGA-II[4] (DEB et al., 2002b) e SMS-EMOA[5] (EMMERICH et al., 2005; BEUME et al., 2007). Segundo Deb e Jain (2014), tais heurísticas são apenas indicadas quando o número de objetivos é moderado (2 a 3). Com intuito de preencher essa lacuna, em um trabalho divido em dois artigos (DEB; JAIN, 2014; JAIN; DEB, 2014), o algoritmo NSGA-III é apresentado. Esse ramo da otimização multiobjetivo é comumente chamado de otimização com muitos objetivos (*many-objective optimization*). Apesar de altamente robustos, esses algoritmos possuem dificuldades na otimização de funções de alto custo computacional pois necessitam de grande número de avaliações das funções objetivo.

## 1.5 EXPERIMENTOS COMPUTACIONAIS E OS PROCESSOS GAUSSIANOS

A modelagem computacional é hoje uma ferramenta padrão no desenvolvimento de projetos de engenharia. Usada principalmente como uma alternativa rápida e barata quando comparada ao processo experimental, sendo, em alguns casos, a única alternativa viável (por exemplo modelagem climática). Mesmo com a constante melhoria de softwares e hardwares, a crescente complexidade dos problemas atuais acarreta em altos tempos de processamento. Assim, o estudo de diferentes concepções de um projeto baseado na exploração exaustiva das possibilidades é, em geral, impossível sob restrições realísticas de tempo.

Chama-se de *experimento computacional* o processo de realização de simulações computacionais à um dado número de diferentes configurações de projetos.

[3] *Pareto envelope based selection algorithm II.*
[4] *Nondominating sorting genetic algorithm II.*
[5] *S-metric selection evolutionary multiobjective optimization algorithm.*

As configurações, por sua vez, são definidas alterando-se os valores das variáveis de projeto. Além disso, normalmente considera-se que os modelos computacionais são determinísticos, ou seja, para um dado conjunto fixo de entradas, um modelo computacional retorna sempre a mesma resposta. Nesse contexto, o principal desafio é escolher quais configurações de projeto devem ser modeladas para que o orçamento computacional seja gasto da maneira mais eficiente possível.

Apesar de similaridades com experimentos físicos, os experimentos computacionais possuem diversas características únicas. Em geral, como já comentado, são tipicamente determinísticos, representam de maneira aproximada o fenômeno físico verdadeiro e são essencialmente funções do tipo *caixa preta* (no inglês, *black-box functions*). Funções caixa preta significam que não se conhece de maneira fechada o funcional que relaciona as entradas e saídas do experimento. Portanto, a única informação que se tem sobre o problema são respostas avaliadas em um número finito de observações. Devido a essa característica, os experimentos computacionais podem ser vistos sob o aspecto probabilístico. Ou seja, o comportamento da função em pontos não observados pode ser apenas *estimado*.

Processos gaussianos (GP) podem ser empregados para representar esse funcional desconhecido. Na teoria das probabilidades, um processo gaussiano é um modelo estatístico onde observações ocorrem em um domínio contínuo (usualmente o tempo e/ou espaço). Em um GP, cada ponto do espaço de entrada $\mathbf{x} \in \mathbb{R}^d$ (projeto ou decisão) é associado a uma variável aleatória $\mathcal{Y}(\mathbf{x})$. Como o processo é definido em um domínio contínuo, a função de distribuição conjunta de todas essas (infinitas) variáveis aleatórias é uma distribuição sobre funções com domínio contínuo. GPs podem ser vistos como uma generalização infinito-dimensional de distribuições normais multivariadas (CHAN, 2013).

Sem dúvidas, o processo gaussiano mais famoso, tanto que por vezes seu nome é usado como sinônimo, é o Kriging. Originado no ramo das geociências (KRIEG, 1951) e ponto de partida para o ramo da geo-estatística (MATHERON, 1963), Kriging, em resumo, é um método de interpolação espacial. Considerando uma função caixa preta (por motivos didáticos apenas escalar) $y : D \subset \mathbb{R}^d \to \mathbb{R}$ como sendo uma realização de um processo estocástico $\mathcal{Y}(\mathbf{x}), \mathbf{x} \in D$, Kriging consiste em prever os valores para $y(\mathbf{x})$ com base na distribuição condicional de $\mathcal{Y}(\mathbf{x})$ dado um conjunto finito de $n$-observações de $y(\mathbf{x})$ (ROUSTANT et al., 2012). A construção da estrutura Bayesiana de predição é mostrada em profundidade em Sacks et al. (1989), Jones et al. (1998), Forrester et al. (2008), Roustant et al. (2012) e Scheuerer et al. (2013) e em termos práticos, um metamodelo de Kriging para uma função mono-objetivo, pode ser simplesmente resumido por

$$\mathcal{Y}(\mathbf{x})|(\mathcal{Y}(\mathbf{X}) = \mathbf{y}) \sim \mathcal{N}\left(\mu = m(\mathbf{x}), \sigma^2 = s^2(\mathbf{x})\right), \tag{1.4}$$

onde $\mathcal{Y}(\mathbf{x})|(Y(\mathbf{X}) = \mathbf{y})$ (ou em uma notação simplificada $\mathcal{Y}|\mathbf{y}$) é o processo gaussiano condicional que representa o metamodelo de Kriging; $\mathbf{x} = \{x_1, \ldots, x_d\} \in D \subset \mathbb{R}^d$ é um projeto qualquer dentro do domínio de projeto; $\mathbf{X} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\}$ é a matriz composta por todos os vetores de projeto onde $y(\mathbf{x})$ já foi avaliado; $\mathbf{y} = \{y(\mathbf{x}^{(1)}), \ldots, y(\mathbf{x}^{(n)})\}^\top$ são as correspondentes respostas; $m(\mathbf{x})$ e $s^2(\mathbf{x})$ são funções que definem a média e a variância do processo estocástico para qualquer $\mathbf{x} \in D$.

Devido a construção do modelo, algumas propriedades são muito úteis à análise dos experimentos computacionais, em especial à otimização. Tais propriedades são apresentadas a seguir em forma de teoremas.

**Teorema 1.1.** A média do processo gaussiano interpola os dados observados. Ou seja:

$$m(\mathbf{X}) = \mathbf{y}. \tag{1.5}$$

*Demonstração.* Pela definição de média:

$$m(\mathbf{x}) = E\left[\mathcal{Y}(\mathbf{x})|(\mathcal{Y}(\mathbf{X}) = \mathbf{y})\right]. \tag{1.6}$$

Portanto

$$\begin{aligned} m(\mathbf{X}) &= E\left[\mathcal{Y}(\mathbf{X})|(\mathcal{Y}(\mathbf{X}) = \mathbf{y})\right], \\ &= E\left[\mathbf{y}\right], \\ &= \mathbf{y}. \end{aligned} \tag{1.7}$$

□

**Teorema 1.2.** A variância $s^2(\mathbf{x})$ é sempre não-negativa, e sendo nula nos pontos experimentais. Ou seja:

$$s^2(\mathbf{x}) \geq 0 \tag{1.8}$$

e

$$s^2(\mathbf{X}) = \mathbf{0}. \tag{1.9}$$

*Demonstração.* Pela definição de variância:

$$\mathrm{Var}[\mathcal{Y}] = \mathrm{E}\left[(\mathcal{Y} - \mathrm{E}[\mathcal{Y}])^2\right] \geq 0. \tag{1.10}$$

Portanto,

$$s^2(\mathbf{x}) = \mathrm{Var}[\mathcal{Y}(\mathbf{x})] \geq 0 \tag{1.11}$$

e

$$\begin{aligned} s^2(\mathbf{X}) &= \mathrm{Var}\left[\mathcal{Y}(\mathbf{X})|(\mathcal{Y}(\mathbf{X}) = \mathbf{y})\right] \\ &= \mathrm{Var}\left[\mathbf{y}\right] \\ &= \mathbf{0}. \end{aligned} \tag{1.12}$$

Mais especificamente, o valor da variância é influenciado principalmente por uma distância entre $\mathbf{x}$ e $\mathbf{X}$, calculada através de um núcleo de covariância. □

Além dessas duas propriedades, Roustant et al. (2012) demonstram que a variância do modelo de Kriging não depende diretamente dos valores observados $\mathbf{y}$ (homocedasticidade[6] das observações ou homogeneidade de variância).

Como o escopo do trabalho de conclusão são processos condicionados (Bayesianos), para simplificação da notação, o processo $\mathcal{Y}(\mathbf{x})|(\mathcal{Y}(\mathbf{X}) = \mathbf{y})$ será simplesmente representado por $\mathcal{Y}(\mathbf{x})$.

Considerando um exemplo unidimensional, dado pela equação

$$y(x) = (6x - 2)^2 \operatorname{sen}(12x - 4) \tag{1.13}$$

avaliada nos pontos $\mathbf{X} = \{\{0{,}0\}, \{0{,}4\}, \{0{,}6\}, \{0{,}8\}, \{1{,}0\}\}^\top$. O modelo de Kriging construído pode ser representado como mostra a Figura 1.4. Nessa figura, a função real (desconhecida pelo metamodelo) é representada com uma linha cheia e a média do processo gaussiano com uma linha tracejada. Os intervalos de confiança de 25%, 50%, 75% e 95% são representados com tons de cinza (do mais escuro para o mais claro, respectivamente). À direita, é representado um corte em $x = 0{,}5$ onde pode ser vista a função densidade de probabilidade.

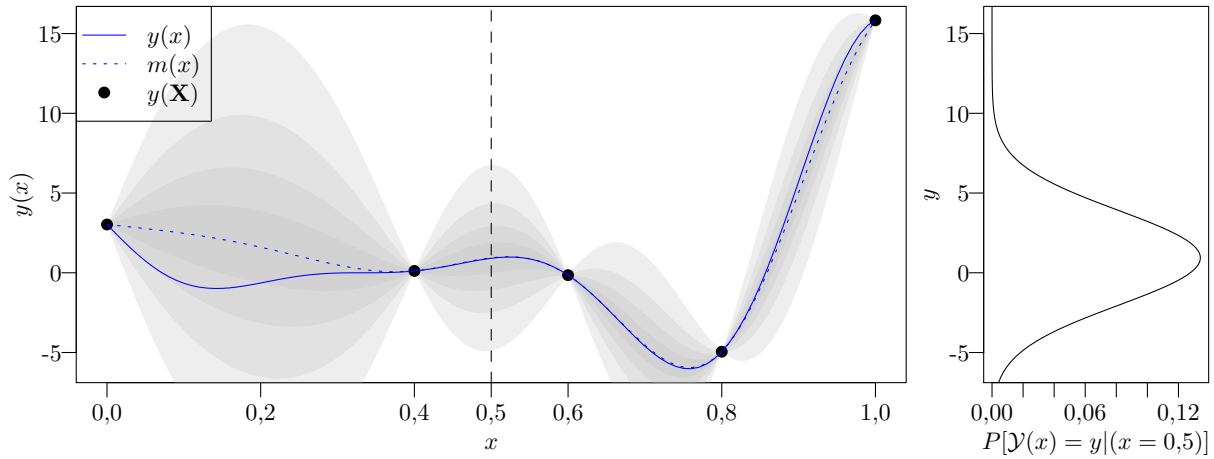



Figura 1.4 – Exemplo de metamodelo de Kriging de uma função escalar de uma variável avaliada em 5 pontos.

Dentre várias vantagens, a modelagem com processos gaussianos permite a criação de indicadores de grande utilidade como o conceito de *melhoria esperada* (EI) e a *probabilidade de melhoria* (PI).

[6] Esta é a hipótese do modelo clássico de regressão linear e pressupõe que a variância do processo é algum número constante igual a $\sigma^2$.

**Teorema 1.3.** A probabilidade de melhoria de um processo gaussiano $\mathcal{Y}(\mathbf{x})$ em relação a um valor arbitrário $y^* \in \mathbb{R}$ é dada por

$$\mathsf{PI}(y^*, \mathcal{Y}(\mathbf{x})) = \Phi\left(\frac{y^* - m(\mathbf{x})}{s(\mathbf{x})}\right), \tag{1.14}$$

onde $\Phi$ é a função de distribuição gaussiana padrão ($\mu = 0, \sigma = 1$).

*Demonstração.* Pode-se calcular a probabilidade que uma realização de $\mathcal{Y}(\mathbf{x})$ seja igual ou mais extrema do que $y^*$ utilizando diretamente a definição de probabilidade

$$\begin{aligned}\mathsf{PI}(y^*, \mathcal{Y}(\mathbf{x})) &= P[\mathcal{Y}(\mathbf{x}) \leq y^*] \\ &= \Phi_{\mathcal{Y}(\mathbf{x})}(y^*),\end{aligned} \tag{1.15}$$

onde $\Phi_{\mathcal{Y}(\mathbf{x})}(y)$ é a função densidade de probabilidade do processo estocástico $\mathcal{Y}(\mathbf{x})$.

O processo $\mathcal{Y}(\mathbf{x})$ pode ser reescrito na forma normal padronizada como

$$\mathcal{Z} = \frac{\mathcal{Y}(\mathbf{x}) - m(\mathbf{x})}{s(\mathbf{x})}, \tag{1.16}$$

e a probabilidade de melhoria reescrita como

$$\begin{aligned}\mathsf{PI}(y^*, \mathcal{Y}(\mathbf{x})) &= P[s(\mathbf{x})\mathcal{Z} + m(\mathbf{x}) \leq y^*] \\ &= P\left[\mathcal{Z} \leq \frac{y^* - m(\mathbf{x})}{s(\mathbf{x})}\right] \\ &= \Phi\left(\frac{y^* - m(\mathbf{x})}{s(\mathbf{x})}\right).\end{aligned} \tag{1.17}$$

Alternativamente, pode-se criar um novo processo estocástico $\mathcal{I}(\mathbf{x})$ que representa a melhoria obtida ao se realizar $\mathcal{Y}(\mathbf{x})$ em relação a $y^*$ dado formalmente por

$$\mathcal{I}(y^*, \mathcal{Y}(\mathbf{x})) = \begin{cases} y^* - \mathcal{Y}(\mathbf{x}), & \mathcal{Y}(\mathbf{x}) < y^*; \\ 0, & \mathcal{Y}(\mathbf{x}) \geq y^*. \end{cases} \tag{1.18}$$

Note que a função de distribuição de $\mathcal{I}(\mathbf{x})$ é diferente da função de distribuição de $\mathcal{Y}(\mathbf{x}) \sim \mathcal{N}\left(m(\mathbf{x}), s^2(\mathbf{x})\right)$ e pode ser obtida aplicando a definição:

$$\Phi_{\mathcal{I}(\mathbf{x})}(i) = P[\mathcal{I}(\mathbf{x}) \leq i]. \tag{1.19}$$

Substituindo $\mathcal{I}$, para $i \geq 0$ (ou $y(\mathbf{x}) \leq y^*$), tem-se:

$$\Phi_{\mathcal{I}(\mathbf{x})}(i) = P[y^* - \mathcal{Y}(\mathbf{x}) \leq i] \tag{1.20}$$

e isolando $\mathcal{Y}$ da Equação 1.16 tem-se

$$\begin{aligned}\Phi_{\mathcal{I}(\mathbf{x})}(i) &= P[y^* - s(\mathbf{x})\mathcal{Z} - m(\mathbf{x}) \leq i], \\ &= P\left[-\mathcal{Z} \leq \frac{i - y^* + m(\mathbf{x})}{s(\mathbf{x})}\right], \\ &= P\left[\mathcal{Z} \geq -\frac{i - y^* + m(\mathbf{x})}{s(\mathbf{x})}\right], \\ &= 1 - P\left[\mathcal{Z} \leq -\frac{i - y^* + m(\mathbf{x})}{s(\mathbf{x})}\right]. \end{aligned} \tag{1.21}$$

Como a melhoria, por definição (Eq. 1.18), é não negativa, $P[\mathcal{I}(\mathbf{x}) < 0] = 0$ e tem-se a função de distribuição de $\mathcal{I}(\mathbf{x})$ dada por:

$$\Phi_{\mathcal{I}(\mathbf{x})}(i) = \begin{cases} 0, & i < 0; \\ 1 - \Phi\left(\frac{(y^* - i) - m(\mathbf{x})}{s(\mathbf{x})}\right), & i \geq 0. \end{cases} \tag{1.22}$$

A probabilidade de se obter alguma melhoria pode ser calculada como sendo a probabilidade de $\mathcal{I}$ se realizar maior do que 0 (ou de não se realizar menor ou igual a 0). Assim, o mesmo resultado pode ser obtido

$$\begin{aligned}\mathsf{PI}(y^*, \mathcal{Y}(\mathbf{x})) &= 1 - P[\mathcal{I} \leq 0] \\ &= 1 - P[\mathcal{I} < 0] - P[\mathcal{I} = 0]) \\ &= 1 - 0 - \left(1 - \Phi\left(\frac{y^* - m(\mathbf{x})}{s(\mathbf{x})}\right)\right) \\ &= \Phi\left(\frac{y^* - m(\mathbf{x})}{s(\mathbf{x})}\right) \end{aligned} \tag{1.23}$$

□

Para exemplificar, considere novamente a função descrita na Equação 1.13 dessa vez avaliada nos pontos $\mathbf{X} = \{\{0{,}1\}, \{0{,}3\}, \{0{,}5\}, \{0{,}6\}, \{0{,}9\}, \{1{,}0\}\}^\top$, representada na Figura 1.5. Na parte direita dessa figura, é apresentado um corte em $x = 0{,}70$, onde pode ser vista a função densidade de probabilidade. Para o cálculo da probabilidade de melhoria é considerado $y^* = \min(\mathbf{y}) = -0{,}149$ (melhor valor amostrado). A área hachurada representa a probabilidade de que ao amostrar o projeto $x = 0{,}70$ obtenha-se uma resposta menor do que $y^*$.

**Teorema 1.4.** A melhoria esperada em um processo gaussiano $\mathcal{Y}(\mathbf{x})$ em relação a um valor arbitrário $y^* \in \mathbb{R}$ é dada por

$$\mathsf{EI}(y^*, \mathcal{Y}(\mathbf{x})) = (y^* - m(\mathbf{x}))\Phi\left(\frac{y^* - m(\mathbf{x})}{s(\mathbf{x})}\right) + s(\mathbf{x})\phi\left(\frac{y^* - m(\mathbf{x})}{s(\mathbf{x})}\right), \tag{1.24}$$

onde $\phi$ é a função de densidade de probabilidade gaussiana padrão ($\mu = 0, \sigma = 1$).

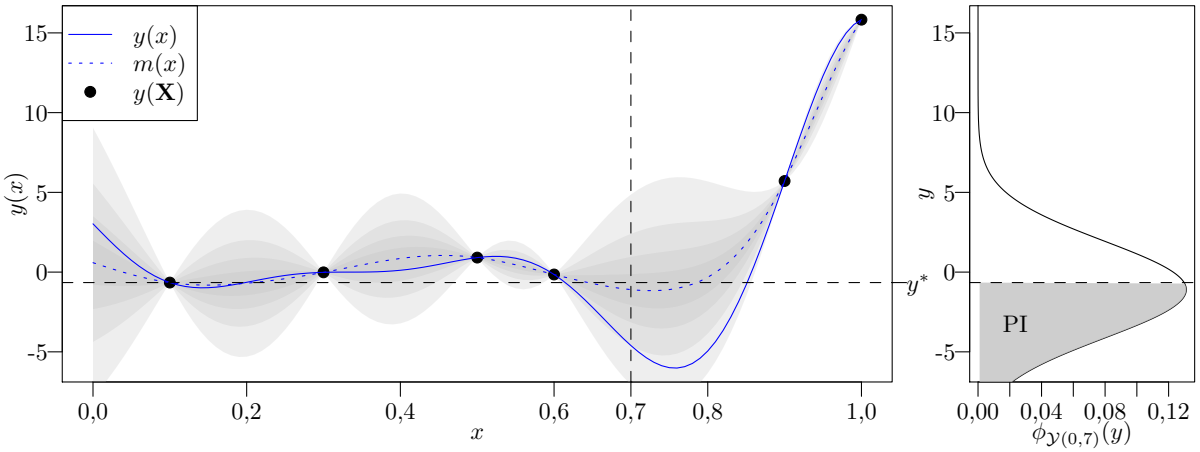



Figura 1.5 – Exemplo da probabilidade de melhoria para um metamodelo de Kriging de uma função escalar de uma variável avaliada em 6 pontos. Na direita, a área hachurada representa a probabilidade de melhoria ao amostrar o valor verdadeiro de $f(0{,}7)$.

*Demonstração.* Para o cálculo da melhoria esperada, aplica-se a esperança matemática no processo melhoria. Para isso, $\mathcal{I}(\mathbf{x})$ é reescrito de maneira mais compacta como $\mathcal{I}(\mathbf{x}) = [y^* - \mathcal{Y}(\mathbf{x})]^+$. Ou ainda, padronizando as variáveis:

$$\mathcal{I}(y^*, \mathcal{Y}(\mathbf{x})) = [y^* - \mathcal{Y}(\mathbf{x})]^+ = [s(\mathbf{x})(z^*(\mathbf{x}) - \mathcal{Z}(\mathbf{x}))]^+ \tag{1.25}$$

onde $z^*(\mathbf{x}) = \dfrac{y^* - m(\mathbf{x})}{s(\mathbf{x})}$ e $\mathcal{Z}(\mathbf{x}) = \dfrac{\mathcal{Y}(\mathbf{x}) - m(\mathbf{x})}{s(\mathbf{x})}$.

Aplicando o operador esperança, tem-se:

$$\begin{aligned} \mathrm{EI}(y^*, \mathcal{Y}(\mathbf{x})) &= E[\mathcal{I}(y^*, \mathcal{Y}(\mathbf{x}))] \\ &= E\left[(s(\mathbf{x})(z^* - Z(\mathbf{x})))^+\right]. \end{aligned} \tag{1.26}$$

Pela definição de esperança matemática e, por simplicidade, omitindo $(\mathbf{x})$ da integral:

$$\mathrm{EI}(y^*, \mathcal{Y}(\mathbf{x})) = \int_{-\infty}^{z^*} s(z^* - z)\phi(z)\ \mathrm{d}z. \tag{1.27}$$

Note que os limites da integral são definidas pela imagem de $\mathcal{I}(\mathbf{x})$. Uma vez que a função densidade de probabilidade de $\mathcal{I}(\mathbf{x})$ é não-nula para $i \in [0, \infty)$, isolando $\mathcal{Z}(\mathbf{x})$ na Equação 1.25, obtém-se

$$\mathcal{Z}^*(\mathbf{x}) = \frac{s\, z^* - \mathcal{I}(\mathbf{x})}{s} \tag{1.28}$$

cuja função densidade de probabilidade é não-nula para $z \in (-\infty, z^*]$.

Separando a integral em duas partes e resolvendo a primeira parcela tem-se que:

$$\mathrm{EI}(y^*, \mathcal{Y}(\mathbf{x})) = sz^* \int_{-\infty}^{z^*} \phi(z)\ \mathrm{d}z - s \int_{-\infty}^{z^*} z\phi(z)\ \mathrm{d}z \tag{1.29}$$

$$= sz^* \left[\Phi(z)\right]_{-\infty}^{z^*} - s \int_{-\infty}^{z^*} z\phi(z)\ \mathrm{d}z. \tag{1.30}$$

A segunda parcela pode ser resolvida explicitando a função densidade de probabilidade e integrando (usando $\int e^u \, \mathrm{d}u = e^u$):

$$\mathsf{EI}(y^*, \mathcal{Y}(\mathbf{x})) = sz^* \left[\Phi(z)\right]_{-\infty}^{z^*} - s \int_{-\infty}^{z^*} z \frac{e^{-z^2/2}}{\sqrt{2\pi}} \, \mathrm{d}z \tag{1.31}$$

$$= sz^* \left[\Phi(z)\right]_{-\infty}^{z^*} + s \left[\frac{e^{-z^2/2}}{\sqrt{2\pi}}\right]_{-\infty}^{z^*} \tag{1.32}$$

$$= sz^* \left[\Phi(z)\right]_{-\infty}^{z^*} + s \left[\phi(z)\right]_{-\infty}^{z^*} \tag{1.33}$$

Aplicando os limites de integração e retornando às variáveis originais obtém-se a equação para a melhoria esperada:

$$\mathsf{EI}(y^*, \mathcal{Y}(\mathbf{x})) = (y^* - m(\mathbf{x}))\Phi\left(\frac{y^* - m(\mathbf{x})}{s(\mathbf{x})}\right) + s(\mathbf{x})\phi\left(\frac{y^* - m(\mathbf{x})}{s(\mathbf{x})}\right). \tag{1.34}$$

□

Outra maneira de se obter a melhoria esperada é através da integral de Riemann–Stieltjes:

$$\mathrm{E}[\mathcal{X}] = \int_{-\infty}^{\infty} x\phi_{\mathcal{X}}(x) \, \mathrm{d}x = \int_{-\infty}^{\infty} x \, \mathrm{d}(\Phi_{\mathcal{X}}(x)) \tag{1.35}$$

Apesar de não auxiliar na obtenção analítica da melhoria esperada, a integral auxilia na visualização gráfica. Na Figura 1.6, o mesmo exemplo da Figura 1.5 é reproduzido. Contudo, na parte direita da figura, tem-se a função de distribuição do processo de melhoria $\Phi_{\mathcal{I}(\mathbf{x})}(i)$. A área hachurada representa exatamente a seguinte integral

$$\mathrm{E}[\mathcal{I}(y^*, \mathcal{Y}(\mathbf{x}))] = \int_{-\infty}^{\infty} i \, \mathrm{d}(\Phi_{\mathcal{I}}(i)), \tag{1.36}$$

que é a melhoria esperada.

## 1.6 EXPERIMENTOS COMPUTACIONAIS MULTIOBJETIVO

É importante ressaltar que o processo gaussiano modela a resposta (objetivo) do experimento. Portanto, apesar de o processo ter como variável o vetor de projeto $\mathbf{x}$, a distribuição de probabilidade se dá apenas em função de $y$. Em outras palavras, dado um vetor de projeto $\mathbf{x}$ qualquer fixo, o processo gaussiano $\mathcal{Y}(\mathbf{x})$ pode ser visto como uma variável aleatória $\mathcal{Y}$ simples na qual os cálculos de probabilidade se dão apenas em $y$. Sendo assim, independente do número de variáveis de projeto (dimensão de $\mathbf{x}$), para dois ou mais objetivos, o processo gaussiano da resposta é multi-variado (múltiplos $y$ para um dado $\mathbf{x}$):

$$\boldsymbol{\mathcal{Y}}(\mathbf{x}) \sim \mathcal{N}\left(\mathbf{m}(\mathbf{x}), \mathbf{S}^2(\mathbf{x})\right), \tag{1.37}$$

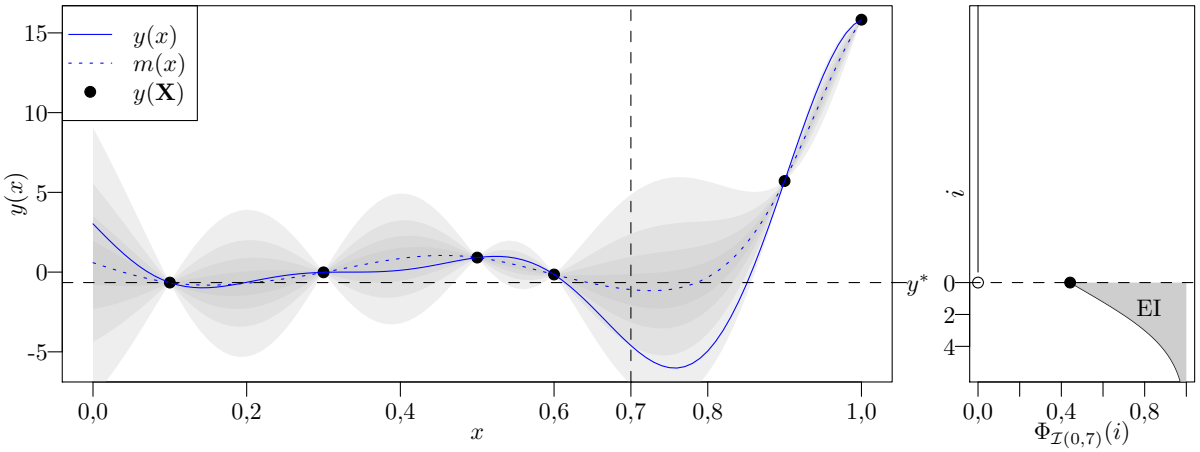



Figura 1.6 – Exemplo da probabilidade de melhoria para um metamodelo de Kriging de uma função escalar de uma variável avaliada em 6 pontos. Na direita, a área hachurada representa a melhoria esperada ao amostrar o valor verdadeiro de $f(0{,}7)$.

onde $\boldsymbol{\mathcal{Y}}$ é o processo gaussiano multivariado, $\mathbf{m}(\mathbf{x})$ é o vetor média e $\mathbf{S}^2(\mathbf{x})$ é a matriz de covariância.

Apesar de ser possível calcular completamente a matriz de covariância $\mathbf{S}^2(\mathbf{x})$, Alvarez et al. (2012), Kleijnen e Mehdad (2014) e Binois e Picheny (2016) afirmam que o aumento da complexidade não justifica o ganho obtido. De fato, a grande maioria dos autores (praticamente todos os artigos contidos na revisão bibliográfica do presente trabalho) não consideram os efeitos da correlação entre as respostas dos diferentes objetivos. Dessa maneira, a matriz de covariância se torna diagonal e o cálculo das funções de distribuição marginais se torna trivial. Dada a hipótese de independência estatística dos objetivos, cada modelo pode ser construído separadamente

$$\boldsymbol{\mathcal{Y}}(\mathbf{x}) = \left\{ \begin{array}{c} \{\mathcal{Y}_1(\mathbf{x}) \sim \mathcal{N}(m_1(\mathbf{x}), s_1^2(\mathbf{x}))\} \\ \{\mathcal{Y}_2(\mathbf{x}) \sim \mathcal{N}(m_2(\mathbf{x}), s_2^2(\mathbf{x}))\} \\ \cdots \\ \{\mathcal{Y}_k(\mathbf{x}) \sim \mathcal{N}(m_k(\mathbf{x}), s_k^2(\mathbf{x}))\} \end{array} \right\}. \tag{1.38}$$

Contudo, Boyle e Frean (2004), Chan (2013) e Shah e Ghahramani (2016) afirmam que considerar a correlação entre os objetivos trazem ganhos significativos em diversos problemas de otimização. De fato, devido à natureza conflitante dos objetivos em problemas reais, não é raro encontrar problemas de otimização onde os objetivos e/ou as restrições estão fortemente correlacionados negativamente. O ganho pode vir a ser especialmente importante no caso de funções com custo computacional extremamente elevado, onde qualquer informação que possa ser inferida dos dados já observados é bem vinda. Atualmente, a maioria absoluta dos trabalhos publicados não consideram a correlação entre os objetivos.

Assim como os processos unidimensionais, os processos multiobjetivo permitem o cálculo de métricas muito utilizadas pelos algoritmos de otimização. Na sequência, são introduzidos os conceitos de probabilidade de melhoria bidimensional, probabilidade de dominância, e hiper-volume dominado esperado.

**Teorema 1.5.** A probabilidade de que uma resposta $\mathbf{y}(\mathbf{x}) = \{y_1(\mathbf{x}), y_2(\mathbf{x})\}$, realização de um processo gaussiano bi-variado $\boldsymbol{\mathcal{Y}}(\mathbf{x})$ independente (i.e. dois objetivos não correlacionados), **domine** algum outro projeto $\mathbf{y}^* = \{y_1^*, y_2^*\}$, pertencente a um conjunto de Pareto $\mathbf{S}$ é dada por

$$\text{PD}(\mathbf{y}, \boldsymbol{\mathcal{Y}}(\mathbf{x})) = \Phi_{\mathcal{Y}_1}\left(\frac{y_1^* - m_1(\mathbf{x})}{s_1(\mathbf{x})}\right)\Phi_{\mathcal{Y}_2}\left(\frac{y_2^* - m_2(\mathbf{x})}{s_2(\mathbf{x})}\right). \tag{1.39}$$

*Demonstração.* No caso de dois objetivos, a região onde um projeto $\mathbf{y}(\mathbf{x})$ dominaria um outro projeto $\mathbf{y}^*$ é representada pela região $\Omega_1$ na Figura 1.7.

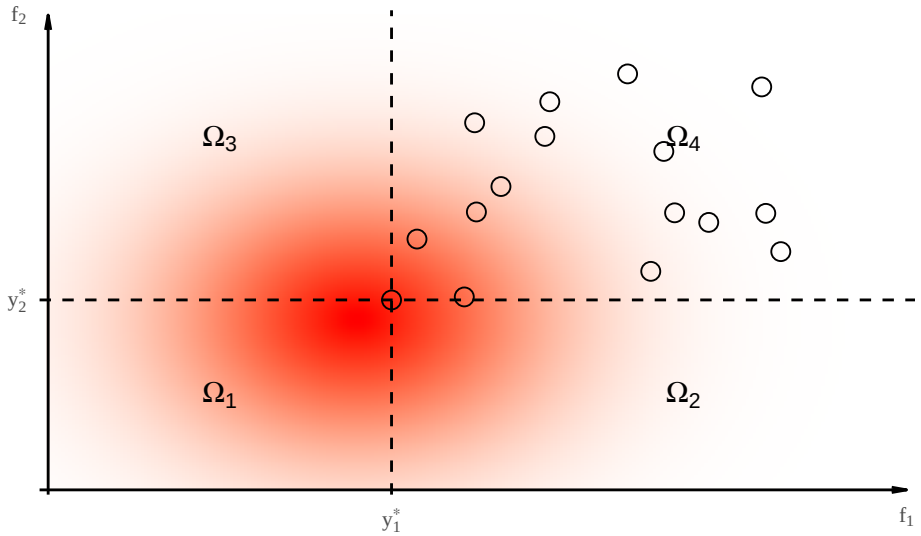



Figura 1.7 – Regiões de dominância ($\Omega_1$), melhoria ($\Omega_2$ e $\Omega_3$) e dominada ($\Omega_4$) por uma frente de Pareto de um único projeto. Em vermelho, a probabilidade da localização no espaço dos objetivos para um dado vetor de projeto.

A probabilidade de dominância pode ser interpretada como a probabilidade de que, para um determinado projeto $\mathbf{x}$ (no domínio de projeto), o processo estocástico $\boldsymbol{\mathcal{Y}}(\mathbf{x})$ gere uma realização $\mathbf{y}(\mathbf{x})$ (no domínio dos objetivos) domine uma resposta de referência $\mathbf{x}$ (se encontre na região $\Omega_1$). Para o exemplo mostrado na Figura 1.7, essa condição pode ser expressa como

$$\text{PD}(\mathbf{y}^*, \boldsymbol{\mathcal{Y}}(\mathbf{x})) = P[\boldsymbol{\mathcal{Y}}(\mathbf{x}) \in \Omega_1] = P[\mathcal{Y}_1 \leq y_1^* \cup \mathcal{Y}_2 \leq y_2^*]. \tag{1.40}$$

Considerando um metamodelo $\boldsymbol{\mathcal{Y}}(\mathbf{x}) = \{\mathcal{Y}_1(\mathbf{x}), \mathcal{Y}_2(\mathbf{x})\}$, com objetivos não correlacionados, a função densidade de probabilidade conjunta pode ser escrita como

$$\phi_{\mathcal{Y}_1,\mathcal{Y}_2} = \phi_{\mathcal{Y}_1}\phi_{\mathcal{Y}_2}, \tag{1.41}$$

onde $\phi_{\mathcal{Y}_1}$ e $\phi_{\mathcal{Y}_2}$ são de fato[7] $\phi_{\mathcal{Y}_1(\mathbf{x})}$ e $\phi_{\mathcal{Y}_2(\mathbf{x})}$ e são as funções densidades de probabilidade de cada um dos metamodelos dos objetivos independentemente. Assim, PD se resume em

$$\begin{aligned}\mathsf{PD}(\mathbf{y}^*, \boldsymbol{\mathcal{Y}}(\mathbf{x})) &= P[\mathcal{Y}_1 \leq y_1]P[\mathcal{Y}_2 \leq y_2] \\ &= \Phi_{\mathcal{Y}_1}\left(\frac{y_1^* - m_1(\mathbf{x})}{s_1(\mathbf{x})}\right)\Phi_{\mathcal{Y}_2}\left(\frac{y_2^* - m_2(\mathbf{x})}{s_2(\mathbf{x})}\right).\end{aligned} \tag{1.42}$$

□

**Teorema 1.6.** A probabilidade de que uma resposta $\mathbf{y}(\mathbf{x}) = \{y_1(\mathbf{x}), y_2(\mathbf{x})\}$, realização de um processo gaussiano bi-variado $\boldsymbol{\mathcal{Y}}(\mathbf{x})$ independente (i.e. dois objetivos não correlacionados), **melhore**[8] um conjunto de Pareto $\mathbf{S}$ definido por apenas um projeto $\mathbf{y}^* = \{y_1^*, y_2^*\}$ é dada por

$$\mathsf{PI}(\mathbf{y}^*, \boldsymbol{\mathcal{Y}}(\mathbf{x})) = \mathsf{PI}(y_1^*, \mathcal{Y}_1(\mathbf{x})) + \mathsf{PI}(y_2^*, \mathcal{Y}_2(\mathbf{x})) - \mathsf{PD}(\mathbf{y}^*, \boldsymbol{\mathcal{Y}}(\mathbf{x})). \tag{1.43}$$

*Demonstração.* No caso de dois objetivos, a região onde um projeto $\mathbf{y}(\mathbf{x})$ melhorará um conjunto de Pareto de um único projeto $\mathbf{y}^*$ é representada por $\Omega_1$, $\Omega_2$ e $\Omega_3$ na Figura 1.7.

A probabilidade de melhoria pode ser interpretada como a probabilidade de uma realização $\mathbf{y}(\mathbf{x})$ se encontrar nas regiões $\Omega_1$, $\Omega_2$ ou $\Omega_3$. Ou, de maneira mais simples, é igual à probabilidade de não se encontrar na região $\Omega_4$

$$\begin{aligned}\mathsf{PI}(\mathbf{y}^*, \boldsymbol{\mathcal{Y}}(\mathbf{x})) &= P[\boldsymbol{\mathcal{Y}}(\mathbf{x}) \in \Omega_1 \cup \Omega_2 \cup \Omega_3] \\ &= P[\boldsymbol{\mathcal{Y}}(\mathbf{x}) \notin \Omega_4] \\ &= 1 - P[\boldsymbol{\mathcal{Y}}(\mathbf{x}) \in \Omega_4] \\ &= 1 - P[\mathcal{Y}_1 > y_1^* \cup \mathcal{Y}_2 > y_2^*] \\ &= 1 - \left(1 - \Phi_{\mathcal{Y}_1}\left(\frac{y_1^* - m_1(\mathbf{x})}{s_1(\mathbf{x})}\right)\right)\left(1 - \Phi_{\mathcal{Y}_2}\left(\frac{y_2^* - m_2(\mathbf{x})}{s_2(\mathbf{x})}\right)\right) \\ &= \Phi_{\mathcal{Y}_1}\left(\frac{y_1^* - m_1(\mathbf{x})}{s_1(\mathbf{x})}\right) + \Phi_{\mathcal{Y}_2}\left(\frac{y_2^* - m_2(\mathbf{x})}{s_2(\mathbf{x})}\right) \\ &\quad - \Phi_{\mathcal{Y}_1}\left(\frac{y_1^* - m_1(\mathbf{x})}{s_1(\mathbf{x})}\right)\Phi_{\mathcal{Y}_2}\left(\frac{y_2^* - m_2(\mathbf{x})}{s_2(\mathbf{x})}\right).\end{aligned} \tag{1.44}$$

Ou ainda,

$$\mathsf{PI}(\mathbf{y}^*, \boldsymbol{\mathcal{Y}}(\mathbf{x})) = \mathsf{PI}(y_1^*, \mathcal{Y}_1(\mathbf{x})) + \mathsf{PI}(y_2^*, \mathcal{Y}_2(\mathbf{x})) - \mathsf{PD}(\mathbf{y}^*, \boldsymbol{\mathcal{Y}}(\mathbf{x})). \tag{1.45}$$

□

[7] A dependência de $\mathbf{x}$ foi suprimida para simplificar a notação.

[8] No presente texto é considerada melhoria quando um projeto domina outro(s) ou aumenta a população da frente de Pareto.

Para frentes de Pareto com mais de um elemento (como mostrado na Figura 1.2), o cálculo das probabilidades é análogo. Note que o procedimento é o mesmo, contudo o domínio de cálculo se torna mais particionado ($\Omega = \Omega_1 \cup \Omega_2 \cup \ldots \cup \Omega_n$).

**Teorema 1.7.** A esperança de melhoria do hiper-volume dominado (EHI) de um dado processo gaussiano bi-variado $\boldsymbol{\mathcal{Y}}(\mathbf{x})$ independente (i.e. dois objetivos não correlacionados) em relação a um conjunto de Pareto $\mathbf{S}$ definido por apenas um projeto $\mathbf{y}^* = \{y_1^*, y_2^*\}$ é dada por

$$\text{EHI}(\mathbf{y}^*, \boldsymbol{\mathcal{Y}}(\mathbf{x})) = \text{PI}\sqrt{(y_1^\mu - y_1^*)^2 + (y_2^\mu - y_2^*)^2} \tag{1.46}$$

onde $y_1^\mu$ e $y_2^\mu$ localizam o centroide da densidade de probabilidade de melhoria do hiper-volume.

*Demonstração.* No caso descrito, a melhoria no hiper-volume dominado por uma realização $\mathbf{y}(\mathbf{x})$ é indicado em vermelho na Figura 1.8 (a). Para o cálculo da EHI, elimina-se a região $\Omega_4$ pois não gera melhoria. Além disso, por conveniência, usualmente limita-se o domínio dos objetivos superiormente com um valor arbitrário **ref**.

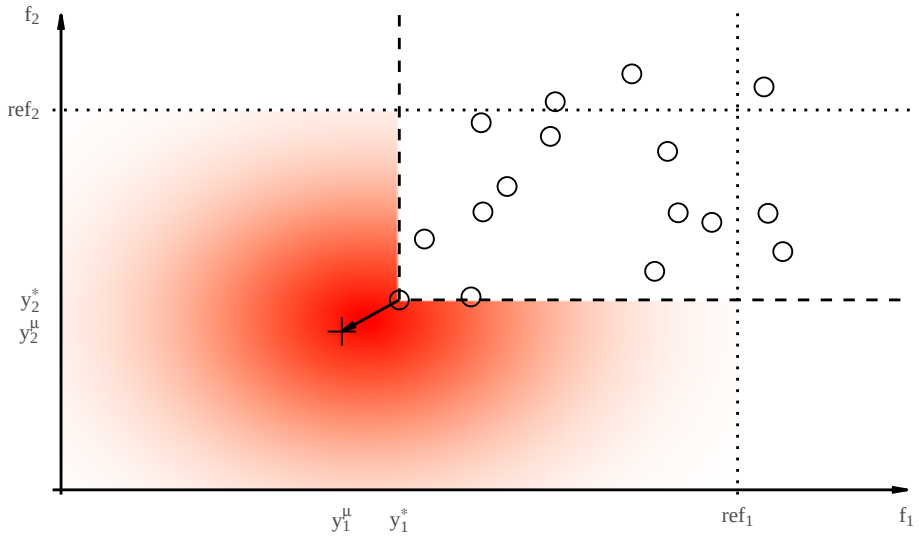



Figura 1.8 – Interpretação gráfica da esperança de melhoria do hiper-volume dominado.

Como a distribuição da realização $\mathbf{y}(\mathbf{x})$ é modelada por um processo estocástico $\boldsymbol{\mathcal{Y}}(\mathbf{x})$, uma realização da melhoria no hiper-volume também pode ser modelada

por um processo $\mathcal{I}_H$

$$\mathcal{I}_H(y^*, \mathcal{Y}(\mathbf{x})) = \begin{cases} (\mathcal{Y}_1(\mathbf{x}) - y_1^*)\,(\mathcal{Y}_2(\mathbf{x}) - y_2^*) & \\ + (\mathcal{Y}_1(\mathbf{x}) - y_1^*)\,(y_{1_{\text{ref}}} - y_1^*) & \mathcal{Y}_1(\mathbf{x}) < y_1^* \cup \mathcal{Y}_2(\mathbf{x}) < y_2^*; \\ + (\mathcal{Y}_2(\mathbf{x}) - y_2^*)\,(y_{2_{\text{ref}}} - y_2^*)\,, & \\ (\mathcal{Y}_1(\mathbf{x}) - y_1^*)\,(y_{2_{\text{ref}}} - \mathcal{Y}_2(\mathbf{x}))\,, & \mathcal{Y}_1(\mathbf{x}) < y_1^* \cup \mathcal{Y}_2(\mathbf{x}) \geq y_2^*; \\ (y_{1_{\text{ref}}} - \mathcal{Y}_1(\mathbf{x}))\,(\mathcal{Y}_2(\mathbf{x}) - y_2^*)\,, & \mathcal{Y}_1(\mathbf{x}) \geq y_1^* \cup \mathcal{Y}_2(\mathbf{x}) < y_2^*; \\ 0, & \mathcal{Y}_1(\mathbf{x}) \geq y_1^* \cup \mathcal{Y}_2(\mathbf{x}) \geq y_2^*. \end{cases} \tag{1.47}$$

Assim, a melhoria esperada do hiper-volume dominado pode ser calculada pela definição de esperança matemática

$$\text{EHI}(\mathbf{y}, \boldsymbol{\mathcal{Y}}(\mathbf{x})) = E[\mathcal{I}_H] = \int_{\Omega} i_h f_{\mathcal{I}_H}(i_h)\,\mathrm{d}i_h, \tag{1.48}$$

onde $\Omega$ é o espaço de probabilidade onde a variável $\mathcal{I}_H$ é definida e $f_{\mathcal{I}_H}(i_h)$ é a função de probabilidade da variável $\mathcal{I}_H$.

Considerando que a esperança matemática é o primeiro momento estatístico, a melhoria esperada do hiper-volume dominado pode ser calculada alternativamente como

$$\text{EHI}(\mathbf{y}, \boldsymbol{\mathcal{Y}}(\mathbf{x})) = E[\mathcal{I}_H] = P[\mathcal{I}_H > 0]\sqrt{(y_1^{\mu} - y_1^*)^2 + (y_2^{\mu} - y_2^*)^2}, \tag{1.49}$$

onde $y_1^{\mu}$ e $y_2^{\mu}$ localizam o centroide de $\mathcal{I}_H$ (indicado pelo símbolo **+** na Figura 1.8). Esse centroide pode ser computado de maneira exata pelo meio de integrais ou numericamente com o uso do método de Monte Carlo.

É importante notar que, devido a uma região de $\Omega$ possuir valor nulo de melhoria, $P[\mathcal{I}_H > 0] < 1$ e consequentemente $y_i^{\mu} < m_i(\mathbf{x})$. Para os casos de dois ou três objetivos, usualmente se computa a melhoria esperada analiticamente calculando a integral indicada pela Equação 1.48. Em Hupkens et al. (2014) podem ser encontrados mais detalhes sobre implementações exatas para o cálculo do EHI para dois e três objetivos. □

Essa breve introdução, apresenta os conceitos necessários para a compreensão do avanços e desafios descritos no capítulo de revisão bibliográfica.

# 2 REVISÃO BIBLIOGRÁFICA E LEVANTAMENTO DO ESTADO DA ARTE

## 2.1 ALGORITMOS EVOLUCIONÁRIOS DE OTIMIZAÇÃO MULTIOBJETIVO

Otimização multiobjetivo (MOO, do inglês *multiobjective optimization*) vem sendo empregada em diferentes ramos da ciência onde decisões ótimas devem ser tomadas frente a múltiplos critérios. Entre as abordagens mais comuns estão os algoritmos evolucionários de otimização multiobjetivo (MOEA, do inglês *multiobjective evolutionary algorithms*). Um dos fatos que torna tais algoritmos atrativos, quando aplicados à problemas de otimização multiobjetivo, é que em geral não necessitam do gradiente das funções, o que possibilita o estudo de funções do tipo caixa preta. Uma desvantagem dos algoritmos evolutivos, é a necessidade de um elevado número de avaliações das funções objetivo (simulações numéricas por exemplo). Além disso, normalmente, algoritmos evolutivos não possuem garantias de convergência para um ótimo, nem critérios de parada robustos.

Umas das primeiras revisões abrangentes sobre o assunto são apresentadas em Fonseca e Fleming (1995), Veldhuizen e Lamont (1998), Zitzler (1999) e Deb et al. (2002a). Esses trabalhos ilustram as ideias básicas bem como os desafios apresentados. Posteriormente, inúmeros algoritmos de otimização foram propostos, alguns podem ser encontrados em Corne et al. (2001), Zitzler et al. (2001), Deb et al. (2002b), Emmerich et al. (2005), Beume et al. (2007), Zhang e Li (2007), Yang (2013) e Chen et al. (2015a). Grande parte dos algoritmos mencionados foram testados por Reed et al. (2013), onde são avaliados os seguintes temas: (1) problemas com muitos objetivos (mais de quatro objetivos), (2) multi-modalidade, (3) não linearidade, (4) problemas discretos, (5) espaço factível reduzido, (6) objetivos estocásticos (medição com erro aleatório), (7) não separabilidade genética dos operadores (epistasia).

Um dos MOEAs mais populares é o NSGA-II (DEB et al., 2002b). Contudo, com a crescente atenção para problemas com muitos objetivos (4+), evidenciaram-se problemas nesse algoritmo devido à degradação de pressão evolutiva com o aumento do número de objetivos. Para aliviar esse problema, Deb e Jain (2014) propõe o NSGA-III, uma melhoria do algoritmo anterior com adição de pontos referenciais à população. No artigo são estudados problemas de 2 a 15 objetivos sem restrições e os resultados apresentados são promissores quando comparados ao uso de MOEAs anteriores. Em Jain e Deb (2014), um segundo volume do artigo anterior, é estendida a aplicação do NSGA-III para problemas com restrições. Além disso, é adicionado um operador adaptativo de remoção de indivíduos, que também inclui novos pontos de referência

durante a execução do algoritmo. Novamente, melhorias são reportadas na representação da frente de Pareto quando comparado a trabalhos anteriores. Jain e Deb (2014) ainda testam o algoritmo em problemas de engenharia (representações analíticas de um impacto lateral de um veículo e de um problema de recursos hídricos).

Subsequentes melhorias no algoritmo NSGA-III são mostradas em Yuan et al. (2014), Ibrahim et al. (2016) e Seada et al. (2017). As melhorias tem principalmente como objetivo uma melhor convergência (elitização) populacional no processo evolutivo. Ambos os trabalhos afirmam terem obtidos resultados mais diversos e mais precisos se aproximando mais fielmente da frente de Pareto ótima. Contudo, há limitações práticas de custo computacional na abordagem empregada em Seada et al. (2017). O algoritmo DC-NSGA-III, proposto em Seada et al. (2017), baseia-se principalmente no uso de uma recente inovação, a métrica KKTPM. A métrica KKTPM (do inglês *Karush-Kuhn-Tucker Proximity Measure*) proposta inicialmente em Tulshyan et al. (2010), Dutta et al. (2013) como critério de parada para problemas de otimização mono-objetivo com restrições se baseia nas aproximação de uma solução aos critérios de otimalidade KKT. Posteriormente, Deb e Abouhawwash (2016) aplicam o critério de proximidade em uma função de escalarização o que permite que o critério seja empregado em problemas de otimização multiobjetivo. O critério KKTPM tem como objetivo transformar o problema de otimização multiobjetivo em uma única função multimodal onde cada mínimo global representa uma solução do conjunto ótimo de Pareto. Assim, o problema de otimização multiobjetivo se transforma na simples minimização global de uma função multimodal. A grande vantagem teórica dessa técnica, é que a métrica KKTPM é não negativa e possui valor zero em todos os pontos ótimos de Pareto. Ou seja, teoricamente a minimização dessa função garante a otimalidade do conjunto encontrado. Contudo, apesar da promissora, a técnica possui uma severa limitação do ponto de vista prático. A necessidade do cálculo do gradiente de todas as funções envolvidas (custo e restrições) impossibilita seu uso em grande parte das aplicações de engenharia, em especial nas abordadas pelo presente trabalho. Uma possível solução é o uso de co-Kriging (LAURENCEAU; SAGAUT, 2008) e Kriging melhorado com gradientes (*gradient-enhanced* Kriging, GEK) (DWIGHT; HAN, 2009; BAAR et al., 2014). Até o momento (Novembro/2017) não foram encontrados trabalhos que explorem essa combinação.

## 2.2 USO DO METAMODELO DE KRIGING NA OTIMIZAÇÃO MULTIOBJETIVO

Como apontado anteriormente, uma das principais desvantagens dos algoritmos evolutivos em geral é o grande número de acessos aos modelos de alta fidelidade. Para contornar esse problema, o uso de metamodelos (modelos substitutos) vem ganhando popularidade. Atualmente, Kriging se tornou o metamodelo padrão na maior

parte das aplicações em ambos mono e multiobjetivo.

Uma revisão abrangente na otimização multiobjetivo com uso do Kriging é mostrada em Tabatabaei et al. (2015). O trabalho cobre definições básicas bem como a construção dos metamodelos. Também são comparados 20 estudos anteriores à 2013. Tabatabaei et al. (2015) ainda classificam as técnicas em dois principais grupos. O primeiro grupo, não iterativo, tem o enfoque em construir a melhor representação possível dos dados. Após a construção da frente de Pareto aproximada, algum MOEA é empregado na determinação de um conjunto ótimo de Pareto com base apenas nas predições realizadas com o metamodelo. Nota-se que com essa abordagem é crucial a qualidade da predição. Técnicas estatísticas de estimativa do erro como a validação cruzada e o erro médio quadrático são fundamentais. O segundo grupo de abordagens é chamado iterativo ou adaptativos. Em geral, o processo é iniciado com a representação grosseira global do espaço solução e então, iterativamente, novos pontos são incorporados aos metamodelos. O enfoque principal dessa abordagem é na escolha dos pontos de preenchimento (novos pontos). É importante que, dado um orçamento computacional limitado, a escolha dos pontos seja realizada da maneira mais eficiente possível de modo à não desperdiçar chamadas dos modelos de alta fidelidade. Exemplos do uso de abordagens não iterativas podem ser vistos em Chen et al. (2015b), Zhang et al. (2016) e Gao et al. (2016). Nesses trabalhos, metamodelos de Kriging são ajustados ao conjunto de respostas iniciais e então o algoritmo NSGA-II é usado para criar uma frente de Pareto usando as médias das predições como funções objetivo.

Quando uma abordagem iterativa é adotada, a maneira mais simples de se escolher os pontos de preenchimento é pelo uso direto de um algoritmo de otimização multiobjetivo substituindo os modelos de alta fidelidade pelos metamodelos. O algoritmo escolhe pontos que, no metamodelo, correspondem a ótimos de Pareto, calculam-se as respostas de alta fidelidade para esses projetos e então os novos resultados são incorporados a novos metamodelos. Esse processo se repete até que algum critério de parada, usualmente o tempo computacional seja atingido. Li (2011) e Zhang et al. (2017) utilizam essa abordagem. Contudo, como mostrado em Forrester et al. (2008) e Forrester e Keane (2009) para o caso de um único objetivo, a otimização direta usando apenas a média predita pelo metamodelo não é confiável. É comum que, devido a falta de fidelidade inicial do metamodelo, o processo de otimização fique preso em mínimos locais devido a falta de exploração. Para resolver esse problema, em otimização mono-objetivo, Jones et al. (1998) e Jones (2001) propoem o algoritmo eficiente de otimização global (EGO, do inglês *Efficient Global Optimization*). Esse algoritmo possui extrema robustez pois combina a exploração de regiões pouco conhecidas do espaço de projeto com a busca de mínimos do preditor. Para isso, o algoritmo

iterativamente maximiza a melhoria esperada (EI). Como na formulação do EI, a variância contribui positivamente para o seu valor, há uma pressão significativa para que o algoritmo busque regiões pouco conhecidas. Além disso, a melhoria tende a zero na proximidade de pontos já visitados, o que garante que o algoritmo não amostre demasiadamente a mesma região.

Para o caso da otimização multiobjetivo, pode-se combinar os objetivos em uma única função com uso de alguma forma de escalarização. Se combinados, o problema multiobjetivo torna-se mono-objetivo e o EGO pode ser empregado. Para se obter uma representação abrangente do conjunto ótimo de Pareto, deve-se repetir o processo diversas vezes variando as constantes de pesagem da função de escalarização. Em Knowles e Hughes (2005), Knowles (2006), Fu et al. (2008) e Passos et al. (2016) é empregada essa técnica, comumente chamada de ParEGO.

Uma outra maneira de abordar o problema multiobjetivo é generalizar o conceito da melhoria esperada. Ao invés de calcular a melhoria esperada de um modelo combinado, pode-se calcular a esperança de melhoria do hipervolume (EHI, do inglês *Expected Hypervolume Improvement*) utilizando metamodelos independentes para cada objetivo. O conceito pode ser visto com maiores detalhes em (EMMERICH et al., 2011; SHIMOYAMA et al., 2013; MARTÍNEZ-FRUTOS; HERRERO-PÉREZ, 2016). Contudo, de maneira simplificada, para um dado ponto não amostrado, calcula-se o hipervolume esperado que esse ponto domina em relação ao conjunto atual de Pareto. Analogamente ao EI, essa métrica equilibra a exploração de regiões desconhecidas com melhoria de pontos bem conhecidos. Apesar de esforços no aumento da eficiência no cálculo do EHI, como os mostrados em Couckuyt et al. (2014) e Hupkens et al. (2014), esse processo possui custo computacional relativamente elevado. Além disso, o custo computacional aumenta significativamente com o aumento do número de pontos no conjunto de Pareto atual. Em problemas cujo tempo de avaliação dos modelos de alta fidelidade são extremamente elevados, o processo de maximização do EHI (que em geral pode levar poucos minutos) pode ser desprezado.

Uma taxonomia para várias metodologias de Kriging aplicados a MOO é proposta em Deb et al. (2017). A taxonomia proposta divide as técnicas em 6 grupos definidos pelo números de metamodelos necessários. Considerando que em um determinado problema haja $M$ objetivos e $J$ restrições, os grupos são definidos como:

**M1** No caso de cada uma das funções (objetivos e restrições) serem modeladas independentemente, o número necessário de metamodelos é $M + J$;

**M2** As restrições podem ser combinadas em uma única função de violação, reduzindo o número de metamodelos para $M + 1$. O artigo propõe o uso de função de

violação baseada no operador colchete. O operador retorna zero caso a violação não ocorra e um valor normalizado da restrição caso a violação aconteça.

**M3** Alternativamente, os objetivos podem ser combinados em uma única função escalar enquanto as restrições são mantidas independentes. Assim, o número de metamodelos é de $J+1$. A sugestão realizada pelo artigo é o uso de uma função de escalarização como a função de escalarização da realização (ASF, do inglês *achievement scalarization function*) ou a função de Tchebyshev. Independentemente, nesse caso as funções de escalarização combinam os objetivos através de um vetor de ponderação. Cada vetor de ponderação resulta em um único ponto ótimo de Pareto, portanto esse vetor deve ser variado durante o processo de otimização;

**M4** Combinando as metodologias M2 e M3 resulta na necessidade de modelagem de apenas duas funções escalares. Nesse caso, o número de metamodelos necessários é 2;

**M5** A quinta metodologia combina todas as funções em uma única métrica escalar, resultando na necessidade de apenas 1 metamodelo. Para isso, o artigo usa a função de seleção (S-function, do inglês *selection function*);

**M6** Finalmente, essa metodologia, descrita no próprio artigo, combina todas as funções em uma única função escalar multimodal usando a função KKTPM. Apesar de também necessitar de apenas um metamodelo, a construção da função KKTPM necessita de estimativas dos gradientes dos modelos de alta fidelidade.

Deb et al. (2017) ainda comentam que inevitavelmente com a redução do número de metamodelos há um aumento na complexidade dos mesmos. No caso extremo da abordagem M6, o artigo mostra que há dificuldades técnicas na obtenção de bons resultados para alguns problemas devido à elevada complexidade e multimodalidade da função KKTPM.

As metodologias M1 e M2 ainda podem ser subdividas com base na técnica de otimização multiobjetivo. Deb et al. (2017) classificam em duas possibilidades: generativa e simultânea. Na abordagem generativa, o algoritmo trabalha para encontrar um projeto ótimo de Pareto por vez, dessa maneira o conjunto de Pareto é “gerado” ponto à ponto. Na abordagem simultânea, múltiplas soluções candidatas são encontradas simultaneamente e a frente avança como um todo, mesmo que um projeto seja amostrado por vez.

Como aplicação da metodologia M1, pode-se citar o trabalho de Hamdaoui et al. (2015). Nesse, restrições não são abordadas e cada um dos objetivos é modelado independentemente. A partir dos valores médios preditos por cada metamodelo,

uma frente de Pareto é construída com uso do algoritmo NSGA-II. Desse conjunto de pontos, são selecionados dois grupos. O primeiro grupo contem projetos que maximizam a distância de aglomeração (do inglês *crowding distance*), parâmetro principal de resposta do algoritmo NSGA-II. Para o segundo grupo, são selecionados projetos que possuam a maior variância para cada um dos objetivos. Desse conjunto de pontos são selecionados aqueles que possuam a maior distância mínima entre os pontos já amostrados (maximiza a distância entre os projetos). Depois de selecionados, as correspondentes respostas dos modelos de alta fidelidade são calculadas, os novos dados incorporados ao arquivo e os metamodelos reconstruídos.

Outras abordagens se baseiam em algum tipo de generalização do EGO para o quadro multiobjetivo. Além do clássico EHI (EMMERICH et al., 2011), pode ser mencionado a melhoria esperada *maximin esperada* (EMI, do inglês *Expected Maximin Improvement*) que é similar ao indicador épsilon (SVENSON; SANTNER, 2016).

Baseado no algoritmo SMS-EMOA (EMMERICH et al., 2005; BEUME et al., 2007), Emmerich et al. (2006), Ponweiser et al. (2008) e Wagner et al. (2010) propõem a redução do custo computacional pelo uso do Kriging (algoritmo SMS-EGO). A principal diferença em relação ao algoritmo puramente evolutivo é que as respostas de alta fidelidade são substituídas pelas predições dos metamodelos. Nas três abordagens com Kriging, a métrica S (usada no SMS-EMOA) é estimada utilizando os intervalos de confiança inferiores para uma nível de confiança que decai com o aumento do número de pontos amostrados. A abordagem possui similaridades ao uso do EHI, possuindo limitações no tempo de execução devido a necessidade do cálculo de integrais numéricas. O tempo de execução se torna especialmente crítico com o aumento do número de pontos na frente de Pareto atual. Em uma abordagem similar, Feliot et al. (2017) otimiza problemas com restrições. Nesse trabalho, é utilizada uma eficiente decomposição das regiões factíveis e não factíveis para acelerar significativamente o cálculo do EHI.

Outro algoritmo onde cada função é modelada por um metamodelo independente é proposta em Picheny (2015). O algoritmo se baseia “na redução de incerteza gradual” (no inglês, *stepwise uncertainty reduction* – SUR) proposta por Villemonteix et al. (2009) como substituto do EI na otimização global eficiente mono-objetivo. A ideia apresentada tenta iterativamente reduzir o volume não dominado. A escolha dos pontos de preenchimento é realizada pela estimativa do ganho informacional sobre os minimizadores globais (no caso de problemas multiobjetivo, do conjunto de Pareto). Uma grande dificuldade que esse método enfrenta, é que o cálculo das probabilidades (do ganho informacional) se baseia em simulações condicionais dos metamodelos de cada um dos objetivos. Para garantir que soluções potencialmente boas não sejam eliminadas, uma amostra relativamente grande deve ser simulada. O problema se agrava

severamente quando o número de variáveis (e por consequência o hiper-volume do espaço de decisão) aumenta.

Em relação ao gerenciamento de restrições em problemas de otimização multiobjetivo, Martínez-Frutos e Herrero-Pérez (2016) afirmam que o assunto ainda é um desafio em aberto. Em problemas com restrições severas, a pequena quantidade de projetos factíveis acarreta em uma má aproximação da frente de Pareto. Essa má aproximação atrasa significativamente a convergência de algoritmos baseados em métricas de hiper-volume. Nesse mesmo artigo é afirmado que a maior parte dos estudos envolvendo Kriging em MOO não consideram problemas com restrições. De fato, a maioria absoluta dos trabalhos contidos na revisão bibliográfica do presente trabalho não contam com restrições. Para solucionar os problemas mencionados, Martínez-Frutos e Herrero-Pérez (2016) empregam uma técnica em dois passos. Inicialmente, são amostrados pontos com grande probabilidade de factibilidade com o objetivo de aumentar a número de pontos e consequentemente a precisão da frente de Pareto predita. Nessa etapa é utilizado como critério de preenchimento uma função que computa o produto entre a probabilidade de factibilidade com a incerteza epistêmica do projeto. Os projetos que maximizam esse critério são amostrados iterativamente em um processo similar ao EGO até que um número mínimo de pontos factíveis seja atingido. No segundo estágio, o enfoque muda para o enriquecimento da frente de Pareto. Nessa etapa, o critério de preenchimento passa a ser o produto entre a melhoria esperada do hiper-volume e a probabilidade de factibilidade.

Um exemplo clássico da metodologia M3 é apresentado em Knowles (2006). Ao invés de cada objetivo ser modelado com um metamodelo independente, a resposta é agregada em uma função escalar usando a função de escalarização de Tchebycheff. Para obter uma frente de Pareto representativa, a cada iteração o vetor dos pesos é aleatoriamente amostrado a partir de um repositório de vetores homogeneamente distribuídos. Em Hussein e Deb (2016), uma estratégia mais sistemática é apresentada, onde o vetor se “move” da direção principal de um objetivo para o outro gradativamente. Após a escalarização dos objetivos, o algoritmo EGO é aplicado. Wagner et al. (2010) e Svenson (2011) afirmam que a técnica é pouco efetiva quando comparada a outras alternativas. De fato, em Passos e Luersen (2015) verifica-se a baixa eficiência do método quando comparado a uma nova metodologia proposta.

Um exemplo interessante do uso da metodologia M5 pode ser visto em Hussein e Deb (2016). Nesse trabalho, funções de teste com restrições e até 3 objetivos são otimizadas. A escalarização é executada por meio da função de seleção (S-function). O que é notável na abordagem proposta é a compartimentalização do espaço solução em pequenos metamodelos locais, próximos às direções de referência (constantes de ponderação). O uso de metamodelos locais (sem usar todos os pontos

amostrados) é vantajoso quando o custo computacional dos modelos de alta fidelidade é mediano e pode-se realizar um número moderado de avaliações. Por exemplo, no trabalho foram realizadas 276 avaliações para a função de teste ZDT3 e 1610 para a DTLZ4. O problema de se ter metamodelos com um número elevado de pontos está principalmente na realização da predição de pontos não visitados. Segundo Roustant et al. (2012), a complexidade do cálculo da média e da variância de um ponto qualquer em um metamodelo com $n$ pontos é de ordem $n^2$ e $2n$, respectivamente, e invaria com a dimensão do espaço decisão. Outra vantajem potencial da compartimentalização é a facilidade de paralelização do processo. Sendo possível calcular independentemente e simultaneamente pontos de enchimento para diversas direções de referência.

No sentido oposto do uso de técnicas de escalarização, Durantin et al. (2016) usam uma abordagem multiobjetivo para melhorar a precisão em problemas de otimização global mono-objetivo com restrições. Usualmente, o problema de otimização com restrições é abordado por uma modificação do EGO onde se maximiza iterativamente o produto da melhoria esperada (EI) e a probabilidade de factibilidade (PI). Contudo, Durantin et al. (2016) demonstram que é vantajoso maximizar simultaneamente a EI e a PI. Segundo os autores, essa técnica garante um equilíbrio entre a otimalidade e a factibilidade.

O capítulo seguinte trataa da metodologia empregada no desenvolvimento do pacote computacional.

# 3 METODOLOGIA

Muito mais do que apenas um ambiente de computação estatística, R vem a cada dia sendo mais utilizado como uma linguagem de programação científica. O ambiente consiste em uma linguagem e um ambiente de interpretação gráfico, um depurador, fornece acesso a determinadas funções do sistema, e a capacidade de executar programas armazenados em arquivos de *script* (HORNIK et al., 2002). Sua página oficial (`https://www.r-project.org/`) contem links para *download* das versões mais atualizadas, que podem ser obtidas e usadas sem nenhum custo.

O núcleo do R é uma linguagem de programação interpretada que permite o uso de laços estruturados, bem como programação modular pelo uso de funções. A maioria das funções visíveis ao usuário em R são escritas em R. Para maior eficiência computacional, é possível o usuário utilizar as funções escritas em R como apenas uma interface amigável para rotinas eficientes escritas em C, C++, Fortran, entre outras. Além disso, R conta com diferentes estruturas voltadas a objetos e métodos e classes.

A distribuição R contém funcionalidade para um grande número de procedimentos estatísticos. Entre elas estão: testes não paramétricos de modelos lineares generalizados, modelos de regressão linear e não linear, análise de séries temporais, paramétricas clássica e, clustering e suavização. Existe também um grande conjunto de funções que proporcionam um ambiente gráfico flexível para criar vários tipos de apresentações de dados. Módulos adicionais ("add-on pacotes") estão disponíveis para uma variedade de propósitos específicos (veja R Add-On Pacotes). Pacotes são unidades fundamentais reprodutíveis de código R . Eles incluem funções reutilizáveis, documentação que descreve seu uso, e possivelmente um banco de dados. A grande vantagem da elaboração de rotinas via pacotes é a facilidade de instalação e uso. Um pacote público, disponível no banco de dados do CRAN (*the Comprehensive R Archive Network*) podem simplesmente ser instalado, carregado e investigado usando:

- Instalar o pacote `"x"` da CRAN com o comando `install.packages("x");`
- Carregar o pacote `"x"` recém instalado com o comando `library("x");`
- Obter ajuda sobre o pacote `"x"` com o comando `package?x` ou `help(package = "x").`

A metodologia adotada segue muito de perto a proposta por Wickham (2015). A filosofia central presente nessa metodologia é de que tudo que *pode* ser automatizado *deve* ser automatizado. O foco do desenvolvedor deve estar no “o que” e no “como” seu pacote deve fazer, ao invés da preocupação dos detalhes estruturais.

Dessa maneira, a metodologia emprega o uso de inúmeras ferramentas de implementação automatizadas descritas minunciosamente em Wickham (2015). Esse objetivo é primariamente atingido pelo uso do pacote `devtools`, apresentado em Wickham e Chang (2016). O objetivo do `devtools` é tornar o processo de desenvolvimento de pacotes o mais simples e rápido possível. Ele atinge esse objetivo encapsulando uma série de boas práticas em comandos robustos e simples. Juntamente com o ambiente de desenvolvimento integrado (IDE, do inglês *Integrated Development Environment*) RStudio (disponível em `www.rstudio.com`), `devtools` isola o usuário dos detalhes de baixo nível da construção de pacotes, liberando tempo para o desenvolvimento da aplicação em si. Os conceitos principais da metodologia adotada são:

**Código R :** Descreve a melhor maneira de organizar os arquivos, funções e diretórios.

**Metadados:** Mostra a construção do arquivo `DESCRIPTION` que descreve o funcionamento do pacote e suas dependências.

**Documentação:** Para que outras pessoas (incluindo um você futuro) possam utilizar e entender como as funções desenvolvidas operam, é fundamental que tudo esteja bem documentado. A abordagem é muito simplificada pelo uso do pacote `roxygen2` (WICKHAM et al., 2015) que permite que a documentação e o código sejam escritos simultaneamente sem perder a formatação padrão da documentação usanda no R .

**Vinhetas (*Vignettes)*:** A documentação padrão descreve cada detalhe de cada função do pacote, enquanto Vinhetas mostram o panorama geral. Vinhetas vem em forma similar à um artigo científico e mostram como, combinando diversas partes do pacote, é possível resolver um problema real. O processo de escrita das vinhetas é simplificado pelo uso da linguagem `Rmarkdown` e da ferramenta `knitr`.

**Testes:** Uma garantia de que o pacote funciona como desejado (e mais importante, continua a funcionar após alterações a atualizações). É uma parte essencial do desenvolvimento e facilita a checagem e correção de erros. O pacote `testthat` automatiza o processo de verificação e teste o que permite que estes sejam executados mais frequentemente, reduzindo a propensão de acúmulo de erros.

**Espaço de nomes (*Namespace*):** Um arquivo que define quais funções são disponibilizadas para outros pacotes ou usuários e quais funções são necessárias de outros pacotes. Essa formalidade é essencial e garante a robustez do pacote, prevenindo que alterações pré ou pós propagadas causem a perda de funcionalidade do pacote.

**Git e Github:** Plataformas de desenvolvimento e controle de versões que facilitam a colaboração e organização do processo.

**Checagem automática:** O R , nativamente, fornece uma poderosa ferramenta de controle de qualidade pelo comando `R CMD check`. A execução rotineira desse comando previne o acúmulo de erros comuns como nomes de arquivos, com-

patibilidades, consistência de objetos e métodos, dependências, documentação, testes, entre outros. `R CMD check` executa atualmente mais de 50 testes individuais que podem receber, em ordem crescente de gravidade, as seguintes respostas: `OK`; `NOTE`; `WARNING`; `ERROR`.

**Publicação:** Finalmente, após desenvolvimento e testes, o pacote é liberado para o uso geral. Há duas maneiras principais de realizar a publicação do pacote: CRAN e github. O github é apenas um repositório geral e qualquer pacote, mesmo em desenvolvimento, pode ser disponibilizado dessa maneira. A instalação de um pacote pelo github pode ser realizada pelo comando: `devtools::install_github("USER_NAME/PACKAGE_NAME")`. Contudo, a grande maioria dos usuários do R não instalam pacotes pelo github, pois CRAN fornece rastreabilidade, confiabilidade, facilidade de instalação e uma especie de "selo de autenticidade". O processo de publicação no CRAN assemelha-se muito com o de submissão de um artigo científico e demanda uma quantidade de trabalho significativamente mais elevada.

Um detalhamento esquemático do processo de elaboração de pacotes para R seguindo essa metodologia pode ser visto no Anexo A.

# 4 RESULTADOS E CONSIDERAÇÕES FINAIS

## 4.1 RESULTADOS

Três diferentes técnicas de otimização multiobjetivo baseadas em Kriging foram implementadas, sendo uma delas um algoritmo novo, desenvolvido pelo autor. As implementações foram disponibilizadas por meio de um pacote computacional em R chamado `moko` (Multiobjective Kriging Optimization). O pacote (PASSOS, 2016) está disponível gratuitamente sob licença *open source* no portal CRAN (Comprehensive R Archive Network), repositório padrão para pacotes oficiais do R .

Os algoritmos implementados (e muitos outros encontrados na literatura) se baseiam na atualização iterativa dos metamodelos e da frente de Pareto, como mostra a Figura 4.1.

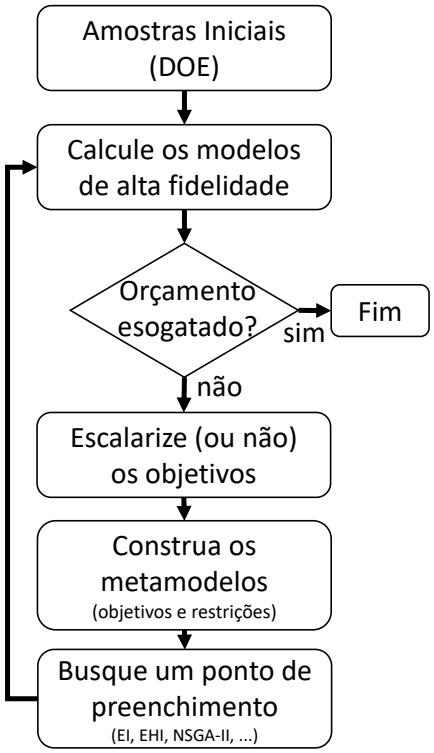



Figura 4.1 – Procedimento de otimização multiobjetivo baseado em metamodelo.

Os algoritmos implementados são descritos a seguir. Em todos os casos, a amostra inicial é um hipercubo Latino otimizado criado através do pacote `lhs` (CARNELL, 2012). Os nomes dos algoritmos foram mantidos em inglês para serem facilmente identificados com as siglas correspondentes e com os artigos já publicados.

### MULTIOBJECTIVE EFFICIENT GLOBAL OPTIMIZATION (MEGO)

MEGO, usualmente também chamado de parEGO, generaliza a ideia do EGO apresentado no capítulo de revisão bibliográfica. EGO, pode ser facilmente adaptado para problemas multiobjetivo pelo uso de funções de escaralização[1] (KNOWLES, 2006). Na implementação presente no pacote `moko`, o algoritmo MEGO utiliza a função de escalarização de Tchebycheff e pode ser sumarizado como:

**Algoritmo 1:** MEGO

Gere um DOE inicial $\mathbf{X}$ utilizando um hipercubo Latino otimizado;
Avalie $\mathbf{X}$ usando modelos de alta fidelidade e armazene as respostas de $\mathbf{f} = [f_1, f_2, \ldots, f_m]^T$ e $\mathbf{g} = [g_1, g_2, \ldots, g_p]^T$ para os $m$-objetivos e as $p$-restrições;
**while** *Orçamento computacional não esgotado* **do**
    Normalize as respostas para um espaço de tamanho $[0, 1]^m$;
    Para cada $\mathbf{x} \in \mathbf{X}$ compute uma quantidade escalar igual a $f_\lambda = \max_{i=1}^{m}(\lambda_i f_i) + \rho \sum_{(i=1)}^{m} \lambda_i f_i$, onde $\lambda$ é amostrado aleatoriamente de um conjunto de vetores unitários igualmente espaçados e $\rho$ é um valor arbitrariamente pequeno (adotado $0.05$ como padrão para o pacote `moko`);
    Construa modelos de Kriging $f_\lambda$ e para cada restrição $\mathbf{g}$;
    Encontre um $\mathbf{x}^\star$ que maximize a melhoria esperada restringida: $\mathbf{x}^\star = \arg(\max(\mathrm{EI}_C(\mathbf{x})))$;
    Avalie os valores verdadeiros de $\mathbf{f}(\mathbf{x}^\star)$ e $\mathbf{g}(\mathbf{x}^\star)$ usando os modelos de alta fidelidade;
    Atualize o banco de dados;
**end**

### EXPECTED HYPERVOLUME IMPROVEMENT (EHI – HEGO)

A melhoria esperada do hipervolume, descrita na introdução, é uma medida popular quando utilizado o metamodelo de Kriging. O cálculo do EHI é realizado pelo

[1] Funções transformam múltiplos objetivos em um único escalar.

uso do pacote computacional `GPareto` (BINOIS; PICHENY, 2016).

**Algoritmo 2:** HEGO

1 Gere um DOE inicial $\mathbf{X}$ utilizando um hipercubo Latino otimizado;
2 Avalie $\mathbf{X}$ usando modelos de alta fidelidade e armazene as respostas de $\mathbf{f} = [f_1, f_2, \ldots, f_m]^T$ e $\mathbf{g} = [g_1, g_2, \ldots, g_p]^T$ para os $m$-objetivos e as $p$-restrições;
3 **while** *Orçamento computacional não esgotado* **do**
4  Normalize as respostas para um espaço de tamanho $[0, 1]^m$;
5  Para cada um dos $m$-objetivos e $p$-restrições, construa um modelo de Kriging;
6  Encontre um $\mathbf{x}^\star$ que maximize a melhoria esperada de hypervolume restrita $\mathbf{x}^\star = \arg(\max(\text{EHI}_C(\mathbf{x})))$;
7  Avalie os valores verdadeiros de $\mathbf{f}(\mathbf{x}^\star)$ e $\mathbf{g}(\mathbf{x}^\star)$ usando os modelos de alta fidelidade;
8  Atualize o banco de dados;
9 **end**

Para esse algoritmo bem como para o anterior, o método do recozimento simulado (*simulated annealing*), disponível no pacote `GenSA` (Yang Xiang et al., 2013), foi usado para a maximização do critério de preenchimento.

### MINIMIZATION OF THE VARIANCE OF THE KRIGING-PREDICTED FRONT (MVPF)

O algoritmo MVPF, proposto em (PASSOS; LUERSEN, 2017b), se baseia na melhoria iterativa da fidelidade da frente de Pareto predita. A partir de um conjunto inicial de dados (DOE) um metamodelo é construído para cada resposta (objetivos e restrições). Utilizando a média dos metamodelos, um algoritmo de otimização multiobjetivo constrói uma Frente de Pareto estimada $\mathbf{P}$ para o problema. Desse conjunto $\mathbf{P}$, seleciona-se o projeto $\mathbf{x}^\star$ com maior variância (projeto mais isolado no espaço de decisão) para ter seu valor verdadeiro avaliado com os modelos de alta fidelidade. Os metamodelos são reconstruídos, agora com mais informações, e o processo se repete até um critério de parada ser atingido. O algoritmo pode ser formalmente descrito

como:

**Algoritmo 3:** MVPF

1 Gere um DOE inicial $\mathbf{X}$ utilizando um hipercubo Latino otimizado;
2 Avalie $\mathbf{X}$ usando modelos de alta fidelidade e armazene as respostas de $\mathbf{f} = [f_1, f_2, \ldots, f_m]^T$ e $\mathbf{g} = [g_1, g_2, \ldots, g_p]^T$ para os $m$-objetivos e as $p$-restrições;
3 **while** *Orçamento computacional não esgotado* **do**
4 Para cada um dos $m$-objetivos e $p$-restrições, construa um modelo de Kriging;
5 Estime um conjunto de Pareto $\mathbf{P}$ utilizando as médias dos modelos de Kriging utilizando um algoritmo de otimização multiobjetivo (NSGA-II no caso do pacote `moko`);
6 Encontre $\mathbf{x}^\star \in \mathbf{P}$ que maximiza a variância do preditor de Kriging $\mathbf{x}^\star = \arg(\max(\mathrm{s_{km}}(\mathbf{x})))$;
7 Avalie os valores verdadeiros de $\mathbf{f}(\mathbf{x}^\star)$ e $\mathbf{g}(\mathbf{x}^\star)$ usando os modelos de alta fidelidade;
8 Atualize o banco de dados;
9 **end**

A implementação NSGA-II utilizada está disponível no pacote `mco` (MERSMANN, 2014).

Mais detalhes da implementação e uso do pacote desenvolvido podem ser vistos no artigo (PASSOS; LUERSEN, 2017a) cuja cópia está disponível no Apêndice D. Atualmente o pacote computacional `moko` recebe aproximadamente 120 downloads mensais[2]

Os apêndices C, D e E trazem exemplos de aplicações dos algoritmos descritos em problemas de engenharia. Dentre os problemas abordados, se encontram otimização das orientações locais de fibras em estruturas de material compósito. Problemas com restrições também são abordados.

## 4.2 CONSIDERAÇÕES FINAIS E SUGESTÕES PARA TRABALHOS FUTUROS

No presente projeto de conclusão são apresentados os conceitos básicos referentes a otimização multiobjetivo com uso do metamodelo de Kriging. O entendimento desses conceitos é fundamental para a compreensão dos algoritmos propostos.

Em uma revisão bibliográfica abrangente, pôde-se notar a relevância do tema em estudo. Além disso, oportunidades de pesquisa foram observadas e suas viabilidades serão investigadas.

O trabalho apresenta um novo algoritmo de otimização multiobjetivo baseada

[2] podendo ser verificado em https://www.rdocumentation.org/packages/moko/versions/1.0.1

no modelo de Kriging (MVPF) cuja eficácia foi testada em problemas reais de engenharia.

Como pode ser visto nos apêndices, o projeto encontra-se avançado com uma boa quantidade de algoritmos implementados e testados. Considera-se também que o projeto apresenta maturidade e grau de inovação adequados, tendo em vista que já houve uma publicação em periódico dos trabalhos desenvolvidos (PASSOS; LUERSEN, 2017b).

Em comparação ao pacote `GPareto`, `moko` apresenta vantagens por possibilitar a otimização de problemas com restrições. Contudo, uma comparação direta em problemas sem restrições não foi realizada. Uma possível comparação não seria significativa para o presente trabalho, pois os algoritmos implementados nos dois pacotes diferem substancialmente. Assim, a validade da implementação não seria avaliada, e sim a qualidade do algoritmo.

Com base no trabalho desenvolvido, foi possível detectar as seguintes oportunidades de pesquisas futuras:

**Integração com o Pacote `GPareto`:** Para facilmente se obter maior variadade de algoritmos implementados, uma integração maior com o pacote GPareto poderia ser realizada. Essa integração, facilmente disponibilizaria para o `moko` os seguintes algoritmos baseados no seguintes indicadores: *Expected Maximin Improvement*, *S-metric selection* e *Expected Excursion Volume Reduction*.

**Indicador Épsilon Esperado:** Aparentemente nenhum trabalho encontrado utiliza o indicador épsilon esperado. Essa métrica, seria obtida ao construir um processo estocástico para o indicador épsilon e subsequentemente calcular seu valor esperado. O vetor de projeto que resultasse no maior épsilon esperado seria selecionado como ponto de preenchimento.

**Função de Distribuição Conjunta:** Como mostrado na introdução, o uso de um único processo Gaussiano multivariado para a representação de funções vetoriais (multiobjetivo) ainda é muito pouco explorado. A elaboração de algoritmos de otimização que se beneficiem dessa abordagem pode vir a ser uma pesquisa promissora.

**Entropia da Frente de Pareto:** A minimização iterativa da variância da frente de Pareto predita foi empregada em Passos e Luersen (2015). Inicialmente, uma frente de Pareto é construída utilizando o algoritmo NSGA-II com a média do metamodelo de Kriging. Dentre os pontos selecionados pelo NSGA-II, os modelos de alta fidelidade são calculados sob o projeto com maior variância. Um critério de pre-

enchimento potencialmente mais robusto é a entropia (estatística) do processo Gaussiano calculada em cada um dos pontos selecionados.

**Abordagem Informacional:** De maneira análoga à executada por Picheny (2015), uma abordagem informacional pode ser empregada para a construção da frente de Pareto. A mudança de paradigma proposta é de pensar no fato de que a localização mais provável (dado o conhecimento prévio das observações) para o mínimo global de uma função não necessariamente fornece o maior ganho de informação sobre a localização do minimizador.

**Probabilidade de Dominância:** Uma abordagem mais simplificada em relação ao EHI, é calcular a probabilidade de que, quando amostrado, um determinado projeto não seja dominado por nenhum outro resultado da frente de Pareto atual. Essa técnica, não observada na literatura investigada, pode servir como complemento para algum outro critério de preenchimento. O uso apenas desse critério provavelmente não é eficiente devido à falta de pressão para a busca em regiões desconhecidas, similar ao que ocorre com a probabilidade de melhoria na otimização mono-objetivo.

**KKTPM + Kriging:** Como mostrado anteriormente, a métrica de proximidade KKTPM é robusta na detecção da otimalidade da frente de Pareto. Contudo, a necessidade do uso do gradiente é impeditiva em grande parte das aplicações reais. Uma combinação do metamodelo de Kriging para a estimativa de tais gradientes pode viabilizar o processo.

**Problemas com Restrições Severas:** Conforme reportado em Martínez-Frutos e Herrero-Pérez (2016) e também observado na presente revisão bibliográfica, problemas com restrições (em especiais em grande número) combinados à otimização multi-objetivo e Kriging ainda são pouco explorados. Diferentes trabalhos identificam dificuldades na representação correta das regiões factíveis quando metamodelos são empregados, o que acaba reduzindo significativamente a eficiência dos algoritmos.

**Otimização com Muitos Objetivos:** Uma área que vem ganhando espaço nos últimos anos é a otimização de problemas com muitos objetivos (4 ou mais). Contudo, esse avanço vem sendo realizado principalmente em algoritmos de otimização heurísticos. O comportamento da modelagem por Kriging para tais problemas ainda é praticamente inexplorado.

# REFERÊNCIAS


ALVAREZ, M. A.; ROSASCO, L.; LAWRENCE, N. D. et al. Kernels for vector-valued functions: A review. **Foundations and Trends in Machine Learning**, v. 4, n. 3, p. 195–266, 2012.

BAAR, J. H. de; DWIGHT, R. P.; BIJL, H. Improvements to gradient-enhanced kriging using a bayesian interpretation. **International Journal for Uncertainty Quantification**, v. 4, n. 3, 2014.

BEUME, N.; NAUJOKS, B.; EMMERICH, M. Sms-emoa: Multiobjective selection based on dominated hypervolume. **European Journal of Operational Research**, v. 181, n. 3, p. 1653–1669, 2007.

BINOIS, M.; PICHENY, V. **GPareto: Gaussian Processes for Pareto Front Estimation and Optimization**. [S.l.], 2016. R package version 1.0.2. Disponível em: <http://CRAN.R-project.org/package=GPareto>.

BLASCO, X.; HERRERO, J.; SANCHIS, J.; MARTÍNEZ, M. A new graphical visualization of n-dimensional pareto front for decision-making in multiobjective optimization. **Information Sciences**, v. 178, n. 20, p. 3908–3924, 2008.

BOYLE, P.; FREAN, M. R. Dependent gaussian processes. In: **NIPS**. [S.l.: s.n.], 2004. v. 17, p. 217–224.

CARNELL, R. **LHS: Latin Hypercube Samples**. [S.l.], 2012. R package version 0.10. Disponível em: <http://CRAN.R-project.org/package=lhs>.

CHAN, A. B. Multivariate generalized gaussian process models. **arXiv preprint arXiv:1311.0360**, 2013.

CHEN, B.; ZENG, W.; LIN, Y.; ZHANG, D. A new local search-based multiobjective optimization algorithm. **Evolutionary Computation, IEEE Transactions on**, v. 19, n. 1, p. 50–73, 2015.

CHEN, S.; SHI, T.; WANG, D.; CHEN, J. Multi-objective optimization of the vehicle ride comfort based on kriging approximate model and nsga-ii. **Journal of Mechanical Science and Technology**, v. 29, n. 3, p. 1007–1018, 2015.

CORNE, D. W.; JERRAM, N. R.; KNOWLES, J. D.; OATES, M. J. et al. PESA-II: Region-based selection in evolutionary multiobjective optimization. In: **Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)**. [S.l.: s.n.], 2001.

COUCKUYT, I.; DESCHRIJVER, D.; DHAENE, T. Fast calculation of multiobjective probability of improvement and expected improvement criteria for pareto optimization. **Journal of Global Optimization**, v. 60, n. 3, p. 575–594, 2014.

DEB, K.; ABOUHAWWASH, M. An optimality theory-based proximity measure for set-based multiobjective optimization. **IEEE Transactions on Evolutionary Computation**, v. 20, n. 4, p. 515–528, 2016.

DEB, K.; HUSSEIN, R.; ROY, P.; TOSCANO, G. Classifying metamodeling methods for evolutionary multi-objective optimization: First results. In: SPRINGER. **International Conference on Evolutionary Multi-Criterion Optimization**. [S.l.], 2017. p. 160–175.

DEB, K.; JAIN, H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. **IEEE Trans. Evolutionary Computation**, v. 18, n. 4, p. 577–601, 2014.

DEB, K.; PRATAP, A.; AGARWAL, S.; MEYARIVAN, T. A fast and elitist multiobjective genetic algorithm: Nsga-ii. **Evolutionary Computation, IEEE Transactions on**, v. 6, n. 2, p. 182–197, 2002b.

DEB, K.; THIELE, L.; LAUMANNS, M.; ZITZLER, E. Scalable multi-objective optimization test problems. In: IEEE. **Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on**. [S.l.], 2002a. v. 1, p. 825–830.

DURANTIN, C.; MARZAT, J.; BALESDENT, M. Analysis of multi-objective kriging-based methods for constrained global optimization. **Computational Optimization and Applications**, v. 63, n. 3, p. 903–926, 2016.

DUTTA, J.; DEB, K.; TULSHYAN, R.; ARORA, R. Approximate kkt points and a proximity measure for termination. **Journal of Global Optimization**, v. 56, n. 4, p. 1463–1499, 2013.

DWIGHT, R.; HAN, Z.-H. Efficient uncertainty quantification using gradient-enhanced kriging. In: **50th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference 17th AIAA/ASME/AHS Adaptive Structures Conference 11th AIAA No**. [S.l.: s.n.], 2009. p. 2276.

EMMERICH, M.; BEUME, N.; NAUJOKS, B. An emo algorithm using the hypervolume measure as selection criterion. In: SPRINGER. **International Conference on Evolutionary Multi-Criterion Optimization**. [S.l.], 2005. p. 62–76.

EMMERICH, M.; DEUTZ, A. H.; KLINKENBERG, J. W. Hypervolume-based expected improvement: Monotonicity properties and exact computation. In: IEEE. **2011 IEEE Congress of Evolutionary Computation (CEC)**. [S.l.], 2011. p. 2147–2154.

EMMERICH, M. T.; GIANNAKOGLOU, K. C.; NAUJOKS, B. Single-and multiobjective evolutionary optimization assisted by gaussian random field metamodels. **IEEE Transactions on Evolutionary Computation**, v. 10, n. 4, p. 421–439, 2006.

FELIOT, P.; BECT, J.; VAZQUEZ, E. A bayesian approach to constrained single-and multi-objective optimization. **Journal of Global Optimization**, v. 67, n. 1-2, p. 97–133, 2017.

FIELDSEND, J.; EVERSON, R. Visualising high-dimensional pareto relationships in two-dimensional scatterplots. In: SPRINGER. **Evolutionary Multi-Criterion Optimization**. [S.l.], 2013. p. 558–572.

FONSECA, C. M.; FLEMING, P. J. An overview of evolutionary algorithms in multiobjective optimization. **Evolutionary computation**, v. 3, n. 1, p. 1–16, 1995.

FORRESTER, A.; SOBESTER, A.; KEANE, A. **Engineering design via surrogate modelling: a practical guide**. Pondicherry, India: [s.n.], 2008.

FORRESTER, A. I.; KEANE, A. J. Recent advances in surrogate-based optimization. **Progress in Aerospace Sciences**, v. 45, n. 1, p. 50–79, 2009.

FU, G.; KHU, S.-T.; BUTLER, D. Multiobjective optimisation of urban wastewater systems using parego: a comparison with nsga ii. In: **11th International Conference on Urban Drainage, Edinburgh, Scotland**. [S.l.: s.n.], 2008.

GAO, Z.; SHAO, X.; JIANG, P.; WANG, C.; ZHOU, Q.; CAO, L.; WANG, Y. Multi-objective optimization of weld geometry in hybrid fiber laser-arc butt welding using kriging model and nsga-ii. **Applied Physics A**, v. 122, n. 6, p. 1–12, 2016.

HAMDAOUI, M.; OUJEBBOUR, F.-Z.; HABBAL, A.; BREITKOPF, P.; VILLON, P. Kriging surrogates for evolutionary multi-objective optimization of cpu intensive sheet metal forming applications. **International Journal of Material Forming**, v. 8, n. 3, p. 469–480, 2015.

HORNIK, K. et al. **The r FAQ**. [S.l.], 2002.

HUPKENS, I.; EMMERICH, M.; DEUTZ, A. **Faster Computation of Expected Hypervolume Improvement**. [S.l.], 2014.

HUSSEIN, R.; DEB, K. A generative kriging surrogate model for constrained and unconstrained multi-objective optimization. In: ACM. **Proceedings of the 2016 on Genetic and Evolutionary Computation Conference**. [S.l.], 2016. p. 573–580.

IBRAHIM, A.; RAHNAMAYAN, S.; MARTIN, M. V.; DEB, K. Elitensga-iii: An improved evolutionary many-objective optimization algorithm. In: IEEE. **Evolutionary Computation (CEC), 2016 IEEE Congress on**. [S.l.], 2016. p. 973–982.

JAIN, H.; DEB, K. An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: Handling constraints and extending to an adaptive approach. **IEEE Trans. Evolutionary Computation**, v. 18, n. 4, p. 602–622, 2014.

JONES, D. R. A taxonomy of global optimization methods based on response surfaces. **Journal of global Optimization**, v. 21, n. 4, p. 345–383, 2001.

JONES, D. R.; SCHONLAU, M.; WELCH, W. J. Efficient global optimization of expensive black-box functions. **Journal of Global Optimization**, v. 13, n. 4, p. 455–492, 1998.

KLEIJNEN, J. P.; MEHDAD, E. Multivariate versus univariate kriging metamodels for multi-response simulation models. **European Journal of Operational Research**, v. 236, n. 2, p. 573–582, 2014.

KNOWLES, J. Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. **Evolutionary Computation, IEEE Transactions on**, v. 10, n. 1, p. 50–66, 2006.

KNOWLES, J.; HUGHES, E. J. Multiobjective optimization on a budget of 250 evaluations. In: SPRINGER. **International Conference on Evolutionary Multi-Criterion Optimization**. [S.l.], 2005. p. 176–190.

KRIEG, D. A statistical approach to some basic mine valuation problems on the Witwatersrand. **Journal of Chemical, Metallurgical, and Mining Society of South Africa**, v. 52, n. 6, p. 119–139, 1951.

LAURENCEAU, J.; SAGAUT, P. Building efficient response surfaces of aerodynamic functions with kriging and cokriging. **AIAA journal**, v. 46, n. 2, p. 498–507, 2008.

LI, M. An improved kriging-assisted multi-objective genetic algorithm. **Journal of Mechanical Design**, v. 133, n. 7, p. 071008, 2011.

MARTÍNEZ-FRUTOS, J.; HERRERO-PÉREZ, D. Kriging-based infill sampling criterion for constraint handling in multi-objective optimization. **Journal of Global Optimization**, v. 64, n. 1, p. 97–115, 2016.

MATHERON, G. Principles of geostatistics. **Economic geology**, v. 58, n. 8, p. 1246–1266, 1963.

MERSMANN, O. **MCO: Multiple Criteria Optimization Algorithms and Related Functions**. [S.l.], 2014. R package version 1.0-15.1. Disponível em: <http://CRAN.R-project.org/package=mco>.

PASSOS, A. G. **moko: Multi-Objective Kriging Optimization**. [S.l.], 2016. R package version 1.0.1. Disponível em: <https://CRAN.R-project.org/package=moko>.

PASSOS, A. G.; LUERSEN, M. A. Kriging-based multiobjective optimization of a fuselage-like composite section with curvilinear fibers. In: **EngOpt 2016 - 5th International Conference on Engineering Optimization**. [S.l.: s.n.], 2015.

PASSOS, A. G.; LUERSEN, M. A. Moko: An open source package for multi-objective optimization with kriging surrogates. In: **MECSOL 2017 - 6th International Symposium on Solid Mechanics**. [S.l.: s.n.], 2017.

PASSOS, A. G.; LUERSEN, M. A. Multiobjective optimization of laminated composite parts with curvilinear fibers using kriging-based approaches. **Structural and Multidisciplinary Optimization**, p. 1–13, 2017.

PASSOS, A. G.; LUERSEN, M. A.; STEEVES, C. A. Optimal curved fibre orientations of a composite panel with cutout for improved buckling load using the efficient global optimization algorithm. **Engineering Optimization**, p. 1–19, 2016.

PICHENY, V. Multiobjective optimization using gaussian process emulators via stepwise uncertainty reduction. **Statistics and Computing**, v. 25, n. 6, p. 1265–1280, 2015.

PONWEISER, W.; WAGNER, T.; BIERMANN, D.; VINCZE, M. Multiobjective optimization on a limited budget of evaluations using model-assisted\ mathcal {S}-metric selection. In: SPRINGER. **International Conference on Parallel Problem Solving from Nature**. [S.l.], 2008. p. 784–794.

REED, P. M.; HADKA, D.; HERMAN, J. D.; KASPRZYK, J. R.; KOLLAT, J. B. Evolutionary multiobjective optimization in water resources: The past, present, and future. **Advances in water resources**, v. 51, p. 438–456, 2013.

ROUSTANT, O.; GINSBOURGER, D.; DEVILLE, Y. DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by kriging-based metamodeling and optimization. **Journal of Statistical Software**, v. 51, n. 1, p. 1–55, 2012.

SACKS, J.; WELCH, W. J.; MITCHELL, T. J.; WYNN, H. P. Design and analysis of computer experiments. **Statistical science**, p. 409–423, 1989.

SCHEUERER, M.; SCHABACK, R.; SCHLATHER, M. Interpolation of spatial data–a stochastic or a deterministic problem? **European Journal of Applied Mathematics**, v. 24, n. 4, p. 601–629, 2013.

SEADA, H.; ABOUHAWWASH, M.; DEB, K. Towards a better balance of diversity and convergence in nsga-iii: First results. In: SPRINGER. **International Conference on Evolutionary Multi-Criterion Optimization**. [S.l.], 2017. p. 545–559.

SHAH, A.; GHAHRAMANI, Z. Pareto frontier learning with expensive correlated objectives. In: **Proceedings of The 33rd International Conference on Machine Learning**. [S.l.: s.n.], 2016. p. 1919–1927.

SHIMOYAMA, K.; JEONG, S.; OBAYASHI, S. Kriging-surrogate-based optimization considering expected hypervolume improvement in non-constrained many-objective test problems. In: IEEE. **Evolutionary Computation (CEC), 2013 IEEE Congress on**. [S.l.], 2013. p. 658–665.

SVENSON, J.; SANTNER, T. Multiobjective optimization of expensive-to-evaluate deterministic computer simulator models. **Computational Statistics & Data Analysis**, v. 94, p. 250–264, 2016.

SVENSON, J. D. **Computer experiments: Multiobjective optimization and sensitivity analysis**. Tese (Doutorado) — The Ohio State University, 2011.

TABATABAEI, M.; HAKANEN, J.; HARTIKAINEN, M.; MIETTINEN, K.; SINDHYA, K. A survey on handling computationally expensive multiobjective optimization problems using surrogates: non-nature inspired methods. **Structural and Multidisciplinary Optimization**, v. 52, n. 1, p. 1–25, 2015.

TORENBEEK, E. **Advanced aircraft design: Conceptual design, technology and optimization of subsonic civil airplanes**. [S.l.: s.n.], 2013.

TULSHYAN, R.; ARORA, R.; DEB, K.; DUTTA, J. Investigating ea solutions for approximate kkt conditions in smooth problems. In: ACM. **Proceedings of the 12th annual conference on Genetic and evolutionary computation**. [S.l.], 2010. p. 689–696.

VELDHUIZEN, D. A. V.; LAMONT, G. B. **Multiobjective evolutionary algorithm research: A history and analysis**. [S.l.], 1998.

VILLEMONTEIX, J.; VAZQUEZ, E.; WALTER, E. An informational approach to the global optimization of expensive-to-evaluate functions. **Journal of Global Optimization**, v. 44, n. 4, p. 509–534, 2009.

WAGNER, T.; EMMERICH, M.; DEUTZ, A.; PONWEISER, W. On expected-improvement criteria for model-based multi-objective optimization. **Parallel Problem Solving from Nature, PPSN XI**, p. 718–727, 2010.

WICKHAM, H. **R packages**. [S.l.]: "O'Reilly Media, Inc.", 2015.

WICKHAM, H.; CHANG, W. **devtools: Tools to Make Developing R Packages Easier**. [S.l.], 2016. R package version 1.11.1. Disponível em: <https://CRAN.R-project.org/package=devtools>.

WICKHAM, H.; DANENBERG, P.; EUGSTER, M. **roxygen2: In-Source Documentation for R**. [S.l.], 2015. R package version 5.0.1. Disponível em: <https://CRAN.R-project.org/package=roxygen2>.

YANG, X.-S. Multiobjective firefly algorithm for continuous optimization. **Engineering with Computers**, v. 29, n. 2, p. 175–184, 2013.

Yang Xiang; GUBIAN, S.; SUOMELA, B.; HOENG, J. Generalized simulated annealing for efficient global optimization: the GenSA package for R. **The R Journal Volume 5/1, June 2013**, 2013. Disponível em: <http://journal.r-project.org/>.

YUAN, Y.; XU, H.; WANG, B. An improved nsga-iii procedure for evolutionary many-objective optimization. In: ACM. **Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation**. [S.l.], 2014. p. 661–668.

ZHANG, J.-x.; MA, Y.-z.; ZHU, L.-y. Multiobjective simulation optimization using stochastic kriging. In: SPRINGER. **Proceedings of the 22nd International Conference on Industrial Engineering and Engineering Management 2015**. [S.l.], 2016. p. 81–91.

ZHANG, M.; GOU, W.; LI, L.; YANG, F.; YUE, Z. Multidisciplinary design and multi-objective optimization on guide fins of twin-web disk using kriging surrogate model. **Structural and Multidisciplinary Optimization**, v. 55, n. 1, p. 361–373, 2017.

ZHANG, Q.; LI, H. Moea/d: A multiobjective evolutionary algorithm based on decomposition. **IEEE Transactions on evolutionary computation**, v. 11, n. 6, p. 712–731, 2007.

ZITZLER, E. **Evolutionary algorithms for multiobjective optimization methods and applications**. [S.l.: s.n.], 1999. v. 63.

ZITZLER, E.; LAUMANNS, M.; THIELE, L.; ZITZLER, E.; ZITZLER, E.; THIELE, L.; THIELE, L. **SPEA2: Improving the strength Pareto evolutionary algorithm**. [S.l.], 2001.

# APÊNDICE A – DOCUMENTAÇÃO DO PACOTE 'MOKO'

# Package 'moko'

September 17, 2016

**Type** Package

**Title** Multi-Objective Kriging Optimization

**Version** 1.0.0

**Description** Multi-Objective optimization based on the Kriging metamodel. Important functions: mkm, VMPF, MEGO and HEGO.

**Depends** R (>= 3.3.0)

**License** GPL-3

**LazyData** TRUE

**Imports** DiceKriging (>= 1.5.5), DiceOptim (>= 1.5), GenSA (>= 1.1.6), emoa (>= 0.5.0), mco (>= 1.0.15.1), GPareto (>= 1.0.2), methods (>= 3.0.0)

**RoxygenNote** 5.0.1

**URL** https://github.com/coldfir3/moko

**BugReports** https://github.com/coldfir3/moko/issues

**Suggests** testthat, knitr, rmarkdown, lhs

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Adriano Passos [aut, cre], Marco Luersen [ctb]

**Maintainer** Adriano Passos <adriano.utfpr@gmail.com>

**Repository** CRAN

**Date/Publication** 2016-09-17 03:09:13

## R topics documented:

---

EHVI *EHVI: Constrained Expected Hypervolume Improvement*

---

## Description

Multi-objective Expected Hypervolume Improvement with respect to the current Pareto front. It's based on the `crit_EHI` function of the `GPareto` package. However, the present implementation accounts for inequalty constrains embeded into the `mkm` model.

## Usage

```
EHVI(x, model, control = NULL)
```

## Arguments

| | |
|---|---|
| `x` | a vector representing the input for which one wishes to calculate `EHI`, |
| `model` | An object of class `mkm`. |
| `control` | An optional list of control parameters, some of them passed to the `crit_EHI` function. One can control:<br>`minimization` logical indicating if the EHVI is minimizing all objectives (`TRUE`, by default) or maximizing all objectives (`FALSE`). Mixed optimization is not currently accepted, if the user needs it, the cost functions should be modified prior Kriging modeling (i.e. inverting or mutilpying the optut by `-1`).<br>`paretoFront` object of class `ps` containing the actual Pareto set. If not provided a Pareto set is built based on the current feasible observations (`model@response[model@feasible,]`<br>`nb.samp` number of random samples from the posterior distribution (with more than two objectives), default to 50, increasing gives more reliable results at the cost of longer computation time |

`seed` seed used for the random samples (with more than two objectives);

`refPoint` reference point for Hypervolume Expected Improvement. If not provided, it is set to the maximum or minimum of each objective.

## Details

The way that the constraints are handled are based on the probability of feasibility. The strong assumption here is that the cost functions and the constraints are uncorrelated. With that assumption in mind, a simple closed-form solution can be derived that consists in the product of the probability that each constraint will be met and the expected improvemen of the objective.

## Value

The constrained expected hypervolume improvement at `x`.

## References

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

## Examples

```
# ------------------------
# The Nowacki Beam
# ------------------------
n <- 20
d <- 2
doe <- replicate(d,sample(0:n,n))/n
res <- t(apply(doe, 1, nowacki_beam, box = data.frame(b = c(10, 50), h = c(50, 250))))
model <- mkm(doe, res, modelcontrol = list(objective = 1:2, lower=rep(0.1,d)))
grid <- expand.grid(seq(0, 1, , 20),seq(0, 1, , 20))
ehvi <- apply(grid, 1, EHVI, model)
contour(matrix(ehvi, 20))
points(model@design, col=ifelse(model@feasible,'blue','red'))
points(grid[which.max(ehvi),], col='green', pch=19)
```

---

# EI *Constrained Expected Emprovement*

---

## Description

This functions extends the `EI` function supplied by the package `DiceOptim`. This enxtension allows usage of multiple expensive constraints. The constraints are passed to the revamped EI function embedded inside the `mkm` object. Currently low-cost (explicit) constraints are not allowed.

## Usage

```
EI(x, model, control = NULL)
```

## Arguments

| | |
|---|---|
| x | A vector representing the input for which one wishes to calculate EI. |
| model | An object of class mkm. This model must have a single objective (model@m == 1). |
| control | An optional list of control parameters, some of them passed to the EI function. One can control: |

minimization logical specifying if EI is used in minimiziation or in maximization (default: TRUE)

plugin optional scalar, if not provided, the minimum (or maximum) of the current feasible observations. If there isn't any feasible design plugin is set to NA and the algorithm returns the value of the probabilty of constraints be met.

envir optional enviroment specifying where to assign intermediate values. Default: NULL.

## Details

The way that the constraints are handled are based on the probability of feasibility. The strong assumption here is that the cost functions and the constraints are uncorrelated. With that assumption in mind, a simple closed-form solution can be derived that consists in the product of the probability that each constraint will be met and the expected improvemen of the objective. Another important consideration is that, by default, the value of the pluging passed to the EI is the best *feasible* observed value.

## References

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

## Examples

```
# ------------------------------------------
# Branin-Hoo function (with simple constraint)
# ------------------------------------------
n <- 10
d <- 2
doe <- replicate(d,sample(0:n,n))/n
fun_cost <- DiceKriging::branin
fun_cntr <- function(x) 0.2 - prod(x)
fun <- function(x) return(cbind(fun_cost(x),fun_cntr(x)))
res <- t(apply(doe, 1, fun))
model <- mkm(doe, res, modelcontrol = list(objective = 1, lower=c(0.1,0.1)))
grid <- expand.grid(seq(0,1,,25),seq(0,1,,25))
ei <- apply(grid, 1, EI, model) # this computation may take some time
contour(matrix(ei,25))
points(model@design, col=ifelse(model@feasible,'blue','red'))
points(grid[which.max(ei),], col='green')
```

HEGO *HEGO: Efficient Global Optimization Algorithm based on the Hypervolume criteria*

## Description

Executes `nsteps` iterations of the HEGO method to an object of class `mkm`. At each step, a kriging model is re-estimated (including covariance parameters re-estimation) based on the initial design points plus the points visited during all previous iterations; then a new point is obtained by maximizing the Expected Hypervolume Improvement criterion (EHVI).

## Usage

```
HEGO(model, fun, nsteps, lower = rep(0, model@d), upper = rep(1, model@d),
  quiet = TRUE, control = NULL, optimcontrol = NULL)
```

## Arguments

| | |
|---|---|
| `model` | An object of class `mkm`. |
| `fun` | The multi-objective and constraint cost function to be optimized. This function must return a vector with the size of `model@m + model@j` where `model@m` are the number of objectives and `model@j` the number of the constraints, |
| `nsteps` | An integer representing the desired number of iterations, |
| `lower` | Vector of lower bounds for the variables to be optimized over (default: 0 with length `model@d`), |
| `upper` | Vector of upper bounds for the variables to be optimized over (default: 1 with length `model@d`), |
| `quiet` | Logical indicating the verbosity of the routine, |
| `control` | An optional list of control parameters, some of them passed to the `crit_EHI` function. One can control:<br>`minimization` logical indicating if the EHVI is minimizing all objectives (`TRUE`, by default) or maximizing all objectives (`FALSE`). Mixed optimization is not currently accepted, if the user needs it, the cost functions should be modified prior Kriging modeling (i.e. inverting or mutilpying the optut by `-1`).<br>`paretoFront` object of class `ps` containing the actual Pareto set. If not provided a Pareto set is built based on the current feasible observations (`model@response[model@feasible,]`<br>`nb.samp` number of random samples from the posterior distribution (with more than two objectives), default to 50, increasing gives more reliable results at the cost of longer computation time<br>`seed` seed used for the random samples (with more than two objectives);<br>`refPoint` reference point for Hypervolume Expected Improvement. If not provided, it is set to the maximum or minimum of each objective. |
| `optimcontrol` | Optional list of control parameters passed to the `GenSA` function. Please, note that the values are passed as the `control` parameter inside the `GenSA` function (`genSA(control = optimcontrol)`). |

## Value

updated mkm model

## Examples

```
# ----------------
# The Nowacki Beam
# ----------------
n <- 20
d <- 2
nsteps <- 1 # value has been set to 1 to save compliation time, change this value to 40.
fun <- nowacki_beam
doe <- replicate(d,sample(0:n,n))/n
res <- t(apply(doe, 1, fun))
model <- mkm(doe, res, modelcontrol = list(objective = 1:2, lower = rep(0.1,d)))
model <- HEGO(model, fun, nsteps, quiet = FALSE)
plot(nowacki_beam_tps$set)
points(ps(model@response[which(model@feasible),model@objective])$set, col = 'green', pch = 19)
```

---

# igd *IGD: Inverted Generational Distance*

---

## Description

The IGD is a perfomance measure function of Pareto front fidelity and corresponds to the average distance between all designs in the true set and the closest design of the current set. Thus, the lower the IGD value, the better the front is.

## Usage

```
igd(aps, tps, method = "manhattan", norm = TRUE)
```

## Arguments

| | |
|---|---|
| aps | An object of type ps containing the "actual" pareto front |
| tps | An object of type ps containing the "true" pareto front |
| method | String stating which distance measure to be used. This must be one of: "euclidean" or "manhattan" (default). |
| norm | Logical (default: TRUE) indicating if both fronts should be normalized. |

## Value

returns the IGD metric

## References

Shimoyama, K., Jeong, S., & Obayashi, S. (2013, June). Kriging-surrogate-based optimization considering expected hypervolume improvement in non-constrained many-objective test problems. In 2013 IEEE *Congress on Evolutionary Computation* (pp. 658-665). IEEE.

## Examples

```
aps <- ps(matrix(rnorm(1:1000),ncol=2))
tps <- ps(matrix(rnorm(1:2000),ncol=2))
igd(aps,tps)

tps <-nowacki_beam_tps$set[1:50 * 10,]
aps <- tps * 1.2
igd(aps,tps)
```

---

`max_EHVI` *max_EHVI: Maximization of the Expected Hypervolume Improvement criterion*

---

## Description

Given an object of class `mkm` and a set of tuning parameters, max_EHVI performs the maximization of the Expected Hypervolume Improvement criterion and delivers the next point to be visited in an HEGO-like procedure.

## Usage

```
max_EHVI(model, lower = rep(0, model@d), upper = rep(1, model@d),
  control = NULL, optimcontrol = NULL)
```

## Arguments

| | |
|---|---|
| `model` | An object of class `mkm`. |
| `lower` | Vector of lower bounds for the variables to be optimized over (default: 0 with length `model@d`), |
| `upper` | Vector of upper bounds for the variables to be optimized over (default: 1 with length `model@d`), |
| `control` | An optional list of control parameters, some of them passed to the `crit_EHI` function. One can control: |

`minimization` logical indicating if the EHVI is minimizing all objectives (`TRUE`, by default) or maximizing all objectives (`FALSE`). Mixed optimization is not currently accepted, if the user needs it, the cost functions should be modified prior Kriging modeling (i.e. inverting or mutilpying the optut by `-1`).

`paretoFront` object of class `ps` containing the actual Pareto set. If not provided a Pareto set is built based on the current feasible observations (`model@response[model@feasible,]`

`nb.samp` number of random samples from the posterior distribution (with more than two objectives), default to 50, increasing gives more reliable results at the cost of longer computation time

`seed` seed used for the random samples (with more than two objectives);

`refPoint` reference point for Hypervolume Expected Improvement. If not provided, it is set to the maximum or minimum of each objective.

optimcontrol — Optional list of control parameters passed to the GenSA function. Please, note that the values are passed as the control parameter inside the GenSA function (genSA(control = optimcontrol)).

## Value

A list with components:

par The best set of parameters found.

value The value of expected hypervolume improvement at par.

## Examples

```
# ------------------------
# The Nowacki Beam
# ------------------------
n <- 20
d <- 2
doe <- replicate(d,sample(0:n,n))/n
res <- t(apply(doe, 1, nowacki_beam, box = data.frame(b = c(10, 50), h = c(50, 250))))
model <- mkm(doe, res, modelcontrol = list(objective = 1:2, lower=c(0.1,0.1)))
max_EHVI(model)
```

---

# max_EI — *max_EI: Maximization of the Constrained Expected Improvement criterion*

---

## Description

Given an object of class mkm and a set of tuning parameters, max_EI performs the maximization of the Constrained Expected Improvement criterion and delivers the next point to be visited in an MEGO-like procedure.

## Usage

```
max_EI(model, lower = rep(0, model@d), upper = rep(1, model@d),
  control = NULL, optimcontrol = NULL)
```

## Arguments

model — An object of class mkm. This model must have a single objective (model@m == 1).

lower — Vector of lower bounds for the variables to be optimized over (default: 0 with length = model@d),

upper — Vector of upper bounds for the variables to be optimized over (default: 1 with length = model@d),

control — An optional list of control parameters, some of them passed to the EI function. One can control:

`minimization` logical specifying if EI is used in minimiziation or in maximization (default: `TRUE`)

`plugin` optional scalar, if not provided, the minimum (or maximum) of the current feasible observations. If there isn't any feasible design plugin is set to `NA` and the algorithm returns the value of the probabilty of constraints be met.

`envir` optional enviroment specifying where to assign intermediate values. Default: `NULL`.

`optimcontrol` Optional list of control parameters passed to the `GenSA` function. Please, note that the values are passed as the `control` parameter inside the `GenSA` function (`genSA(control = optimcontrol)`).

## Value

A list with components:

`par` The best set of parameters found.

`value` The value of expected hypervolume improvement at par.

Vector. The best set of parameters found.

## Examples

```
# ------------------------------------------
# Branin-Hoo function (with simple constraint)
# ------------------------------------------
n <- 10
d <- 2
doe <- replicate(d,sample(0:n,n))/n
fun_cost <- DiceKriging::branin
fun_cntr <- function(x) 0.2 - prod(x)
fun <- function(x) return(cbind(fun_cost(x),fun_cntr(x)))
res <- t(apply(doe, 1, fun))
model <- mkm(doe, res, modelcontrol = list(objective = 1, lower=c(0.1,0.1)))
max_EI(model)
```

---

`MEGO` *MEGO: Multi-Objective Efficient Global Optimization Algorithm based on scalarization of the objectives*

---

## Description

Executes `nsteps` iterations of the MEGO method to an object of class `mkm`. At each step, a weighted kriging model is re-estimated (including covariance parameters re-estimation) based on the initial design points plus the points visited during all previous iterations; then a new point is obtained by maximizing the Constrained Expected Improvement criterion (EI).

## Usage

```
MEGO(model, fun, nsteps, lower = rep(0, model@d), upper = rep(1, model@d),
  quiet = TRUE, control = NULL, optimcontrol = NULL)
```

## Arguments

| | |
|---|---|
| model | An object of class mkm. This model must have a single objective (model@m == 1). |
| fun | The multi-objective and constraint cost function to be optimized. This function must return a vector with the size of model@m + model@j where model@m are the number of objectives and model@j the number of the constraints, |
| nsteps | An integer representing the desired number of iterations, |
| lower | Vector of lower bounds for the variables to be optimized over (default: 0 with length = model@d), |
| upper | Vector of upper bounds for the variables to be optimized over (default: 1 with length = model@d), |
| quiet | Logical indicating the verbosity of the routine, |
| control | An optional list of control parameters, some of them passed to the EI function. One can control:<br>minimization logical specifying if EI is used in minimiziation or in maximization (default: TRUE)<br>plugin optional scalar, if not provided, the minimum (or maximum) of the current feasible observations. If there isn't any feasible design plugin is set to NA and the algorithm returns the value of the probabilty of constraints be met.<br>envir optional enviroment specifying where to assign intermediate values. Default: NULL. |
| optimcontrol | Optional list of control parameters passed to the GenSA function. Please, note that the values are passed as the control parameter inside the GenSA function (genSA(control = optimcontrol)). |

## Details

Note that since MEGO is works by scalarizing a cost function, this technique is well suited for single objective problems with multiple constraints.

## Value

updated mkm model

## References

Knowles, J. (2006). ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1), 50-66.

## Examples

```
# ----------------
# The Nowacki Beam
# ----------------
n <- 20
d <- 2
nsteps <- 1 # value has been set to 1 to save compliation time, change this value to 40.
fun <- nowacki_beam
doe <- replicate(d,sample(0:n,n))/n
res <- t(apply(doe, 1, fun))
model <- mkm(doe, res, modelcontrol = list(objective = 1:2, lower = rep(0.1,d)))
model <- MEGO(model, fun, nsteps, quiet = FALSE, control = list(rho = 0.1))
plot(nowacki_beam_tps$set)
points(ps(model@response[which(model@feasible),model@objective])$set, col = 'green', pch = 19)

###########################################
#### some single objective optimization ####
###########################################
## Not run:
## Those examples are flagged as "don't run" only to save compilation time. ##
n.grid <- 20
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
response.grid <- apply(design.grid, 1, DiceKriging::branin)
z.grid <- matrix(response.grid, n.grid, n.grid)

# ----------------------------------
# Branin-Hoo function (unconstrained)
# ----------------------------------
n <- 10
d <- 2
doe <- replicate(d,sample(0:n,n))/n
fun <- DiceKriging::branin
res <- apply(doe, 1, fun)
model <- mkm(doe, res, modelcontrol = list(lower=rep(0.1,d)))
model <- MEGO(model, fun, 10, quiet = FALSE)
contour(x.grid,y.grid,z.grid,40)
points(model@design, col=ifelse(model@feasible,'blue','red'))
# ---------------------------------------
# Branin-Hoo function (simple constraint)
# ---------------------------------------
n <- 10
d <- 2
doe <- replicate(d,sample(0:n,n))/n
fun_cost <- DiceKriging::branin
fun_cntr <- function(x) 0.2 - prod(x)
fun <- function(x) return(c(fun_cost(x),fun_cntr(x)))
res <- t(apply(doe, 1, fun))
model <- mkm(doe, res, modelcontrol = list(objective = 1, lower=rep(0.1,d)))
model <- MEGO(model, fun, 10, quiet = FALSE)
contour(x.grid,y.grid,z.grid,40)
points(model@design, col=ifelse(model@feasible,'blue','red'))
```

```
# ---------------------------------------
# Branin-Hoo function (narrow constraint)
# ---------------------------------------
n <- 10
d <- 2
doe <- replicate(d,sample(0:n,n))/n
fun_cost <- DiceKriging::branin
fun_cntr <- function(x){
 g1 <- 0.9 - sum(x)
 g2 <- sum(x) - 1.1
 g3 <- - x[1] + 0.75
 g4 <- x[2] - 0.25
 return(c(g1,g2,g3,g4))
}
fun <- function(x) return(c(fun_cost(x),fun_cntr(x)))
res <- t(apply(doe, 1, fun))
model <- mkm(doe, res, modelcontrol = list(objective = 1, lower=rep(0.1,d)))
model <- MEGO(model, fun, 10, quiet = FALSE)
contour(x.grid,y.grid,z.grid,40)
points(model@design, col=ifelse(model@feasible,'blue','red'))
# ---------------------------------------------
# Branin-Hoo function (disconnected constraint)
# ---------------------------------------------
n <- 10
d <- 2
doe <- replicate(d,sample(0:n,n))/n
Griewank <-  function(x) {
 ii <- c(1:length(x))
  sum <- sum(x^2/4000)
  prod <- prod(cos(x/sqrt(ii)))
  y <- sum - prod + 1
  return(y)
}
fun_cost <- DiceKriging::branin
fun_cntr <- function(x) 1.6 - Griewank(x*10-5)
fun <- function(x) return(c(fun_cost(x),fun_cntr(x)))
res <- t(apply(doe, 1, fun))
model <- mkm(doe, res, modelcontrol = list(objective = 1, lower=c(0.1,0.1)))
model <- MEGO(model, fun, 10, quiet = FALSE)
contour(x.grid,y.grid,z.grid,40)
points(model@design, col=ifelse(model@feasible,'blue','red'))

## End(Not run)
```

---

`mkm` *Multi-objective Kriging model*

---

## Description

This function creates a multi-objective kriging model. It is based on the `km` function of the `DiceKriging` package and creates a structured list of `km` objects.

## Usage

```
mkm(design, response, modelcontrol = NULL)
```

## Arguments

| | |
|---|---|
| design | Numeric data.frame of the designs (decision space) |
| response | Numeric data.frame of the observed responses (objectives and constraints) at each design point. |
| modelcontrol | An optional list of control parameters passed to the km function. One can control:<br>objective (default: 1:ncol(response))<br>quiet (default: TRUE)<br>formula (default: ~1)<br>covtype (default: "matern5_2")<br>nugget.estim (default: FALSE)<br>estim.method (default: "MLE")<br>optim.method (default: "BFGS")<br>multistart (default: 1)<br>gr (default: TRUE)<br>iso (default: FALSE)<br>scaling (default: FALSE)<br>type (default: 'UK')<br>se.compute (default: TRUE)<br>light.return (default: TRUE)<br>bias.correct (default: FALSE)<br>checkNames (default: FALSE)<br>For more details, one can check km. |

## Value

S4 An object of class mkm-class

## Examples

```
# ------------------------
# The Nowacki Beam
# ------------------------
n <- 10
d <- 2
doe <- replicate(d,sample(0:n,n))/n
res <- t(apply(doe, 1, nowacki_beam))
model <- mkm(doe, res, modelcontrol = list(objective = 1:2))
```

## mkm-class *A S4 class of multiple Kriging models*

### Description

A S4 class of multiple Kriging models

### Usage

```
## S4 method for signature 'mkm'
show(object)
```

### Arguments

| | |
|---|---|
| object | A mkm object. |

### Methods (by generic)

- show: Custom print for mkm objects

### Slots

km A list of km objectives.

objective A Numeric vector representing the index of the objective models in km.

design Numeric data.frame of the designs (decision space).

d,n,m,j Numeric values for the number of dimensions, designs, objectives and constraints, respectively.

response Numeric data.frame of the observed responses (objectives and constraints) at each design point.

feasible Logical vector stating which designs are feasible.

control A list of controls for function backtracking, this list contains all the input parameters that are passed to the km function.

## moko *moko: Multi-objective Kriging Optimization*

### Description

The package moko provides the user with methods for constrained and unconstraiend multi-objective optimization based on the popular Kriging surrogate model.

### Details

The main functions provided by moko are: MEGO, HEGO and VMPF.

nowacki_beam *Test function: The Nowacki Beam*

## Description

This function is a variation of the classic multi-objective optimization problem (NOWACKI, 1980). In this problem the aim is to design a tip loaded cantilever eam for minimum cross-sectional aera and lowest bending stress subject to a number of constraints.

## Usage

```
nowacki_beam(x, g = c(5, 240, 120, 10, 2), l = 1500, F = 5000,
  E = 216620, G = 86650, v = 0.27, box = data.frame(b = c(10, 50), h =
  c(20, 250)))
```

## Arguments

| | |
|---|---|
| x | vector of length 2 correspon the normalized beath and height of the beam |
| g | vector of lenght 5 containing the upper limits of each constraint |
| l | numeric length of the beam |
| F | numeric force applied at the beam tip |
| E | numeric elastic longitudinal moduli |
| G | numeric elastic transversal moduli |
| v | numeric poison ratio |
| box | data.frame structure containing the upper and lower limits for b and h |

## Value

vector of objective and constrain responses

## References

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

## Examples

```
grid <- expand.grid(seq(0, 1, , 50),seq(0, 1, , 50))
res <- apply(grid, 1, nowacki_beam, box = data.frame(b = c(10, 50), h = c(50, 250)))
par(mfrow = c(3,3))
for(i in 1:nrow(res))
 contour(matrix(res[i,],50))
```

## nowacki_beam_tps *True pareto front for the nowacki beam problem*

### Description

True pareto front for the nowacki beam problem

### Usage

```
nowacki_beam_tps
```

### Format

An object of class ps of length 4.

## pdist *Distance betwen vector and matrix*

### Description

This function computes and returns the minimum distance between a vector and a matrix

### Usage

```
pdist(point, set, method = "manhattan")
```

### Arguments

| | |
|---|---|
| point | numeric vector |
| set | numeric matrix |
| method | String stating which distance measure to be used. This must be one of: "euclidean" or "manhattan" (default). |

### Value

numeric value indicating the minimum distance between point and set.

predict,mkm-method *Predictor for a multiobjective Kriging model*

## Description

This functions performs predictions for a given dataset into a collection of Kriging models (mkm object)

## Usage

```
## S4 method for signature 'mkm'
predict(object, newdata, modelcontrol = NULL)
```

## Arguments

| | |
|---|---|
| object | An object of class mkm |
| newdata | a vector, matrix or data frame containing the points where to perform predictions. |
| modelcontrol | An optional list of control parameters to the mkm function (default: object@control). |

## Examples

```
# ------------------------
# The Nowacki Beam
# ------------------------
n <- 100
d <- 2
N <- 50
doe <- replicate(d,sample(0:n,n))/n
res <- t(apply(doe, 1, nowacki_beam))
model <- mkm(doe, res, modelcontrol = list(objective = 1:2, lower = rep(0.01, d)))
newx <- expand.grid(replicate(d,seq(0,1,,N),FALSE))
pred <- predict(model, newx)
realv <- t(apply(newx, 1, nowacki_beam))
par(mfrow=c(2,3), mar=c(2,2,1,1))
for (i in 1:6){
  contour(matrix((realv[,i]),N), col='red', lty=2, labels='')
  contour(matrix((pred$mean[,i]),N), add = TRUE)
}
```

predict_front *Predicted Pareto front*

## Description

This function creates a predicted pareto front based on the mean of Kriging models. The predicted mean of each objective and constraint is passed to the nsga2 algorithm that builds .

## Usage

```
predict_front(model, lower, upper, control = NULL, modelcontrol = NULL)
```

## Arguments

| | |
|---|---|
| model | Object of class mkm. |
| lower | Vector of lower bounds for the variables to be optimized over (default: 0 with length model@d). |
| upper | Vector of upper bounds for the variables to be optimized over (default: 1 with length model@d). |
| control | An optional list of control parameters that controlls the optimization algorithm. One can control:<br>popsize (default: 200);<br>generations (default: 30);<br>cdist (default: 1/model@d);<br>mprob (default: 15);<br>mdist (defult: 20). |
| modelcontrol | An optional list of control parameters to the mkm function (default: object@control). |

## Value

object of class ps containing the predicted Pareto front

## Examples

```
# ------------------------
# The Nowacki Beam
# ------------------------
n <- 100
doe <- cbind(sample(0:n,n),sample(0:n,n))/n
res <- t(apply(doe, 1, nowacki_beam))
model <- mkm(doe, res, modelcontrol = list(objective = 1:2, lower=c(0.1,0.1)))
pf <- predict_front(model, c(0,0), c(1,1))
plot(nowacki_beam_tps$set)
points(pf$set, col='blue')
```

## ps *Creates a pareto set from given data*

### Description

Return those points which are not dominated by another point in y This is the Pareto front approximation of the design set.

### Usage

```
ps(y, minimization = TRUE, light.return = FALSE)
```

### Arguments

| | |
|---|---|
| y | design space data |
| minimization | logical representing if the set is to be minimized or not |
| light.return | logical indicating if the indexes should be writen on the 'ps' object |

### Value

S3 class object that contains information of the Pareto set

### Examples

```
aps <- ps(matrix(rnorm(1:1000),ncol=2))
print(aps)
```

## radviz *Plot a multiresponse or multivariate dataset indo a 2d radViz graph*

### Description

Description

### Usage

```
radviz(data, ...)
```

### Arguments

| | |
|---|---|
| data | data.frame containing the variables or observations to be ploted |
| ... | opitional plotting argumentos passed to points function. |

## Examples

```
data <- data.frame(matrix(rnorm(1:50),ncol=5))
radviz(data, col='red')
```

---

Tchebycheff *Augmented Tchebycheff function*

---

## Description

The Augmented Tchebycheff function (KNOWLES, 2006) is a scalarizing function witch the advantages of having a non-linear term. That causes points on nonconvex regions of the Pareto front can bve minimizers of this function and, thus, nonsupported solutions can be obtained.

## Usage

```
Tchebycheff(y, s = 100, rho = 0.1)
```

## Arguments

| | |
|---|---|
| y | Numerical matrix or data.frame containing the responses (on each column) to be scalarized. |
| s | Numerical integer (default: 100) setting the number of partitions the vector lambda has. |
| rho | A small positive value (default: 0.1) setting the "strenght" of the non-linear term. |

## References

Knowles, J. (2006). ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1), 50-66.

## Examples

```
grid <- expand.grid(seq(0, 1, , 50),seq(0, 1, , 50))
 res <- t(apply(grid, 1, nowacki_beam))
 plot(nowacki_beam_tps$x, xlim=c(0,1), ylim=c(0,1))
 grid <- grid[which(as.logical(apply(res[,-(1:2)] < 0, 1, prod))),]
 res <- res[which(as.logical(apply(res[,-(1:2)] < 0, 1, prod))),1:2]
for (i in 1:10){
sres <- Tchebycheff(res[,1:2], s=100, rho=0.1)
points(grid[which.min(sres),], col='green')
}
```

test_functions *Test functions for optimization*

## Description

This page is a collection of test functions commonly used to test optimization algorithms

## Usage

```
Shaffer1(x)

Shaffer2(x)

Fonseca(x)

Kursawe(x)

Viennet(x)

Binh(x)
```

## Arguments

| | |
|---|---|
| x, | numeric value (or vector for multivariable functions) |

## References

https://en.wikipedia.org/wiki/Test_functions_for_optimization

http://www.sfu.ca/~ssurjano/optimization.html

## Examples

```
#function should be evaluated in the -A < x < A interval,
#where A is from 10 to 10^5 and \length(x) = 1
Shaffer1(0)

#function should be evaluated in the -5 < x < 10 interval \length(x) = 1
Shaffer2(0)

#function should be evaluated in the -20 < x < 20 interval and \length(x) >= 1
Fonseca(rep(0,10))

#function should be evaluated in the -5 < x < 5 interval and \length(x) == 3
Kursawe(rep(0,3))

#function should be evaluated in the -3 < x < 3 interval and \length(x) == 2
Viennet(c(0.5,0.5))
```

```
#function should be evaluated in the 0 < x < (5,3) interval and \length(x) == 2
Binh(c(0,0))
```

---

VMPF *VMPF: Variance Minimization of the Predicted Front*

---

## Description

Executes `nsteps` iterations of the VMPF algorithm to an object of class `mkm`. At each step, a multi-objective kriging model is re-estimated (including covariance parameters re-estimation).

## Usage

```
VMPF(model, fun, nsteps, lower = rep(0, model@d), upper = rep(1, model@d),
  quiet = TRUE, control = NULL, modelcontrol = NULL)
```

## Arguments

| | |
|---|---|
| `model` | An object of class `mkm`, |
| `fun` | The multi-objective and constraint cost function to be optimized. This function must return a vector with the size of `model@m + model@j` where `model@m` are the number of objectives and `model@j` the number of the constraints, |
| `nsteps` | An integer representing the desired number of iterations, |
| `lower` | Vector of lower bounds for the variables to be optimized over (default: 0 with length `model@d`), |
| `upper` | Vector of upper bounds for the variables to be optimized over (default: 1 with length `model@d`), |
| `quiet` | Logical indicating the verbosity of the routine, |
| `control` | An optional list of control parameters that controlls the optimization algorithm. One can control:<br>`popsize` (default: `200`);<br>`generations` (default: `30`);<br>`cdist` (default: `1/model@d`);<br>`mprob` (default: `15`);<br>`mdist` (defult: `20`). |
| `modelcontrol` | An optional list of control parameters to the `mkm` function (default: `object@control`). |

## Details

The infill point is sampled from the most uncertain design of a predicted Pareto set. This set is predicted using nsga-2 algorithm and the mean value of the mkm predictor.

## Value

an updated object of class mkm.

## Examples

```
# ----------------
# The Nowacki Beam
# ----------------
n <- 20
d <- 2
nsteps <- 2 # value has been set to 2 to save compliation time, change this value to 40.
fun <- nowacki_beam
doe <- replicate(d,sample(0:n,n))/n
res <- t(apply(doe, 1, fun))
model <- mkm(doe, res, modelcontrol = list(objective = 1:2, lower = rep(0.1,d)))
model <- VMPF(model, fun, nsteps, quiet = FALSE)
plot(nowacki_beam_tps$set)
points(ps(model@response[which(model@feasible),model@objective])$set, col = 'green', pch = 19)
```

# Index

# APÊNDICE B – VINHETA DO PACOTE MOKO

# Multiobjective Optimization of the Nowacki Beam

*Adriano G. Passos*

*2016-10-20*

In this paper, the well known multiobjective optimization problem: The Nowacki beam is solved. here, three different frameworks for dealing with many-objective problems using Kriging surrogate models are compared:

1. The efficient global optimization (EGO) algorithm applied to a single objective function of the combined objectives responses (MEGO);
2. The iterative maximization of the expected hypervolume improvement (EHVI);
3. A novel approach is also proposed here, the variance minimization of the Kriging-predicted Pareto front (VMKF).

To evaluate the efficiency of these three methods, a baseline solution is created by multiobjective direct optimization (no surrogates are used) applying the NSGA-II algorithm.

## Introduction

Multiobjective optimization is a field of interest for many real-world applications. Usually, projects have multiple and conflicting goals and, most of the time, the relationship between the decision space (design variables) and the outcome is highly complex.

In the past years, Kriging have become one of the most popular surrogates on the industry (Forrester and Keane 2009). When using Kriging, usually the efficient global optimization (EGO) algorithm is the standard technique for single objective optimization. For costly multiple objectives, direct combination of the Kriging predictions and a multiobjective genetic algorithm (MOGA) can be used such as in (Li 2011). However, according to (Forrester and Keane 2009; Forrester, Sobester, and Keane 2008), there are currently two popular ways of constructing Pareto sets. The first approach, is to combine all goals into a single quantity and carry out the EGO algorithm. The weighting function have adjustable parameters that changes during the optimization problem so that the algorithm can potentially sweep all the objective space. For simplicity, here, this approach will be simply called MEGO. Another popular way to address many-objective problems is to generalize the expected improvement (EI) criterion into what is called the expected hypervolume improvement (EHVI) (Emmerich, Deutz, and Klinkenberg 2011; Shimoyama, Jeong, and Obayashi 2013). Although there are some efficient algorithms to calculate and/or estimate the expected hypervolume improvement such as (Hupkens, Emmerich, and Deutz 2014), it is usually a costly operation which significantly scales with the size of the Pareto set. To overcome this issue, a simpler, yet robust, algorithm is proposed by the present work. Here, each goal is modeled using Kriging then a state-of-the-art multiobjective algorithm (NSGA-II) is used to generate a Pareto set of the predicted mean of the surrogate models. From them, the design with higher value of predicted variance is chosen as an infill point.

## Surrogate Multiobjetive Approaches

In the current work, three different Kriging-based multiobjective frameworks are studied, which are discussed in the following subsections. The derivation of the Kriging predictor and the design of experiments (DOE) concept are not covered in this paper. The reader can find a comprehensive mathematical description of these subjects in (Forrester, Sobester, and Keane 2008). The Kriging models where built using the `R` package `DiceKriging` (O. Roustant, Ginsbourger, and Deville 2012).

### Multiobjective Efficient Global Optimization (MEGO)

EGO, proposed by Jones (Jones, Schonlau, and Welch 1998) for mono-objective optimization, consists in, from an initial set of samples $\mathbf{X}$, a Kriging model is built using the responses of a high-fidelity model, then the algorithm sequentially maximizes the expected improvement (EI) and updates the model at each iteration (including re-estimation of the hyperparameters).

The basic idea of the EI criterion is that by sampling a new point $\mathbf{x}^\star$ the results will be improved by $y_{\min} - y(\mathbf{x}^\star)$ if $y(\mathbf{x}^\star) < y_{\min}$ or 0 otherwise, where $y_{\min}$ is the lowest value of $\mathbf{y}$ so far. Obviously, the value of this improvement is not known in advance because $y(\mathbf{x}^\star)$ is unknown. However, the expectation of this improvement can be obtained using the information from the Kriging predictor. The EI criterion has important properties for sequential exploitation and exploration (filling) of the design space: it is null at points already visited (thus preventing searches in well-known regions and increasing the possibility of convergence); and at all other points it is positive and its magnitude increases with predicted variance (favoring searches in unexplored regions) and decreases with the predicted mean (favoring searches in regions with low predicted values).

The EGO algorithm can be easily imported to a multiobjective framework by creating a combined function of the qualities (Knowles 2006). The constrains of the optimization problem can be considered simply by building independent metamodels for each constraint and multiplying the EI of the composed objective function by the probability of each constraint to be met (Sasena, Papalambros, and Goovaerts 2002). The MEGO algorithm can be summarized as follows:

1. Generate an initial DOE $\mathbf{X}$ using an optimized *Latin hypercube*;
2. Evaluate $\mathbf{X}$ using high-fidelity models and store responses of $\mathbf{f} = [f_1, f_2, \ldots, f_m]^T$ and $\mathbf{g} = [g_1, g_2, \ldots, g_p]^T$ for the $m$-objectives and $p$-constraints.
    1. **while** computational budget not exhausted **do**:
    a. Normalize the responses to fit in a hypercube of size $[0, 1]^m$;
    b. For each $\mathbf{x} \in \mathbf{X}$ Compute a scalar quality by making $f_\lambda = \max_{i=1}^{m}(\lambda_i f_i) + \rho \sum_{(i=1)}^{m} \lambda_i f_i$, where $\lambda$ is drawn uniformly at random from a set of evenly distributed unit vectors and $\rho$ is an arbitrary small value which we set to 0.05;
    c. Build Kriging models for $f_\lambda$ and for the constraints $\mathbf{g}$;
    d. Find $\mathbf{x}^\star$ that maximizes the *constrained expected improvement*: $\mathbf{x}^\star = \arg(\max(\mathrm{EI}_C(\mathbf{x})))$;
    e. Evaluate the "true" values of $f(\mathbf{x}^\star)$ and $g(\mathbf{x}^\star)$ using high-fidelity models and update the database.
3. **end while**.

Here, the EI is computed using a custom modified version of the functions provided on the `R` package `DiceOptim` (D. Ginsbourger et al. 2013) so that it could handle constrains. Also, on all approaches, the optimized Latin hypercube is built using the `R` package `lhs` (Carnell 2012).

### Expected Hypervolume Improvement (EHVI)

For comparison, the expected hypervolume improvement (EHVI) is used as infill criterion. The EHVI is based on the theory of the hypervolume indicator (Zitzler and Thiele 1998), a metric of dominance of non-dominated solutions have. This metric consists in the size of the hypervolume fronted by the non-dominated set bounded by reference maximum points. I that sense, the EHVI is the expected improvement at the hypervolume size we would get by sampling a new point $\mathbf{x}^\star$. Here, the EHVI is computed using the R package `GPareto` (Binois and Picheny 2016). The EHVI function provided by this package do not account for constrains so a custom modification had to be implemented. The algorithm used here is similar to the MEGO and can be summarized as follows:

1. Generate an initial DOE $\mathbf{X}$ using an optimized *Latin hypercube*;
2. Evaluate $\mathbf{X}$ using high-fidelity models and store responses of $\mathbf{f} = [f_1, f_2, \ldots, f_m]^T$ and $\mathbf{g} = [g_1, g_2, \ldots, g_p]^T$ for the $m$-objectives and $p$-constraints.
3. **while** computational budget not exhausted **do**:

   a. Normalize the responses to fit a hypercube of size $[0, 1]^m$;
   b. For each of the $m$-objectives and $p$-constraints, build a Kriging model;
   c. Find $\mathbf{x}^\star$ that maximizes the *constrained expected hypervolume improvement*: $\mathbf{x}^\star = \arg(\max(\text{EHVI}_C(\mathbf{x})))$;
   d. Evaluate the "true" values of $f(\mathbf{x}^\star)$ and $g(\mathbf{x}^\star)$ using high-fidelity models and update the database.
4. **end while**.

For this and the previous approach, the algorithm used to maximize the infill criteria ($\text{EHVI}_C$ and $\text{EI}_C$, respectively) is the one provided by the `R` package `GenSA` (Yang Xiang et al. 2013) which stands for generalized simulated annealing.

#### Variance Minimization of the Kriging-predicted Front (VMKF)

The proposed framework, VMKF, is based on the iterative improvement of the predicted Pareto set fidelity. Here, the idea is, from a given initial set of Kriging models (one for each cost or constraint function), to build a Pareto front using the predictor's mean of each model as input functions. From the estimated front $\mathbf{P}$, the design with higher variance $\mathbf{x}^\star$ (i.e.: most isolated on the decision space) have it's "true" value evaluated using the high fidelity models. A new set of Kriging models are then rebuilt and the process repeats until a stopping criteria is met. The proposed algorithm can be summarized as follows:

1. Generate an initial DOE $\mathbf{X}$ using an optimized *Latin hypercube*;
2. Evaluate $\mathbf{X}$ using high-fidelity models and store responses of $\mathbf{f} = [f_1, f_2, \ldots, f_m]^T$ and $\mathbf{g} = [g_1, g_2, \ldots, g_p]^T$ for the $m$-objectives and $p$-constraints.
3. **while** computational budget not exhausted **do**:
   a. For each of the $m$-objectives and $p$-constraints, build a Kriging model;
   b. Generate a Pareto set $\mathbf{P}$ using the mean predictor of the Kriging models using a state-of-art multiobjective optimization algorithm (such as NSGA-II);
   c. Find $\mathbf{x}^\star \in \mathbf{P}$ that maximizes the *variance of the Kriging predictor*: $\mathbf{x}^\star = \arg(\max(\text{s}_{\text{km}}(\mathbf{x})))$;
   d. Evaluate the "true" values of $f(\mathbf{x}^\star)$ and $g(\mathbf{x}^\star)$ using high-fidelity models and update the database.
4. **end while**

Here, the NSGA-II implementation used is the one provided by the `R` package `mco` (Mersmann 2014).

## The Nowacki Beam

However Kriging-based optimization is more useful for costly black box optimization problems, here we will demonstrate the technique using an analytic function for didactic proposes.

in the well known Nowacki beam optimization problem (Nowacki 1980), the aim is to design a tip loaded cantilever beam for minimum cross-sectional area and bending stress. The beam length is $l = 1500$ mm and at is subject to a tip load force of $F = 5000$ N. The cross-section of the beam is rectangular, with breadth $b$ and height $h$, which are the design variables. The design is constrained by 5 requisites and the optimization problem can be formally defined as the following:

$$
\begin{aligned}
\text{find: } & \{b, h\}, \\
\text{where: } & \{20 \leq h \leq 250\}, \\
\text{and: } & \{10 \leq b \leq 50\}, \\
\text{to minimize A: } & A = b\,h \\
\text{and minimize B: } & \sigma = \frac{6Fl}{b^2 h}, \\
\text{subject to 1: } & \delta = \frac{12Fl^3}{Ebh^3} \leq 5, \\
\text{subject to 2: } & \sigma = \frac{6Fl}{b^2 h} \leq 240, \\
\text{subject to 3: } & \tau = \frac{3F}{2bh} \leq 120, \\
\text{subject to 4: } & \text{AR} = \frac{h}{b} \leq 10, \\
\text{subject to 5: } & F_{\text{crit}} = -\frac{4}{l^2}\sqrt{G\frac{(b^3h + hb^3)}{12}E\frac{b^3h}{12}\frac{1}{(1-v^2)}} \leq -2F.
\end{aligned}
$$

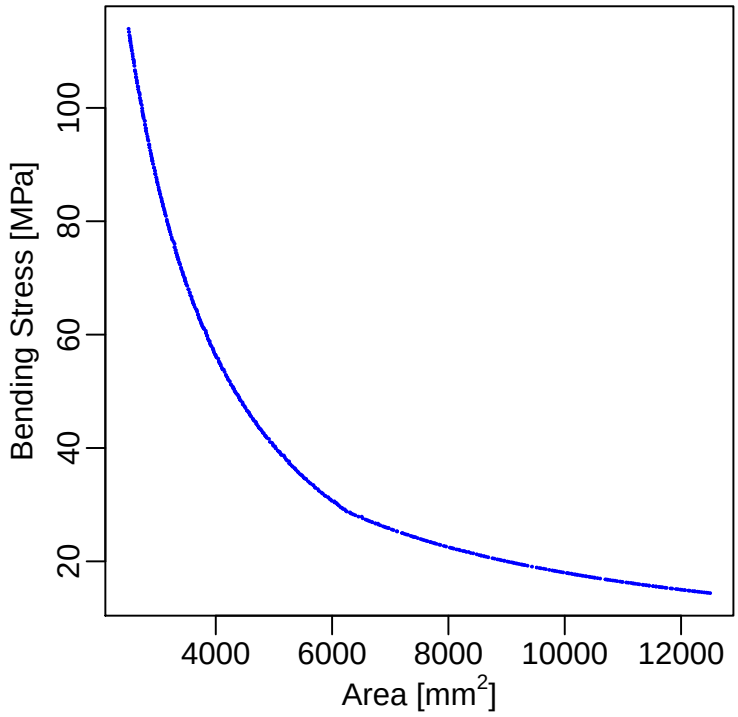
The material used on the original problem is a mild steel with a yield stress of $\sigma_Y = 240$MPa, Young's modulus $E = 216.62$GPa, Poisson ratio $\nu = 0.27$ and shear modulus calculated as $G = 86.65$GPa. For consistency, all values are physically interpreted on the unit system [mm, N, MPa].

## Quality Metric

Here, the quality of the Pareto sets found are compared using the inverted generational distance (IGD) metric (Shimoyama, Jeong, and Obayashi 2013). The IGD can be defined as

$$
\text{IGD}(\mathbf{T}, \mathbf{P}) = \frac{1}{|\mathbf{T}|}\sum_{\mathbf{t}\in\mathbf{T}} \min(d(\mathbf{t} - \mathbf{p}))_{\mathbf{p}\in\mathbf{P}},
$$

where $\mathbf{T}$ and $\mathbf{P}$ are the true and the current Pareto sets, $|\mathbf{T}|$ is the number of designs in the true Pareto set and $\mathbf{t}$ and $\mathbf{p}$ are normalized vectors of length $m$ of the $m$-objectives of the true and the actual Pareto sets, respectively, and $d(\quad)$ is a distance metric that here is the Manhattan's. Hence, IGD corresponds to the average distance between all designs in the true set and the closest design of the current set. Thus, the lower the IGD value, the better the method is. For the validation case, the "true" Pareto front (Fig. 1) is obtained by direct optimization using the NSGA-II algorithm using a population size of 500 and 100 generations, resulting in a near-uniform Pareto set of $|\mathbf{T}| = 500$.




## Methodology

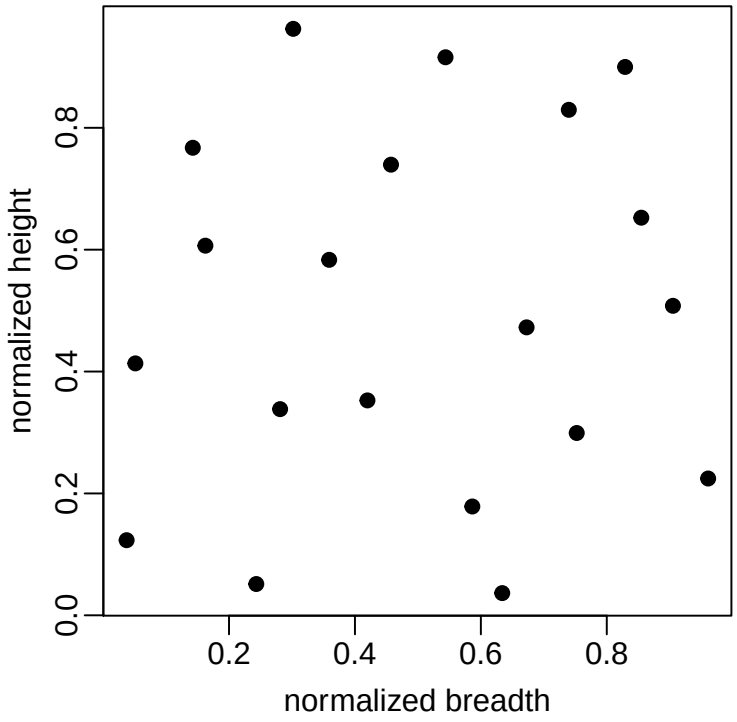
First we load the `moko` package and the `lhs` package that we will use here for optimal DOE generation.

```
library(moko)
library(lhs)
```

After loading the necessary packages, we generate an initial DOE using an optimized Latin hypercube of `n = 20` samples in two dimensions (`d = 2`) by doing:

```
n = 20
d = 2
set.seed(18)
doe <- optimumLHS(n,d)
```

The `seed` is arbitrary set to 100 so we can achieve reproducibility. This is how our sample looks:




Now, we load the Nowacki beam function and compute the output.

```
fun <- nowacki_beam
res <- t(apply(doe, 1, fun))
```

The `res` object consists in a numeric matrix with 20 lines and 7 columns:

```
##           [,1]     [,2]    [,3]     [,4]     [,5]   [,6]         [,7]
##  [1,] 7516.075   35.203  -3.567 -204.797 -119.002 -6.152 -1220372.063
##  [2,]  627.667 2253.893 485.655 2013.893 -108.051 -8.388      227.153
##  [3,] 3469.892  181.044  12.501  -58.956 -117.839 -8.521  -296387.036
##  [4,] 2631.625  107.202  -0.346 -132.798 -117.150 -0.332  -136765.942
##  [5,] 1003.562 1579.816 380.423 1339.816 -112.527 -9.197   -23914.067
##  [6,] 1386.874  281.947  11.965   41.947 -114.592 -0.450   -30766.970
##  [7,] 5377.703   44.022  -3.396 -195.978 -118.605 -3.281  -606336.425
##  [8,] 4749.309   73.629  -1.038 -166.371 -118.421 -6.513  -484642.232
##  [9,] 2709.246  164.276   6.251  -75.724 -117.232 -6.227  -150067.191
## [10,] 5329.015   34.989  -3.996 -205.011 -118.593  0.930  -591132.666
## [11,] 3756.095   77.710  -1.510 -162.290 -118.003 -3.672  -291089.815
## [12,] 9796.517   20.239  -4.383 -219.761 -119.234 -4.742 -2049326.860
## [13,] 2078.114  221.387  10.673  -18.613 -116.391 -5.396   -83157.145
## [14,] 7323.116   26.646  -4.200 -213.354 -118.976 -2.738 -1131134.696
## [15,] 3560.476  142.310   6.096  -97.690 -117.894 -7.785  -283193.475
## [16,] 3084.387   74.261  -2.383 -165.739 -117.568  2.514  -191180.808
## [17,]  555.750 1674.726 234.854 1434.726 -106.505 -5.794     3307.882
## [18,] 6321.283   52.033  -2.366 -187.967 -118.814 -7.039  -879055.382
## [19,] 2043.343  360.602  35.886  120.602 -116.330 -8.175   -90355.853
## [20,] 8344.867   25.582  -4.160 -214.418 -119.101 -4.675 -1483593.992
```

Each line of this matrix is a single design where the two first columns are the outputs that we need to maximize and the remaining columns are the constraints values. Any value that is grater than zero does not

meet the constraint so the design is unfeasible. Note that, on this case, only the samples number 1, 4, 7, 8, 11, 12, 14, 18 and 20 are feasible. I does not matter right now, but we will check that latter, after fitting the model.

Now, we can create a multi-objective kriging model by calling the function `mkm`. Note that we need to setup the `modelcontrol` argument in order to tell the function that our data have two objectives. By doing that, the remaining columns of the response will be flagged as constraints. Also, in order to increase stability, we set the lower bounds for the kriging hyperparameter estimation as `0.1` for all variables (for more information check the *Identifiability issues caused by large design interdistances* section of (O. Roustant, Ginsbourger, and Deville 2012))

```
model <- mkm(doe, res, modelcontrol = list(objective = 1:2, lower = rep(0.1, d)))
```

The `model`is an `S4` objects with some usefull slots. For example, one can check which designs are feasible by simply calling:

```
which(model@feasible)
```

```
## [1]  1  4  7  8 11 12 14 18 20
```

which returns the index of the feasible designs. Furthermore, one can get the feasible designs themselves by calling:

```
model@design[model@feasible,]
```

```
##          x.1       x.2
## 1  0.8548210 0.6524978
## 4  0.1624574 0.6065604
## 7  0.4572788 0.7394968
## 8  0.6726555 0.4725473
## 11 0.3590855 0.5833455
## 12 0.8290982 0.8998293
## 14 0.5438668 0.9157191
## 18 0.9050853 0.5078878
## 20 0.7397084 0.8295263
```

or the responses associated with those feasible designs with:

```
model@response[model@feasible,]
```

```
##         y.1       y.2        y.3       y.4       y.5        y.6        y.7
## 1  7516.075  35.20321 -3.5667043 -204.7968 -119.0021 -6.1515376 -1220372.1
## 4  2631.625 107.20218 -0.3461603 -132.7978 -117.1501 -0.3317974  -136765.9
## 7  5377.703  44.02198 -3.3963261 -195.9780 -118.6054 -3.2811412  -606336.4
## 8  4749.309  73.62939 -1.0380129 -166.3706 -118.4208 -6.5131656  -484642.2
## 11 3756.095  77.71011 -1.5096264 -162.2899 -118.0032 -3.6720934  -291089.8
## 12 9796.517  20.23905 -4.3825070 -219.7610 -119.2344 -4.7418885 -2049326.9
## 14 7323.116  26.64577 -4.1999213 -213.3542 -118.9758 -2.7375915 -1131134.7
## 18 6321.283  52.03267 -2.3664740 -187.9673 -118.8135 -7.0388722  -879055.4
## 20 8344.867  25.58238 -4.1596088 -214.4176 -119.1012 -4.6754254 -1483594.0
```

One can even filter only the feasible designs objective's by:

```
model@response[model@feasible,model@objective]
```

```
##         y.1       y.2
## 1  7516.075  35.20321
## 4  2631.625 107.20218
## 7  5377.703  44.02198
## 8  4749.309  73.62939
```

```
## 11 3756.095  77.71011
## 12 9796.517  20.23905
## 14 7323.116  26.64577
## 18 6321.283  52.03267
## 20 8344.867  25.58238
```

This is only a small number of operations that can be done by using the slots of the `mkm` model. More details on the slots can be found on the help using `?'mkm-class'`.

Now that we executed steps 1 and 2 for all optimization techniques presented here, we will apply the VMPF algorithm on the initial model to demonstrate how to handle the `mkm` object. Considering a total budget of 40 evaluations (which 20 were already spent building the initial model) we can code the technique as follows:

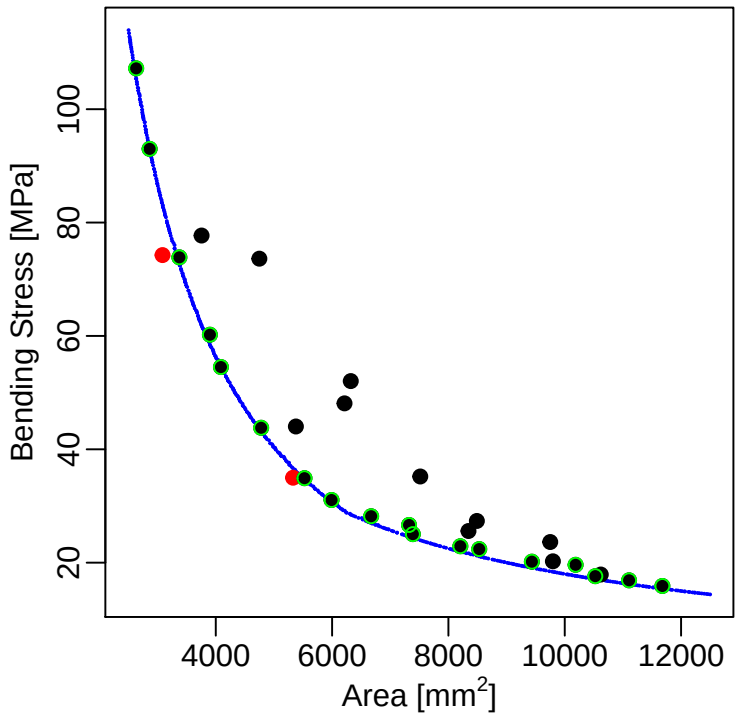
```
for (i in 21:40){
  pred_ps <- predict_front(model, lower = rep(0,d), upper = rep(1,d))
  pred_ps$sd <- predict(model, pred_ps$x)$norm_sd
  x_star <- pred_ps$x[which.max(pred_ps$sd),]
  y_star <- fun(x_star)
  model <- mkm(
    rbind(model@design, x_star),
    rbind(model@response, y_star),
    modelcontrol = model@control)
}
```

To check the IGD metric we first need to build a `ps` object from the actual data.

```
actual_ps <- ps(model@response[model@feasible,model@objective])
print(igd(actual_ps, true_ps))
```

```
## [1] 0.03799367
```

Now we can visualize the actual Pareto front and check how good it is against the true front.



Alternatively, one can use the `VMPF` function and save some coding lines. This function is basically a wrapper for the demonstrated algorithm, it receives a `mkm` model as input and returns the updated model after `niter` iterations. There are also wrappers for the other two algorithms that could be used as follows:

```
model <- mkm(doe, res, modelcontrol = list(objective = 1:2, lower = rep(0.1, d)))
niter <- 20
model.MEGO <- MEGO(model, fun, niter)
model.HEGO <- HEGO(model, fun, niter)
model.VMPF <- VMPF(model, fun, niter)
```

# References


Binois, Mickael, and Victor Picheny. 2016. *GPareto: Gaussian Processes for Pareto Front Estimation and Optimization*. https://cran.r-project.org//package=GPareto.

Carnell, Rob. 2012. *LHS: Latin Hypercube Samples*. https://cran.r-project.org//package=lhs.

Emmerich, Michael, André H Deutz, and Jan Willem Klinkenberg. 2011. "Hypervolume-Based Expected Improvement: Monotonicity Properties and Exact Computation." In *2011 Ieee Congress of Evolutionary Computation (Cec)*, 2147–54. IEEE.

Forrester, Alexander, and Andy J Keane. 2009. "Recent Advances in Surrogate-Based Optimization." *Progress in Aerospace Sciences* 45 (1). Elsevier: 50–79.

Forrester, Alexander, Andras Sobester, and Andy Keane. 2008. *Engineering Design via Surrogate Modelling: A Practical Guide*. Pondicherry, India: John Wiley & Sons.

Ginsbourger, D., V. Picheny, O. Roustant, with contributions by C. Chevalier, and T. Wagner. 2013. *DiceOptim: Kriging-Based Optimization for Computer Experiments*. https://cran.r-project.org//package=DiceOptim.

Hupkens, Iris, Michael Emmerich, and André Deutz. 2014. "Faster Computation of Expected Hypervolume Improvement." *ArXiv Preprint ArXiv:1408.7114*. LIACS.

Jones, Donald R, Matthias Schonlau, and William J Welch. 1998. "Efficient Global Optimization of Expensive Black-Box Functions." *Journal of Global Optimization* 13 (4): 455–92.

Knowles, Joshua. 2006. "ParEGO: A Hybrid Algorithm with on-Line Landscape Approximation for Expensive Multiobjective Optimization Problems." *Evolutionary Computation, IEEE Transactions on* 10 (1). IEEE: 50–66.

Li, Mian. 2011. "An Improved Kriging-Assisted Multi-Objective Genetic Algorithm." *Journal of Mechanical Design* 133 (7). American Society of Mechanical Engineers: 071008.

Mersmann, Olaf. 2014. *MCO: Multiple Criteria Optimization Algorithms and Related Functions*. https://cran.r-project.org//package=mco.

Nowacki, Horst. 1980. "Modelling of Design Decisions for Cad." In *Computer Aided Design Modelling, Systems Engineering, Cad-Systems*, 177–223. Springer.

Roustant, Olivier, David Ginsbourger, and Yves Deville. 2012. "DiceKriging, DiceOptim: Two R Packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization." *Journal of Statistical Software* 51 (1): 1–55.

Sasena, Michael J, Panos Papalambros, and Pierre Goovaerts. 2002. "Exploration of Metamodeling Sampling Criteria for Constrained Global Optimization." *Engineering Optimization* 34 (3). Taylor & Francis: 263–78.

Shimoyama, Koji, Shinkyu Jeong, and Shigeru Obayashi. 2013. "Kriging-Surrogate-Based Optimization Considering Expected Hypervolume Improvement in Non-Constrained Many-Objective Test Problems." In

*Evolutionary Computation (Cec), 2013 Ieee Congress on*, 658–65. IEEE.

Yang Xiang, Sylvain Gubian, Brian Suomela, and Julia Hoeng. 2013. "Generalized Simulated Annealing for Efficient Global Optimization: The GenSA Package for R." *The R Journal Volume 5/1, June 2013*. http://journal.r-project.org/.

Zitzler, Eckart, and Lothar Thiele. 1998. "Multiobjective Optimization Using Evolutionary Algorithms—a Comparative Case Study." In *Parallel Problem Solving from Nature—PPSN V*, 292–301. Springer.

# APÊNDICE C – ARTIGO: KRIGING-BASED MULTIOBJECTIVE OPTIMIZATION OF A FUSELAGE-LIKE COMPOSITE SECTION WITH CURVILINEAR FIBERS

**EngOpt 2016 - $5^{th}$ International Conference on Engineering Optimization**
Iguassu Falls, Brazil, 19-23 June 2016.

# Kriging-Based Multiobjective Optimization of a Fuselage-Like Composite Section with Curvilinear Fibers

**Adriano G. Passos**[1] **Marco A. Luersen**[1]

[1]Mechanical Engineering Department, Federal University of Technology – Parana, Curitiba – PR, Brazil

**Abstract**

This paper presents the multiobjective optimization of an aircraft fuselage-like composite section with curvilinear fiber paths. The structure studied is a reinforced composite panel with an oblong cutout. The skin of the fuselage section, which is the part to be optimized, is a 24-ply symmetric and balanced laminate and the fiber paths are parameterized using two parameters per ply, totalizing 12 continuous design variables in the [0, 90] degree space. The objectives examined are: the first natural frequency (to be maximized), the maximum displacement (to be minimized) and the reserve factor of the Tsai-Wu failure criterion (to be maximized). Also, the buckling under shearing load is imposed as a constraint in the optimization problem. To overcome the problem of long computational run time, Kriging-based approaches are used here. Three different frameworks for dealing with many-objective problems using Kriging surrogate models are compared: (i) the efficient global optimization (EGO) algorithm applied to a single objective function composed by the combined objectives responses (MEGO), (ii) the iterative maximization of the expected hypervolume improvement (EHVI), and (iii), a novel approach proposed here, the variance minimization of the Kriging predicted Pareto front (VMKF). To evaluate the efficiency of these three methods, a baseline solution is created using multiobjective direct optimization (no surrogates are used) applying the well-known NSGA-II algorithm. Finally, the results are compared and discussed, showing the computational burden reduction and the efficiency of VMKF technique.

**Keywords:** Multiobjective Optimization; Kriging; Variable Stiffness Composites.

## 1 Introduction

Multiobjective optimization is a field of interest for many real-world applications. Usually, projects have multiple and conflicting goals and, most of the time, the relationship between the decision space (design variables) and the outcome is highly complex. Around the 2000's, some comprehensive reviews were made that cover many of the basic ideas and challenges related to many-objective optimization [1–3]. Today, there are many efficient algorithms for dealing with multiobjective optimization, most of them being based on evolutionary techniques (evolutionary multiobjective optimization algorithm – EMOA). Some state-of-the-art algorithms can be found in [4–10]. However, even when using such efficient frameworks, if the cost of evaluation of the designs are too high, a surrogate approach must be used to alleviate the computational burden.

In the past years, Kriging have become one of the most popular surrogates on the industry [11]. When using Kriging, usually the efficient global optimization (EGO) algorithm is the standard technique for single objective optimization. For costly multiple objectives, direct combination of the Kriging predictions and a multiobjective genetic algorithm (MOGA) can be used such as in [12]. However, according to [11, 13], there are currently two popular ways of constructing Pareto sets. In the first approach, all goals are combined into a single quantity, then the EGO algorithm is performed. The weighting function have adjustable parameters that change during the optimization problem so that the algorithm can potentially explore all the objectives space. Here, this approach is called MEGO. Another popular way to address many-objective problems is to generalize the expected improvement (EI) criterion into what is called the expected hypervolume improvement (EHVI) [14, 15]. Although there are some efficient algorithms to calculate and/or estimate the expected hypervolume improvement such as [16], it is usually a costly operation which increases significantly with the size of the Pareto set. To overcome this issue, a simple, yet robust, algorithm is proposed by the present work. Here, each goal is modeled using Kriging then a state-of-the-art multiobjective algorithm (NSGA-II) is used to generate a Pareto set of the predicted mean of the surrogate models. From them, the design with higher value of predicted variance is chosen as an infill point.

# APÊNDICE D – ARTIGO: MOKO: AN OPEN SOURCE PACKAGE FOR MULTI-OBJECTIVE OPTIMIZATION WITH KRIGING SURROGATES

MECSOL Joinville 2017
April 26-28, 2017
Joinville, Santa Catarina, Brazil

# MOKO: An Open Source Package for Multi-Objective Optimization with Kriging Surrogates

**Adriano Gonçalves dos Passos**
Mechanical Engineering Department
Federal University of Technology - Parana, Curitiba, Parana, Brazil
adriano.utfpr@gmail.com

**Marco Antônio Luersen**
Mechanical Engineering Department
Federal University of Technology - Parana, Curitiba, Parana, Brazil
luersen@utfpr.edu.br

## ABSTRACT

Many modern real-world designs rely on the optimization of multiple competing goals. For example, most components designed for the aerospace industry must meet some conflicting expectations. In such applications, low weight, low cost, high reliability, and easy manufacturability, are desirable. In some cases, bounds for these requirements are not clear, and performing a mono-objective constrained optimization might not provide a good landscape of optimal choices. For these cases, finding a set of Pareto optimal designs might give the designer a comprehensive set of options from where to choose the best design. This article shows the main features of an open source package, developed by the authors, to solve constrained multi-objective problems. The package, named `moko` (Multi-Objective Kriging Optimization), was built under the open source programming language R. Popular Kriging based multi-objective optimization strategies, as the expected volume improvement and the weighted expected improvement, are available in the package. In addition, a novel approach based on the exploration using a predicted Pareto front is implemented. The latter approach showed to be more efficient than the remainder ones in some didactic and real-life multi-objective applications performed by the authors with `moko`.

**Keywords:** Multi-Objective Optimization, Surrogates, Kriging, Open Source Package

## 1. INTRODUCTION

Multi-objective optimization is a field of interest for many real-world applications. Usually, projects have multiple and conflicting goals and, most of the time, the relationship between the decision space (design variables) and the outcome is highly complex. Around the 2000's, some comprehensive reviews were made that cover many of the basic ideas and challenges related to many-objective optimization [1–3]. Today, there are many efficient algorithms for dealing with multi-objective optimization, most of them being based on evolutionary techniques (evolutionary multi-objective optimization algorithm – EMOA). Some state-of-the-art algorithms can be found in [4–10]. However, even when using such efficient frameworks, if the cost of evaluation of the designs are too high, a surrogate approach is usually used to alleviate the computational burden.

To address this issue, the authors have developed an open-source package based on the Kriging surrogate model. The package, written in R language, presents three optimization algorithms. Two

# APÊNDICE E – ARTIGO: MULTIOBJECTIVE OPTIMIZATION OF LAMINATED COMPOSITE PARTS WITH CURVILINEAR FIBERS USING KRIGING-BASED APPROACHES

Struct Multidisc Optim
DOI 10.1007/s00158-017-1800-7

RESEARCH PAPER

# Multiobjective optimization of laminated composite parts with curvilinear fibers using Kriging-based approaches

**A. G. Passos**[1] · **M. A. Luersen**[1]

Received: 10 April 2017 / Revised: 21 June 2017 / Accepted: 28 August 2017


**Abstract** This paper describes the multiobjective optimization of parts made with curvilinear fiber composites. Two structures are studied: a square plate and a fuselage-like section. The square plate is designed in two ways. First, classical lamination theory (CLT) is used to obtain the structural response for a plate with straight fibers designed for maximum buckling load and maximum stiffness. The same plate is then designed with curved fibers using finite element analysis (FEA) to determine the structural response. Next, the fuselage-like section is designed using the same FEA approach. The problems have three to twelve variables. To enable the resulting Pareto front to be visualized more clearly, only two objectives are considered. The first two optimization problems are unconstrained, while the last one is constrained by two project requirements. To overcome the problem of long computational run time when using FEA, Kriging-based approaches are used. Three such approaches suitable for multiobjective problems are compared: (i) the efficient global optimization algorithm (EGO) is applied to a single-objective function consisting of a weighted combination of the objectives, (ii) a technique that involves sequential maximization of the expected hypervolume improvement, and (iii) a novel approach proposed here based on sequential minimization of the variance of the predicted Pareto front. Comparison of the results using the inverted generational distance (IGD) metric revealed that the approach (iii) had the best performance (mean) and best robustness (standard deviation) for all the cases studied.

✉ A. G. Passos
adriano.utfpr@gmail.com

[1] Department of Mechanical Engineering, Federal University of Technology, Paraná, Curitiba, PR, Brazil

**Keywords** Multiobjective optimization · Kriging · Curvilinear fiber composites

## 1 Introduction

Multiobjective optimization has been applied in many fields of science where optimal decisions need to be taken in the presence of trade-offs between two or more conflicting objectives. Usually, a single solution that simultaneously optimizes each objective does not exist. In such cases, the objectives are said to be conflicting, and there is a set of solutions called Pareto optimal solutions that consists of all the non-dominated solutions, i.e. those that cannot be improved in any way without degrading at least one of the objectives. Some comprehensive reviews covering the basic concepts involved in multiobjective optimization and the challenges it poses were published at the end of the last century and beginning of this one (Van Veldhuizen and Lamont 1998; Zitzler 1999; Deb et al. 2002a), since when many efficient algorithms have been proposed. Some of these can be found in (Corne et al. 2001; Zitzler et al. 2001; Deb et al. 2002b; Emmerich et al. 2005; Beume et al. 2007; Deb 2014; Chen et al. 2015). However, since most are based on evolutionary techniques (evolutionary multiobjective optimization algorithms – EMOAs), a surrogate approach is usually adopted to speed up the optimization process if evaluation of the designs is too time consuming.

A comprehensive review of surrogate-based multiobjective optimization can be found in (Tabatabaei et al. 2015). It covers topics ranging from basic definitions of multiobjective optimization to the construction of basic surrogates and compares and summarizes over twenty papers published before 2013. The authors classify surrogate-based methods in two large groups according to when the models are

Published online: 08 September 2017

# ANEXO A – GUIA DE REFERÊNCIA DO PROCESSO DE CRIAÇÃO DE UM PACOTE

As páginas a seguir apresentam um guia rápido de referência do processo de criação de pacotes obtido na página `https://www.rstudio.com/wp-content/uploads/2015/03/devtools-cheatsheet.pdf` em Agosto de 2016.

# Package Development

**with devtools** Cheat Sheet

RStudio

## Package Structure

A package is a convention for organizing files into directories.

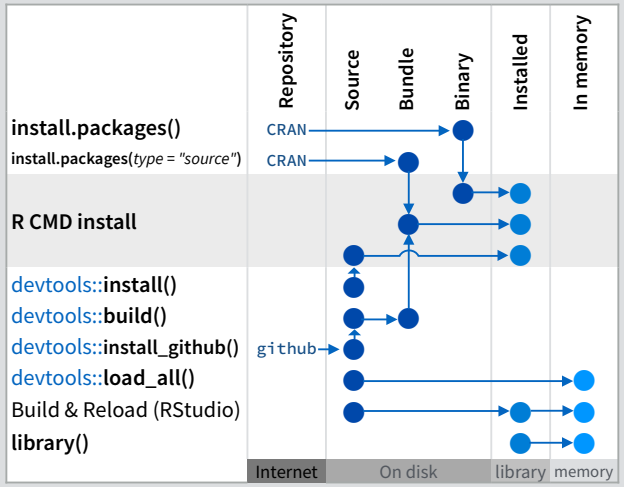
This sheet shows how to work with the 7 most common parts of an R package:

| | |
|---|---|
| Package | |
| DESCRIPTION | Setup |
| R/ | Write code |
| tests/ | Test |
| man/ | Document |
| vignettes/ | Teach |
| data/ | Add data |
| NAMESPACE | Organize |

The contents of a package can be stored on disk as a:

- **source** - a directory with sub-directories (as above)
- **bundle** - a single compressed file (*.tar.gz*)
- **binary** - a single compressed file optimized for a specific OS

Or installed into an R library (loaded into memory during an R session) or archived online in a repository. Use the functions below to move between these states.




devtools::**add_build_ignore**(*"file"*)

Adds file to .Rbuildignore, a list of files that will not be included when package is built.

## Setup ( DESCRIPTION)

The DESCRIPTION file describes your work and sets up how your package will work with other packages.

- You must have a DESCRIPTION file
- Add the packages that yours relies on with

devtools::**use_package()**

Adds a package to the Imports field (or Suggests field (if second argument is "Suggests").

| CC0 | MIT | GPL-2 |
|---|---|---|
| No strings attached. | MIT license applies to your code if re-shared. | GPL-2 license applies to your code, *and all code anyone bundles with it*, if re-shared. |

```
Package: mypackage
Title: Title of Package
Version: 0.1.0
Authors@R: person("Hadley", "Wickham", email =
    "hadley@me.com", role = c("aut", "cre"))
Description: What the package does (one paragraph)
Depends: R (>= 3.1.0)
License: GPL-2
LazyData: true
Imports:
    dplyr (>= 0.4.0),
    ggvis (>= 0.2)
Suggests:
    knitr (>= 0.1.0)
```

**Import** packages that your package *must have* to work. R will install them when it installs your package.

**Suggest** packages that are not very essential to yours. Users can install them manually, or not, as they like.

## Write code ( R/)

All of the R code in your package goes in R/. A package with just an R/ directory is still a very useful package.

- Create a new package project with

devtools::**create**(*"path/to/name"*)

Create a template to develop into a package.

- Save your code in R/ as scripts (extension .R)

### Workflow

1. Edit your code.
2. Load your code with one of

devtools::**load_all()**

Re-loads all *saved* files in R/ into memory.

**Ctrl/Cmd + Shift + L** (keyboard shortcut)

Saves all open files then calls load_all().

3. Experiment in the console.
4. Repeat.

- Use consistent style with **r-pkgs.had.co.nz/r.html#style**
- Click on a function and press **F2** to open its definition
- Search for a function with **Ctrl + .**

## Visit r-pkgs.had.co.nz for more



## Test ( tests/)

Use tests/ to store unit tests that will inform you if your code ever breaks.

- Add a **tests/** directory and import **testthat** with

devtools::**use_testthat()**

Sets up package to use automated tests with testthat

- Write tests with **context()**, **test()**, and expectations
- Save your tests as .R files in **tests/testthat/**

### Workflow

1. Modify your code or tests.
2. Test your code with one of

devtools::**test()**

Runs all tests *saved* in tests/.

**Ctrl/Cmd + Shift + T**

(keyboard shortcut)

3. Repeat until all tests pass

### Example test

```
context("Arithmetic")

test_that("Math works", {
  expect_equal(1 + 1, 2)
  expect_equal(1 + 2, 3)
  expect_equal(1 + 3, 4)
})
```

| | |
|---|---|
| expect_equal() | is equal within small numerical tolerance? |
| expect_identical() | is exactly equal? |
| expect_match() | matches specified string or regular expression? |
| expect_output() | prints specified output? |
| expect_message() | displays specified message? |
| expect_warning() | displays specified warning? |
| expect_error() | throws specified error? |
| expect_is() | output inherits from certain class? |
| expect_false() | returns FALSE? |
| expect_true() | returns TRUE? |

# Document ( man/)

man/ contains the documentation for your functions, the help pages in your package.

- Use roxygen comments to document each function beside its definition
- Document the name of each exported data set
- Include helpful examples for each function

## Workflow

1. Add roxygen comments in your .R files
2. Convert roxygen comments into documentation with one of

   devtools::**document()**
   Converts roxygen comments to .Rd files and places them in man/. Builds NAMESPACE.

   **Ctrl/Cmd + Shift + D** (Keyboard Shortcut)
3. Open help pages with **?** to preview documentation
4. Repeat

## .Rd formatting tags

```
\emph{italic text}
\strong{bold text}
\code{function(args)}
\pkg{package}

\dontrun{code}
\dontshow{code}
\donttest{code}

\deqn{a + b (block)}
\eqn{a + b (inline)}

\email{name@@foo.com}
\href{url}{display}
\url{url}

\link[=dest]{display}
\linkS4class{class}
\code{\link{function}}
\code{\link[package]{function}}

\tabular{lcr}{
  left \tab centered \tab right \cr
  cell \tab cell     \tab cell  \cr
}
```

## The roxygen package

**roxygen** lets you write documentation inline in your .R files with a shorthand syntax.

- Add roxygen documentation as comment lines that begin with **#'**.
- Place comment lines directly above the code that defines the object documented.
- Place a roxygen **@** tag (right) after **#'** to supply a specific section of documentation.
- Untagged lines will be used to generate a title, description, and details section (in that order)

```
#' Add together two numbers.
#'
#' @param x A number.
#' @param y A number.
#' @return The sum of \code{x} and \code{y}.
#' @examples
#' add(1, 1)
#' @export
add <- function(x, y) {
  x + y
}
```

## Common roxygen tags

| | | |
|---|---|---|
| @aliases | @inheritParams | **@seealso** |
| @concepts | @keywords | @format |
| @describeIn | **@param** | @source (data) |
| **@examples** | @rdname | @include |
| **@export** | **@return** | @slot (S4) |
| @family | @section | @field (RC) |

# Teach ( vignettes/)

vignettes/ holds documents that teach your users how to solve real problems with your tools.

- Create a vignettes/ directory and a template vignette with

  devtools::**use_vignette()**
  Adds template vignette as vignettes/my-vignette.Rmd.
- Append YAML headers to your vignettes (like right)
- Write the body of your vignettes in R Markdown (rmarkdown.rstudio.com)

```
---
title: "Vignette Title"
author: "Vignette Author"
date: "`r Sys.Date()`"
output: rmarkdown::html_vignette
vignette: >
  %\VignetteIndexEntry{Vignette Title}
  %\VignetteEngine{knitr::rmarkdown}
  \usepackage[utf8]{inputenc}
---
```

# Add data ( data/)

The data/ directory allows you to include data with your package.

- Store data in one of **data/**, **R/Sysdata.rda**, **inst/extdata**
- Always use **LazyData: true** in your DESCRIPTION file.
- Save data as .Rdata files (suggested)

devtools::**use_data()**
Adds a data object to data/
(R/Sysdata.rda if **internal = TRUE**)

devtools::**use_data_raw()**
Adds an R Script used to clean a data set to data-raw/. Includes data-raw/ on .Rbuildignore.

Store data in

- **data/** to make data available to package users
- **R/sysdata.rda** to keep data internal for use by your functions.
- **inst/extdata** to make raw data available for loading and parsing examples. Access this data with **system.file()**

# Organize ( NAMESPACE)

The NAMESPACE file helps you make your package self-contained: it won't interfere with other packages, and other packages won't interfere with it.

- Export functions for users by placing **@export** in their roxygen comments
- Import objects from other packages with **package::object** (recommended) or **@import**, **@importFrom**, **@importClassesFrom**, **@importMethodsFrom** (not always recommended)

## Workflow

1. Modify your code or tests.
2. Document your package (devtools::**document()**)
3. Check NAMESPACE
4. Repeat until NAMESPACE is correct

# Submit your package

r-pkgs.had.co.nz/release.html