Trabalho final de AGE e OSD

R. Vieira*

20 de Dezembro de 2020

Resumo

O relatório apresenta o resultado da otimização de uma chapa em alumínio com dois objetivos e três parâmetros. Essa chapa é um elemento essencial de um sistema de teste para placas de circuito impresso sendo imperioso otimiza-la quanto ao peso e rigidez. Utilizou-se uma equação presente na literatura para modelar a deformação da chapa em funcionamento. Implementou-se um algoritmo genético NSGA-II utilizando uma livraria *open source Pymoo*. Este convergiu sem violação de restrições para uma frente de Pareto. Aplicando pesos concluiu-se que a solução ideal é encontrada para a deformação $0.0023m$ e massa $0.0023kg$ com os parametros $a = 0.2500m$ $b = 0.2500m$ e $h = 0.0017m$.

1 Introdução

As placas de circuito impresso são essenciais numa era de crescente digitalização. É essencial garantir que não existem defeitos de produção para que operem normalmente na vida do componente. Para isso fazem-se testes com um sistema que mede tensões e correntes no circuito. Devido ao facto de os circuitos serem da ordem dos 70 µm é essencial que este sistema de teste tenha uma alta rigidez (pouca deformação em operação) e de baixo peso (para que os operadores troquem o sistema de teste facilmente).

O elemento mais importante desse sistema é uma chapa em Alumínio cujo comportamento mecânico pode ser modelado matematicamente. Assim, as funções objetivo (a minimizar) são:

$$w_0 = \sum_{n=1}^{\infty}\sum_{m=1}^{\infty} W_{mn} sin\left(\frac{m\pi x}{a}\right) sin\left(\frac{n\pi y}{b}\right) \tag{1}$$

$$Mass = abh\rho_{material} \tag{2}$$

*Departamento de Engenharia Mecânica, Universidade do Minho, `ae5333@alunos.uminho.pt`

Ou seja, a massa do sistema é modelado na equação 2 e os deslocamentos na equação 1 [?]. Sabendo que:

$$D_{const} = \frac{Eh^3}{12\left(1-\nu^2\right)} \tag{3}$$

$$k = \frac{kb^4}{D_{const}\pi^4} \tag{4}$$

$$\delta T = \frac{T\alpha D_{const}\left(1+\nu\right)\pi^2}{b^2} \tag{5}$$

$$W_{mn} = \frac{\frac{b^4}{D_{const}\pi^4}\left(q_{mn} + \delta T\left(m^2s^2+n^2\right)\right)}{\left(m^2s^2+n^2\right)^2 + k} \tag{6}$$

Para além disso, existem restrições técnicas sobre os três parâmetros geométricos. O comprimento e largura da chapa são limitados por limitação de espaço. O problema pode ser formalizado como:

$$\begin{aligned} \min \quad & w_0(a,b,h) \\ \min \quad & Mass(a,b,h) \\ s.a \quad & 0.25 \leq a \leq 0.5 \\ & 0.25 \leq b \leq 0.5 \\ & 0.001 \leq h \leq 0.01 \end{aligned} \tag{7}$$

Onde a e b são os lados da chapa e h é a sua espessura. Na figura 1, vemos que a função deformação w_0 minimiza com a diminuição de a, b e aumento de h. Nessa mesma figura é visivel que a função massa minimiza com os parâmetros a, b e h. Ou seja, o parâmetro h dá às funções um comportamento antagónico. Assim, o objetivo será conseguir um compromisso entre estas funções.

2 Solução do problema com algoritmo genético

A livraria utilizada foi o Pymoo [?] [?] implementada para a linguagem de programação Python. Para resolver um problema multiobjetivo foi escolhido o algoritmo genetico NSGA-II [?]. Este algoritmo utiliza uma estratégia elitista com partilha de parâmetros (isto é características) na população. Assim, uma população de n indivíduos gera uma população de descendentes através de uma selecção em torneio usando mutações para garantir variabilidade.

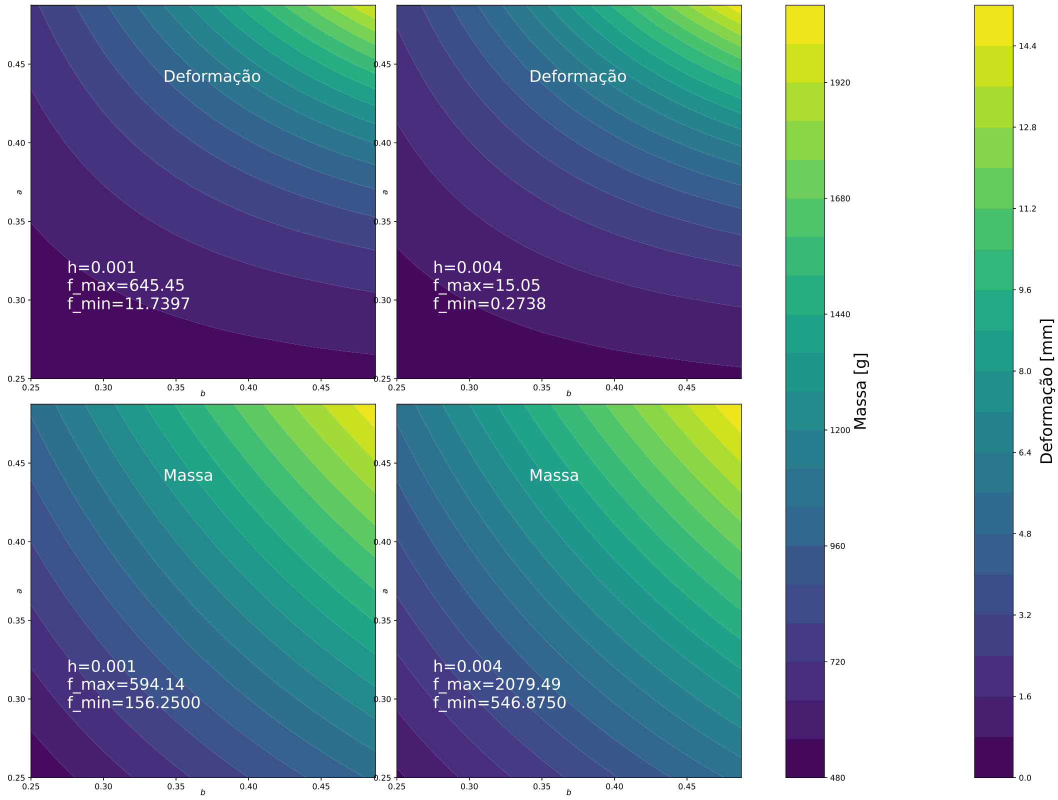


Figura 1: A função massa e deformação desenhada com $h = 0.001, 0.004$

2.1 Parametros do algoritmo relevantes

Para o parâmetro *sampling* for utilizado um campo aleatório. Foi definida uma população de 40 indivíduos com uma descendência de apenas 10. Esta é uma implementação ambiciosa que melhora a convergência em problemas relativamente simples. Para além disso foi ativada a opção que evita uma descendência igual à população original.

2.2 Convergencia

O metodo convergiu em 140 gerações. Visto que que neste problema temos apenas 3 parametros a variar podemos usar o *Hypervolume* como um indicador de performance. Ele compara o ponto de referencia arbritario com a solução encontrada. Estabilidade deste indicador mostrado na figura 2 ajuda a confirmar convergencia.

2.3 Solução

O resultado do método é um conjunto de soluções numa frente de Pareto, resta agora escolher aquela que é mais adequada. Para isso usou-se um método de decomposição,

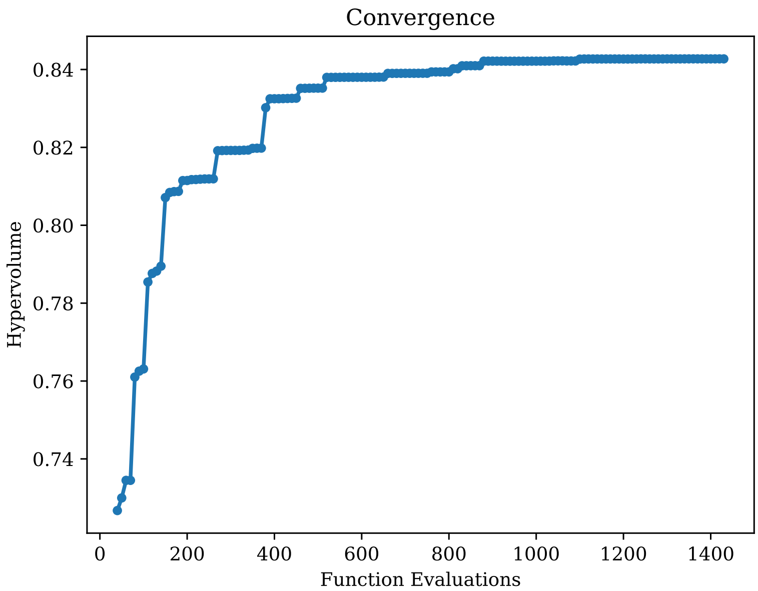
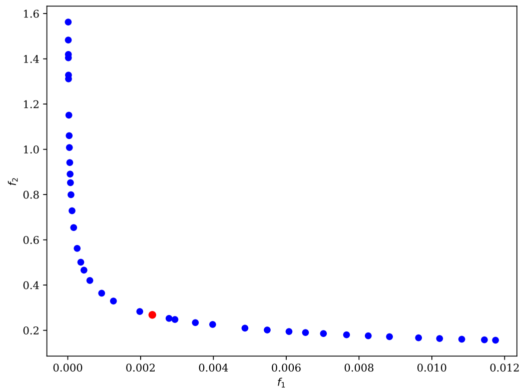
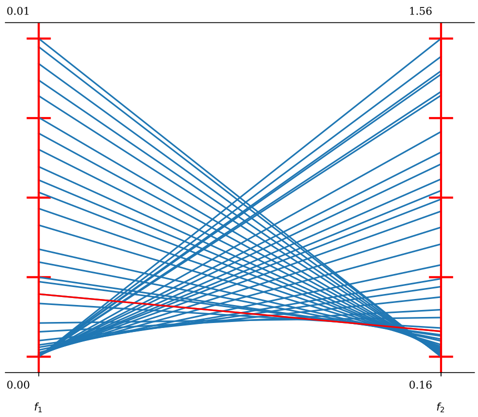


Figura 2: Indicador *Hypervolume* em função do numero de avaliações da função

onde, utilizando uns pesos adequados em função da importância dos objetivos, o problema multidimensional é avaliado como se só existisse um objetivo. Na figura 3, podemos ver esse $trade-off$ entre a massa do sistema e a sua deformação do sistema de teste. Finalmente, solução ideal é encontrada para a deformação $0.0023m$ e massa $0.0023kg$ com os parâmetros $a = 0.2500m$ $b = 0.2500m$ e $h = 0.0017m$.



(a) Frente de Pareto



(b) Soluções em coordenadas paralelas

Figura 3: A solução mais equilibada está a vermelho

3 Otimização sem derivadas

A função a ser minimizada é a 8. Esta não está sujeita a restrições e pode ser visualizada graficamente na figura 4.

$$f = \max[x^2 + y^2, (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x(1) + xy^3)^2] \quad (8)$$

A equação 8 não tem derivadas pelo que as funções MATLAB $fminsearch$, $pattersearch$ e $PSwarm$ foram utilizadas para resolver o problema.

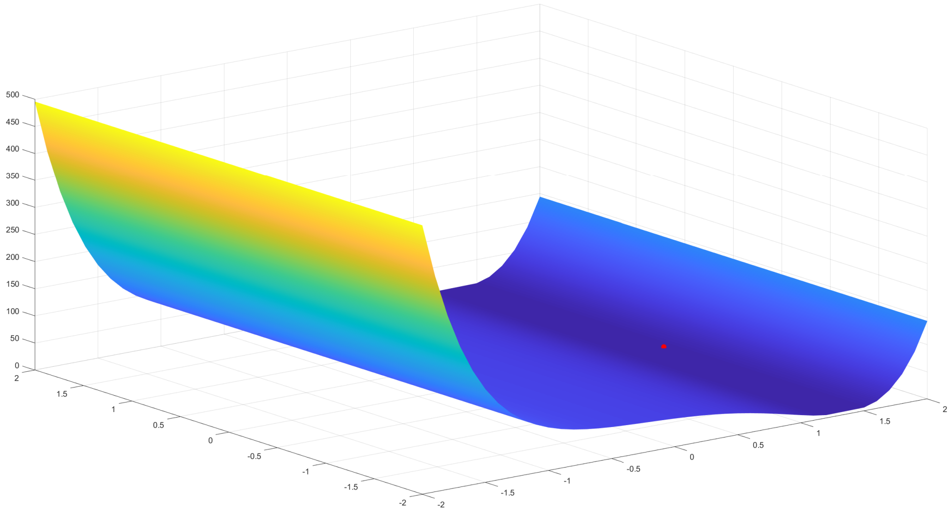


Figura 4: Função f

3.1 $fminsearch$

A função $fminsearch$ encontra o mínimo de uma função multidimensional usando o método de Nelder-Mead simplex direct search. Na primeira iteração as variaveis tomam o valor de x_0=0 e y_0=0. Valor de x e y satisfazem o critériode paragem para $TolX$ de 1e-04 e f(x,y) satisfaz o critériode convergênciapara $TolFun$ de 1e-04 sendo que o processo convergiu para um ponto estacionário. A solução aproximada encontrada toma o valor de 2.0691 para x=1.4383, y=-0.0155 e ocorre na 95ª iteração.

3.2 $pattersearch$

A função $patternsearch$ calcula o mínimo de uma função multidimensional usando o método da pesquisa em padrão. Obteve-se um $exitflag = 1$, ou seja, o método convergiu pois o tamanho da malha é menor que o limite estabelecido. Na 54ª iteração obtivemos o valor da função 2.0693 para x=1.4385, y=-0.0063

3.3 *particleswarm*

A função *particleswarm* determina o mínimo de uma função iterativamente tentando melhorar à solução candidata segundo uma medida de qualidade. Essa medida neste caso foi a mudança relativa do valor da função objetivo. O algoritmo move as candidatas a soluções, ou partículas, num espaço de procura de acordo com uma posição e velocidade.
Obteve-se um $exitflag = 1$. Na 95ª iteração obtivemos o valor da função 2.0691 para x=1.4383, y=-0.0155

3.4 Comentários

Como é visível na tabela houve pouca dispersão de resultados nos diferentes métodos. O algoritmo com pior performance foi o *pattersearch* apresentando o maior desvio especialmente no valor de y.

Método	**x**	**y**	**fVal**
fminsearch	1.4383	-0.0155	2.0691
pattersearch	1.4385	-0.0063	2.0693
particleswarm	1.4383	-0.0155	2.0691

4 Conclusões

A livraria Pymoo é uma interessante ferramenta *opensource* que implementa em C os principais algoritmos de otimização, com um API em python que torna a sua utilização simples flexível e com uma boa performance computacional.
A escolha do algoritmo NSGA-II foi o correto para este tipo de problema, com uma convergência suave sem violações de restrições durante o processo de calculo.
Com a análise da figura 3 é possível verificar que a aplicação de pesos num método de decomposição permitiu obter uma solução equilibrada, evitando soluções na frente de Pareto que privilegiem apenas um dos objetivos.

A O código NSGA2

Neste apêndice está uma parte do código fonte que pode ser consultado na sua totalidade em:
https://github.com/RuiVieira89/top4ICT

Aqui podemos ver a implemtação do modelo com os seus principais parâmetros.

```
from pymoo.model.problem import Problem

class MyProblem(Problem):

def __init__(self):
super().__init__(n_var=3,
n_obj=2,
n_constr=2,
xl=np.array([0.25, 0.25, 0.001]),
xu=np.array([0.5, 0.5, 0.01]),
# elementwise_evaluation=True
)

def _evaluate(self, X, out, *args, **kwargs):

w, sigma_max = function(X)
sysMass = np.prod(X, axis=1)*MaterialDensity

out["F"] = np.column_stack([w, sysMass])
out["G"] = np.column_stack([-w,-sysMass])


problem = MyProblem()

from pymoo.algorithms.nsga2 import NSGA2
from pymoo.factory import get_sampling, get_crossover, get_mutation
from pymoo.algorithms.so_genetic_algorithm import GA

algorithm = NSGA2(
pop_size=40,
n_offsprings=10,
sampling=get_sampling("real_random"),
crossover=get_crossover("real_sbx", prob=0.9, eta=15),
mutation=get_mutation("real_pm", eta=20),
eliminate_duplicates=True
```

```
)

from pymoo.util.termination.default import MultiObjectiveDefaultTermination

termination = MultiObjectiveDefaultTermination(
    x_tol=1e-8,
    cv_tol=1e-6,
    f_tol=0.0025,
    nth_gen=5,
    n_last=30,
    n_max_gen=1000,
    n_max_evals=100000
)

from pymoo.optimize import minimize

res = minimize(problem,
algorithm,
termination,
seed=1,
save_history=True,
verbose=True
)
```

B O código MATLAB

```
function Y = func(x)

    Y = max([x(1).^2 + x(2).^2, (1.5 - x(1) + x(1)*x(2)).^2 + ...
        (2.25 - x(1) + x(1)*x(2).^2).^2 + ...
        (2.625 - x(1) + x(1)*x(2).^3).^2]);

end


close all
clear all
clc



%% fminsearch

options = [];

lb = [];
ub = [];
x0 = [0 0];

fun = 'func';
[x,fval,exitflag,output] = fminsearch(fun,x0,options);


%% pattersearch

options = [];
A = [];
b = [];
Aeq = [];
beq = [];
nonlcon = [];

[x1,fval1,exitflag1,output1] = patternsearch(@func,x0,A,b,Aeq,beq,lb,ub,nonlcon,optio
```

```
%% PSwarm

nvars = 2;
options = [];

[x2,fval2,exitflag2,output2] = particleswarm(@func,nvars,lb,ub,options);

%% PLOTS

[X,Y] = meshgrid(-2:.1:2);

Z(length(X), length(Y)) = 0;

for i=1:length(X)
    for j=1:length(Y)
        Z(i,j) = func([X(i), Y(j)]);
    end
end

surf(X,Y,Z)
hold on
scatter3(x2(1), x2(2),fval2+5, 500,'.', 'r')

shading interp

print = '\n%s X=[%.4f, %.4f]; fval=%.4f \n';
fprintf(print, 'fminsearch', x(1), x(2), fval)
fprintf(print, 'pattersearch', x1(1), x1(2), fval1)
fprintf(print, 'PSwarm', x2(1), x2(2), fval2)
```