Otimização Não-Linear em Engenharia

Trabalhos e Aplicações 2020/2021

universidade de aveiro
theoria poiesis praxis

GRIDS

Conteúdo

Capítulo 1

Introdução

1.1 Enquadramento

No contexto da Unidade Curricular (UC) de Otimização Não-Linear em Engenharia (ONLE), integrada no primeiro semestre do 5º ano do Mestrado Integrado em Engenharia Mecânica (MIEM) da Universidade de Aveiro (UA), os estudantes realizaram um conjunto de trabalhos ao longo do semestre. Estes foram sendo desenvolvidos em contínuo, de forma autónoma, e discutidos semanalmente em aula. Organizados em grupos de dois elementos, cada trabalho incluiu a procura e definição de um problema de Engenharia, a formulação completa do problema de otimização e o desenvolvimento/implementação de métodos adequados à sua resolução.

Com o objetivo de tornar o seu trabalho acessível a outros colegas, do mesmo ano letivo e de anos subsequentes, por iniciativa dos estudantes do Departamento de Engenharia Mecânica, decidiu-se dar continuidade à edição desta compilação de trabalhos. Neste sentido, alguns estudantes participaram não só com o trabalho desenvolvido ao longo do semestre, mas também com algum esforço de edição necessário ao seu ajuste para este formato. Os docentes agradecem esse esforço, convictos de que continuará a constituir uma verdadeira mais-valia no contexto da UC de ONLE e na esperança de que seja uma cada vez mais participada série de documentos desenvolvidos pelos estudantes para toda a comunidade académica no âmbito da Engenharia Mecânica na Universidade de Aveiro.

1.2 Unidade Curricular

O objetivo principal da UC é introduzir os conceitos fundamentais associados ao estudo, à compreensão e à utilização da otimização em problemas de engenharia. Dá-se especial relevo a problemas de Engenharia Mecânica, nomeadamente problemas passíveis de serem modelados computacionalmente. Contudo, a apresentação dos temas é, tanto quanto possível, genérica e transversal a outras áreas do conhecimento. A Otimização Não-linear em Engenharia "[...] é a disciplina das Ciências Aplicadas e Engenharias dedicada a extrair o melhor rendimento possível dos sistemas não-lineares de engenharia recorrendo a técnicas e métodos analíticos, numéricos e computacionais". Estes métodos podem ser, por exemplo, mas não exclusivamente, a simulação e modelação matemáticas.

Consequentemente, como se compreenderá facilmente, a Otimização Não-linear em Engenharia, quando aplicada à resolução de problemas de engenharia, vai buscar as suas raízes a várias disciplinas básicas da matemática, da física, da mecânica e, particularmente, à mecânica computacional. No que diz respeito à matemática, são as disciplinas de análise numérica, álgebra matricial e cálculo diferencial as mais necessárias para o desenvolvimento da Otimização Não-linear em Engenharia. A programação não-linear é a disciplina da matemática que constitui a base para as técnicas que selecionam as melhores alternativas para se atingir os objetivos determinados. No entanto, a utilização destas técnicas na engenharia faz com que o carácter da UC de Otimização Não-linear em Engenharia seja diferente. Adicionalmente, na UC que aqui se apresenta, o uso de técnicas heurísticas e aproximadas diverge da disciplina de programação não-linear da matemática.

No caso da Engenharia, a maior importância não se prende com as demonstrações das formulações e metodologias matemáticas de otimização, mas com a sua aplicação em problemas de aplicação real. Nestes problemas, o objetivo principal não será necessariamente encontrar o ótimo global do problema, mas sim encontrar uma solução admissível que seja melhor do que atual. Adicionalmente, na Otimização Não-linear em Engenharia, utilizam-se também técnicas numéricas (tais como metaheurísticas e métodos aproximados) cuja demonstração matemática de convergência para o ótimo não é possível. Alguns destes métodos, recentemente utilizados com elevado sucesso em Engenharia, são métodos probabilísticos. Refira-se ainda que, nesta UC, os métodos, ferramentas e aplicações da otimização são abordados muitas vezes com recurso a outros métodos numéricos para avaliação do sistema real.

Capítulo 2

Projetos de Otimização em Engenharia

Seguem-se os relatórios de projeto dos grupos da UC. Salienta-se que este documento é efetivamente uma compilação de alguns trabalhos avulsos. Constitui mais uma iteração na divulgação e partilha de trabalhos dentro de uma UC da Engenharia Mecânica na Universidade de Aveiro.

O primeiro documento corresponde ao *benchmark* proposto pelos docentes para o respetivo ano letivo. Este constituiu um problema de referência para todos os estudantes testarem os seus algoritmos e encontra-se aqui formulado, com as diversas soluções e abordagens apresentadas em cada trabalho. Os restantes documentos correspondem aos trabalhos de alguns grupos. Contêm a formulação dos seus próprios problemas e o desenvolvimento dos projetos de otimização, resumindo o resultado das tarefas semanais que foram planeadas e executadas ao longo do semestre. Seguem um formato predefinido, de modo a incluir até 5 páginas do projeto realizado durante o semestre e 2 páginas adicionais referentes à solução apresentada para o *benchmark* de otimização proposto pelos docentes. Na página de *e-learning* da UC poderão ainda ser encontrados outros anexos, nomeadamente os códigos desenvolvidos ao longo de cada projeto.

Benchmark 2020: Weight minimisation of a speed reducer

Constrained design optimisation of a mechanical part

A. Andrade-Campos, J. Dias-de-Oliveira

Initial Guidelines: 20/11/2020

Resumo Today, mechanical design must be performed efficiently. Therefore, optimisation procedures must be applied when designing new mechanical parts. In this challenge, the weight of a speed reducer must be minimised subject to the part's mechanical integrity and reliability. Therefore, constraints on bending stress of the gear teeth, surface stress, transverse deflections of the shafts and stresses in the shafts must be taken into account. This design problem involves seven design variables that characterise the geometry and mechanic characteristics of the part. Integer and continuous variables must be found, making this benchmark a challenge for all optimisation techniques. This design problem is a constrained optimisation problem.

Palavras-Chave Mechanical design · Weight minimisation · Non-linear optimisation · constrained optimisation · Benchmark

1 Introduction

Most engineering design problems are formulated as mathematical programming models. In the last decades, these nonlinear engineering problems have been investigated by different methods. To compare the performance of different optimisation algorithms, several structural engineering applications are often solved to validate or test the suitability of the optimisation algorithms. The speed reducer problem is one of the benchmark problems in structural optimisation. The problem represents the design of a simple gear box used in a light airplane between the engine and propeller to allow each to rotate at its most efficient speed [1].

A speed reducer is simply a gear train between the motor and the machinery that is used to reduce the speed with which power is transmitted. Speed reducers, also called gear reducers, are mechanical gadgets by and large utilised for two purposes. First, they take the torque created by the power source (the input) and multiply it. Second, speed reducers, much as the name implies, reduce the speed of the input so that the output is the correct speed. In other words, gear reducers essential use is to duplicate the measure of torque produced by an information power source to expand the measure of usable work.

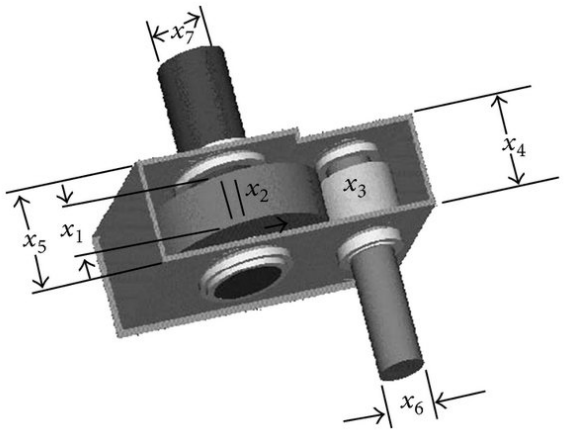

Figura 1. Speed reducer design [1].

The weight of a speed reducer is to be minimised subjected to constraints on levels of bending stress of the gear teeth, surface stress, transverse deflections of the shafts and stresses in the shafts [2]. This design

A. Andrade-Campos, J. Dias-de-Oliveira
Non-Linear Optimisation in Engineering
Year 2020/21
E-mail: gilac@ua.pt; jalex@ua.pt

problem involves seven design variables, as shown in Figure 2, which are the face width, x_1, module of the teeth, x_2, number of teeth on the pinion, x_3, length of the first shaft between bearings, x_4, length of the second shaft between bearings, x_5, diameter of the first shaft, x_6, and diameter of the second shaft, x_7. The third variable, x_3, is an integer, while the rest are continuous. With eleven constraints, this is a constrained optimisation problem.

2 Formulation

The problem is formulated as a constrained nonlinear mathematical programming to minimise the objective function $f(\mathbf{x})$, subjected to the inequality constraints $g_j(\mathbf{x})$, with $j = 1,2,...,11$, stated as

$$\begin{aligned}\text{minimize}\quad f(\mathbf{x}) &= 0.7854x_1x_2^2 \\ &\times \left(3.3333x_3^2 + 14.9334x_3 - 43.0934\right) \\ &- 1.508x_1\left(x_6^2 + x_7^2\right) \\ &+ 7.4777\left(x_6^3 + x_7^3\right) \\ &+ 0.7854\left(x_4x_6^2 + x_5x_7^2\right), \end{aligned} \tag{1}$$

$$\text{subj. to}\quad g_1(\mathbf{x}) = \frac{27}{x_1x_2^2x_3} - 1 \leq 0\,, \tag{2}$$

$$g_2(\mathbf{x}) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \leq 0\,,$$

$$g_3(\mathbf{x}) = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \leq 0\,,$$

$$g_4(\mathbf{x}) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \leq 0\,,$$

$$g_5(\mathbf{x}) = \frac{\sqrt{\left(\frac{745x_4}{x_2x_3}\right)^2 + 16.9\text{E}6}}{110x_6^3} - 1 \leq 0\,,$$

$$g_6(\mathbf{x}) = \frac{\sqrt{\left(\frac{745x_5}{x_2x_3}\right)^2 + 157.5\text{E}6}}{85x_7^3} - 1 \leq 0\,,$$

$$g_7(\mathbf{x}) = \frac{x_2x_3}{40} - 1 \leq 0\,,$$

$$g_8(\mathbf{x}) = \frac{5x_2}{x_1} - 1 \leq 0\,,$$

$$g_9(\mathbf{x}) = \frac{x_1}{12x_2} - 1 \leq 0\,,$$

$$g_{10}(\mathbf{x}) = \frac{1.5x_6 + 1.9}{x_4} - 1 \leq 0\,,$$

$$g_{11}(\mathbf{x}) = \frac{1.1x_7 + 1.9}{x_5} - 1 \leq 0\,,$$

and

$$\begin{aligned}&2.6 \leq x_1 \leq 3.6,\\ &0.7 \leq x_2 \leq 0.8,\\ &17 \leq x_3 \leq 28.0,\\ &7.3 \leq x_4 \leq 8.3,\\ &7.8 \leq x_5 \leq 8.3,\\ &2.9 \leq x_6 \leq 3.9 \text{ and}\\ &5.0 \leq x_7 \leq 5.5\,.\end{aligned}$$

3 Constraints as external penalties

This constrained problem can be transformed into an unconstrained problem using a penalty function. The method of the exterior penalty function is a simple and common approach to handle constraints. The idea behind this method is to transform a constrained optimisation problem into an unconstrained problem, by adding (or subtracting) a penalty function P to the objective function. Solutions that violate the constraints are penalised and the problem is defined as

$$\text{minimize}\quad F(\mathbf{x},r_\text{h},r_\text{g}) = f(\mathbf{x}) + P(\mathbf{x},r_\text{h},r_\text{g})\,, \tag{3}$$

$$\text{subj. to}\quad x_i^\text{min} \leq x_i \leq x_i^\text{max}\,, \quad i = 1,2,...,D\,, \tag{4}$$

where r_h and r_g are penalty factors and F designates the augmented objective function. A general formulation of the exterior penalty function is defined as

$$\begin{aligned}P(\mathbf{x},r_\text{h},r_\text{g}) = r_\text{h}&\left[\sum_{k=1}^{l}[h_j(\mathbf{x})]^\gamma\right] \\ &+r_\text{g}\left[\sum_{j=1}^{m}\left[\max\{0,g_j(\mathbf{x})\}\right]^\beta\right].\end{aligned} \tag{5}$$

where, γ and β are positive penalty constants. The penalty function P is non-existent when the constraint functions h and g are not active.

Furthermore, an exterior penalty function can be classified as dynamic when the current iteration number is associated with the corresponding penalty factors, normally defined in such a way that the value of the penalty function increases over time. Joines and Houck [4] proposed a dynamic penalty method in which individuals are evaluated at iteration i as

$$F(\mathbf{x}) = f(\mathbf{x}) + (C\,i)^\alpha \text{SVC}(\mathbf{x},\beta)\,, \tag{6}$$

where C, α and β are pre-defined constants. SVC($\mathbf{x}$,β) is defined as

$$\text{SVC}(\mathbf{x},\beta) = \sum_{k=1}^{l} H_k(\mathbf{x}) + \sum_{j=1}^{m} G_j^\beta(\mathbf{x})\,, \tag{7}$$

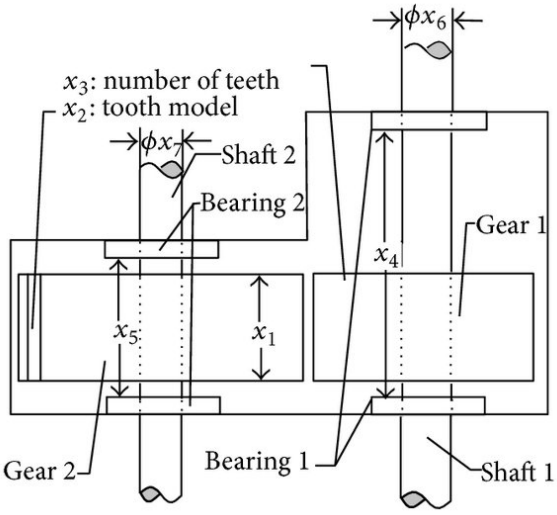

Figura 2. Illustrative representation of the speed reducer design geometry [3].

where functions H_k and G_j are respectively defined as

$$H_k(\mathbf{x}) = \begin{cases} 0 & \text{if } -\epsilon \leq h_k(\mathbf{x}) \leq \epsilon \\ |h_k(\mathbf{x})| & \text{otherwise} \end{cases} \quad \text{and} \quad (8)$$

$$G_j(\mathbf{x}) = \begin{cases} 0 & \text{if } g_j(\mathbf{x}) \leq 0 \\ |g_j(\mathbf{x})| & \text{otherwise} \end{cases}. \quad (9)$$

In this approach, equality constraints are transformed into inequality constraints, where ϵ is the allowed tolerance (normally a very small value). These methods are simple and easy to implement in the advanced optimisation methods. However, a disadvantage is related to the necessity of tuning the parameters depending on the optimisation problem. This problem cannot evaluate and take into account unfeasible solutions (which are not desired). For some optimisation techniques, such as metaheuristics, in order to handle the inequality constraints presented in the design problem, a dynamic penalty function can be used with the parameters $C = 60$, $\alpha = 2$ and $\beta = 1$.

4 Results

As the problem is solved by many authors [5–7] using different algorithms, the best reported result is $f(\mathbf{x}) = 2996.348165$ located at $\mathbf{x} = [3.499999, 0.7, 17, 7.3, 7.8, 3.350215, 5.286683]$. The reported result serves as a reference for the global optimum in the analysis of results. The selected number of function evaluations as a stopping criterion is 10^5.

5 Delivery Instructions

Assessment elements for this *benchmark* must follow the format provided in the respective template (English or Portuguese versions).

References

[1] Ming-Hua Lin et al. "Design Optimization of a Speed Reducer Using Deterministic Technique". In: *Mathematical Problems in Engineering* 23.419043 (2013), pp. 1–7.

[2] J. Golinski. "Optimal synthesis problems solved by means of nonlinear programming and random methods". In: *Journal of Mechanisms* 5.3 (1970), pp. 287–309.

[3] R.V. Rao and J.V. Savsani. *Mechanical Design Optimization Using Advanced Optimization Techniques*. Springer-Verlag London, 2012.

[4] J.A. Joines and C.R. Houck. "*On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's*". In: *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*. Vol. 2. 1994, pp. 579–584.

[5] A. Baykasoglu. "Design optimization with chaos embedded great deluge algorithm". In: *Applied Soft Computing* 12.3 (2012), pp. 1055–1067.

[6] N.B. Guedria. "Improved accelerated PSO algorithm for mechanical engineering optimization problems". In: *Applied Soft Computing* 40 (2015), pp. 455–467.

[7] R.V. Rao and G.G. Waghmare. "A new optimization algorithm for solving complex constrained design optimization problems". In: *Engineering Optimization* 49.1 (2017), pp. 60–83.

Otimização Não-Linear em Engenharia | MIEM.2021
(artigo/relatório de projecto)

Resolução de problemas de otimização com o algoritmo *Spiral Optimization*

Maximização da velocidade média de uma corrida automóvel e otimização de forma de uma cremalheira para minimizar picos de binário na roda traseira da bicicleta

Álvaro Francisco · Luís Verdasca

Submetido: 02/03/2021

Resumo O presente documento descreve o estudo e implementação do algoritmo *Spiral Optimization* em problemas de otimização de engenharia. Os problemas escolhidos e analisados vêm do quotidiano dos autores: planeamento das paragens na *box* para ter a melhor velocidade média; modelar uma cremalheira dianteira para atenuar os picos de binário na roda traseira de uma bicicleta. É também apresentada a implementação do mesmo algoritmo num problema modelo, o *benchmark*, proposto pelos docentes da Unidade Curricular (U.C.) com o objetivo de minimizar o peso de um redutor de velocidades. Este último com o objetivo de testar a robustez e eficácia do algoritmo selecionado pelos autores.

Palavras-Chave Otimização · SPO · Velocidade média · Spiral Optimization Algorithm · Binário · Engenharia · Cremalheira

1 Introdução

O primeiro problema foca-se no desporto motorizado, em que não só se valoriza a capacidade de construir o melhor veículo, como a melhor estratégia. Uma corrida pode ser perdida por uma ordem errada. A equipa que tiver a melhor estratégia é a favorita a vencer, sendo importante otimizá-la.

O tema do segundo problema advém da perda de rendimento pela abordagem mais cuidadosa a certas condições do percurso na modalidade de *mountain biking*. A conjugação da elevação do piso com o facto do mesmo ser escorregadio obrigam a uma gestão mais técnica da força disponibilizada na pedalada, que em consequência, leva a uma adaptação do ritmo do ciclista, sendo benéfico o estudo de uma solução.

Por fim, o terceiro problema é um *benchmark* proposto pelos docentes da U.C., transversal a todos os grupos de trabalho, neste caso de forma a avaliar o desempenho e robustez dos algoritmos selecionados para otimização dos problemas autopropostos.

A. Francisco
n.º 80685
E-mail: alvaromcfrancisco@ua.pt

L. Verdasca
n.º 86025
E-mail: lcverdasca@ua.pt

2 Definição de problemas de engenharia

2.1 Maximização da velocidade média numa corrida automóvel

Para uma equipa saber a estratégia ideal de corrida, deve conhecer tanto o comportamento geral do desgaste dos pneus, como a quantidade de combustível para determinar a evolução dos tempos por volta. Assim, com esses perfis gerados por modelos numéricos é possível obter o número teórico de voltas que irá ser possível completar.

Existem 2 tipo de corridas: resistência e *sprint*. As provas de resistência têm um limite de tempo ou distância a ser cumprido e as corridas de *sprint* contam com um limite de voltas ao circuito. Para otmizar a estratégia, deve-se maximizar a distância percorrida para o primeiro tipo de corrida ou minimizar o tempo total para completar o *sprint*. Em ambos os casos pode-se traduzir o problema na maximização da velocidade média na corrida. Para compreender melhor o comportamento de cada composto durante o seu ciclo de utilização divide-se a corrida em secções, chamadas de *stints*. Cada *stint* representa a porção de corrida desde que o piloto sai da *box* até reentrar para fazer a mudança de pneus e/ou reabastecer o automóvel, ou seja, cada

conjunto de pneus irá ter uma duração de *stint* e uma distância percorrida sobre a forma de voltas ao circuito que fez. O estudo vai incidir sobre uma corrida de resistência disputada no Circuito Internacional do Algarve, sendo que a Figura 1 é um exemplo ilustrativo dos resultados que se pretendem obter.

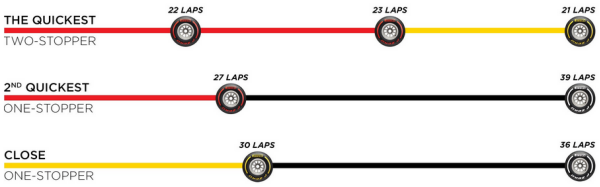

Figura 1. Comparação de estratégias numa corrida de Fórmula 1 [1].

2.2 Minimização dos picos de binário na roda traseira da bicicleta

A pedaleira de uma bicicleta funciona sobre o princípio da geração de binário por meio de uma força aplicada num braço alavanca. Este sistema de equilíbrio de dois ciclos desfasados em 180 graus cria um perfil de força de rotação aproximadamente harmónico, como se pode ver na Figura 2. Nas zonas de pico, a força resultante na roda, por vezes, ultrapassa o valor da força de atrito, criando assim perdas de tração pontuais. De facto, uma forma de atenuar este problema é modelar o binário originado na pedalada reformulando os componentes constituintes da pedaleira. Sendo que atualmente já existem exemplares de cremalheiras não circulares, resultantes de testes experimentais, e que a alteração do braço do pedal ao longo do ciclo de rotação não é viável, o foco é modelar teoricamente a configuração ideal de uma cremalheira dianteira.

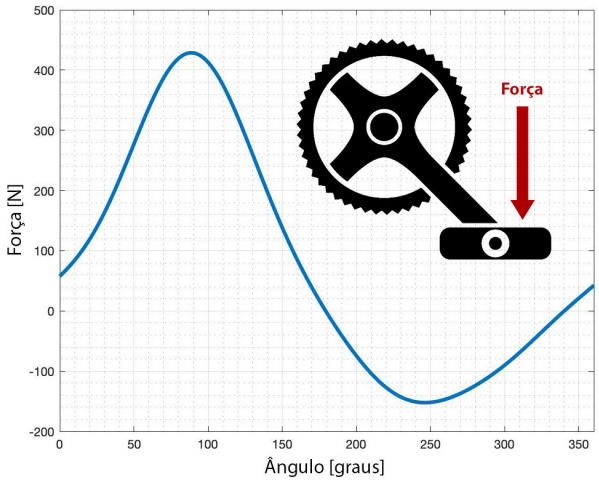

Figura 2. Perfil de força num pedal num ciclo de revolução e diagrama da força na pedaleira.

3 Formulação dos problemas de otimização

3.1 Maximização da velocidade média numa corrida automóvel

Como definido na Secção 2.1 o tempo de *stint* de cada composto representa a variável de projeto:

Procurar t_{stint} de modo a: (1)

$$\text{Maximizar} f(t_{\text{stint}}) = \frac{\sum_{i=1}^{N}(\frac{3600 \cdot n_{\text{lap}} \cdot d_{\text{track}}}{t_{\text{stint}}})}{N}$$

$$\text{sujeito a } \sum^{n} t_{\text{stint}_N} \geq 14.400, \quad N = 1,...,n,$$

Onde n_{lap} é o número de voltas realizado nesse *stint* e d_{track} é o perímetro da pista em km. É preciso restringir o t_{stint} para planear as paragens na *box*, onde Q é a quantidade de combustível no tanque e W a percentagem de desgaste do pneu:

$$Q \geq 3{,}5 \text{ l} + 1 \text{ volta}, \quad (2)$$
$$W \geq 5\ \% + 1 \text{ volta}.$$

A Tabela 1 junta as restrições da quantidade de compostos que a equipa tem disponível para a corrida, n_{tyre}, com o t_{pit}, em segundos, associado a cada composto:

Tabela 1. Tabela de compostos disponíveis e tempo perdido na *box*.

	Soft	Medium	Hard
t_{pit}	60,54	60,55	63,54
n_{tyre}	1	2	2

Para obter um somatório dos tempos por volta t_{lap} criaram-se funções que replicam a sua evolução num estado estacionário. Extraíram-se dados do simulador para recriar de forma realista a influência da degradação dos pneus e do consumo de combustível e aplicaram-se como penalizações a um tempo base. Para efeitos de teste não foi considerada a variabilidade aleatória [2]. Então,

$$t_{\text{lap}} = t_{\text{base}} + p_{\text{f}} + p_{\text{t}}, \quad (3)$$

onde, as funções p_{f} e p_{t} são os fatores corretivos ao tempo base devido ao consumo de combustível e ao estado de degradação dos pneus por volta, respetivamente. Cada tipo de pneu irá ter um conjunto de funções únicas que definem o seu comportamento (v.d. Anexo B.1).

Na Figura 3 observa-se uma comparação da evolução do tempo por volta para os três compostos: a vermelho está o composto duro; a azul o composto médio e a preto representa o composto macio.

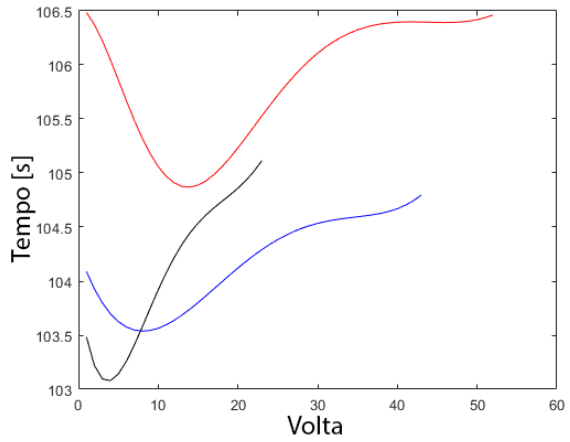

Figura 3. Tempos por volta para os diversos compostos.

Este problema é não-linear de pequena escala, com a variável de projeto a apresentar natureza contínua.

3.2 Minimização dos picos de binário na roda traseira da bicicleta

$$\begin{aligned} &\text{Procurar } \mathbf{R} \quad \text{de modo a :} \qquad (4)\\ &\text{maximizar } f(\mathbf{R}) = \sum_{n=1}^{N} (M_{\text{original}}(n) - M_{\text{novo}}(n))^2 \\ &\text{com: } M_{\text{novo}}(\mathbf{R}) = \frac{M_{\text{frente}_i} \cdot r_{\text{traseiro}}}{R} \\ &\qquad 59.95 \leq R_i \leq 68.05, \quad i = 1{,}7 \\ &\qquad \overline{M_{\text{original}}} \leq \overline{M_{\text{novo}}} \end{aligned}$$

O objetivo de atenuar os picos de binário pode ser alcançado através do aumento da distância entre as curvas de Binário Original e a Nova, gerada com base na variável de projeto. Sendo a primeira constante, o aumento da distância da segunda curva traduz-se numa diminuição do valor do pico. Com isto, a função objetivo tem por base o método dos mínimos quadrados [3], neste caso com o efeito de aumentar a diferença entre estas duas curvas. A variável de projeto será um vetor de raios da cremalheira para 7 posições, compreendidas em meio ciclo de rotação (0 graus até 180) e espaçadas igualmente entre si.

O tamanho da cremalheira a modelar é de 30 dentes e de acordo com [4] esta modelação restringe o intervalo de procura entre exemplares de 28 e 32 dentes, raios 59,95 mm e 68,05 mm, respetivamente. O binário resultante no eixo da cremalheira dianteira, $\mathbf{M}_{\text{frente}}$, é previamente calculado para cada ponto, através de um perfil de força experimental de um ciclista em [5] para um braço de pedal de 175 mm. Para o cálculo do $\mathbf{M}_{\text{novo}}$ foi fixado o valor de r_{traseiro} de 96,35 mm, correspondente a uma cremalheira de 46 dentes. Todos estes parâmetros foram definidos com base no equipamento dos autores, mas é importante deixar a ressalva de que poderiam ter sido escolhidos outros valores para a cremalheira a modelar e o respetivo intervalo de raios, para o comprimento do braço do pedal e ainda para o raio da cremalheira traseira.

Este problema é classificado como não linear de pequena escala, com uma variável de natureza contínua.

4 Análise de sensibilidade

4.1 Maximização da velocidade média numa corrida automóvel

Uma vez definida a formulação do trabalho, testou-se a variação da velocidade média da corrida, em km/h, usando diversos compostos, como descrito na Tabela 2.

Tabela 2. Análise de sensabilidade ao comportamento da função objetivo

stint 1	*stint* 2	*stint* 3	*stint* 4	*race*
144,65	144,65	145,73	-	145,01
144,65	144,65	148,07	-	145,79
146,65	146,65	146,65	147,61	146,89

Conclui-se que é irrelevante o número de paragens desde que se use sempre o composto mais rápido, uma vez que o tempo que ganha por volta compensa as paragens extra que necessita.

4.2 Minimização dos picos de binário na roda traseira da bicicleta

De modo a perceber a tendência do problema, através da ferramenta *MATLAB* foi modelado de forma iterativa um vetor $\mathbf{R}$ e calculados os vetores dos Momentos para cada posição.

O valor médio do Binário Novo de saída aumenta em 2,6 % e o valor correspondente da função objetivo é 174,72 N · m.

5 Avaliação

5.1 Maximização da velocidade média numa corrida automóvel

As funções auxiliares de evolução de tempo por volta descritas no Anexo B.1 não apresentam variação, ou

seja, 2 conjuntos de pneus do mesmo composto apresentam a mesma velocidade média. Não foi considerada nenhuma penalização por ritmo do piloto porque são permitidos reabastecimentos, não fazendo sentido haver poupança de combustível. Outra consideração foi o facto de não ter sido implementada uma paragem na *box* para a mudança de piloto obrigatória. A geração do vetor para avaliar a função-objetivo foi aleatória, com todas as restrições ativas, numa família de 10.000 vetores.

5.2 Minimização dos picos de binário na roda traseira da bicicleta

Desprezando o efeito da perna dominante, cada uma executa o mesmo perfil de força, apenas com desfasamento de 180 graus, sendo assim previsto a simetria da cremalheira. Mais, inicialmente foi programada uma terceira restrição com o intuito de evitar discrepâncias abruptas de raio para raio no vetor **R**. Alguns testes preliminares demonstraram que esta não permitia a atualização do vetor. A reformulação desta condição, levou à redução do comprimento do vetor de raios para 7 pontos, como referido na Secção 3.2. Desta forma, a diferença entre os valores dos raios poderia ser maior, sendo que os pontos intermédios aos do vetor teriam de ser interpolados de forma a obter uma curva progressiva. Por fim, e para simplificação dos passos intermédios considerou-se a eficiência de transmissão da corrente 100 %.

6 Algoritmos de otimização

O algoritmo selecionado é conhecido como o *Spiral Optimization Algorithm* ou SPO. É da família dos métodos metaheurísticos e inspirou-se no comportamento das espirais orgânicas [7]. O algoritmo recebe 4 *inputs*: o número de iterações como critério de paragem, a população de pontos no espaço de procura, um ângulo de rotação e um raio de procura em relação ao centro da espiral. Com essa informação, o algoritmo compara, em cada iteração, toda a população à volta do melhor ponto, e recalcula uma matriz rodada e contraída sobre ele, como possível verificar na Figura 9, presente no Anexo A. A programação de ambos os problemas foi implementada em linguagem *Python* sendo que os autores dispunham de um código base do algoritmo [9]. Foi também testado o seu funcionamento com funções matemáticas multidimensionais (n $>$ 2) como *Rastrigin* e *Sphere*, com resultados de convergência promissores.

7 Resultados

7.1 Maximização da velocidade média numa corrida automóvel

O algoritmo funcionou numa configuração de procura direta, por causa da programação simplificada do problema, executando 10.000 avaliações. Convergiu sempre para a solução ideal e em menos de 1.000 avaliações. Na Figura 4 é possível observar a progressão da função custo até convergir na melhor solução.

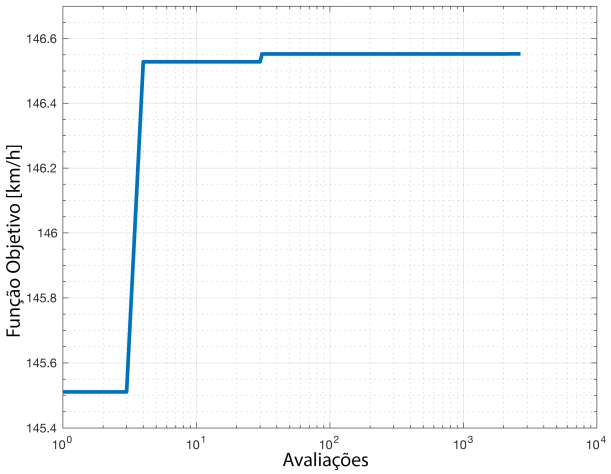

Figura 4. Evolução da função objetivo para o melhor resultado.

A solução ideal deste problema é $f(\mathbf{x})$ =146,56 km/h, com **x**=[148,24; 146,66; 146,66; 144,65], que se traduz no uso de dois compostos médios, um composto duro e um composto macio. Isto vai ao encontro das conclusões retiradas na Secção 4.1, onde os compostos preferidos são os mais rápidos, independentemente do número de paragens na *box*.

O vetor solução **x** é arbitrário, não tendo ordem específica. No entanto, quando o último *stint* ultrapassa a restrição do tempo limite de corrida, a velocidade média foi admitida como um *stint* completo, quando na verdade ele deveria ser interrompido assim que cumprisse a restrição e resultaria num vetor **x** diferente.

Deve-se ter em conta que a componente estratégica depende muito da decisão humana, que as simulações não são capazes de prever, mas ter conhecimento do que pode acontecer é imprescindível. No futuro deverá ser desenvolvido um caso de estudo para uma pista mais longa e analisar se a solução é diferente.

7.2 Minimização dos picos de binário na roda traseira da bicicleta

Depois de alguns ensaios teste, os resultados mais consistentes fizeram-se com uma população inicial de 1000 indivíduos para 1000 iterações, perfazendo um total de 10^6 de avaliações.

O ângulo de rotação e o raio de procura escolhidos foram respetivamente $\frac{\pi}{32}$ e 99 %, incluindo todo o espaço de procura. O valor final máximo da função objetivo é de, aproximadamente, 257 N · m, sendo **R**=[61,65; 68,04; 59,95; 59,95; 59,95; 59,95; 62,25] mm a solução correspondente ao vetor. É possível discernir que a solução esperada seria um vetor bem distribuído em todo o intervalo, igualmente espaçado entre raios. Estes resultados podem ser vistos nas Figuras 5 e 6.

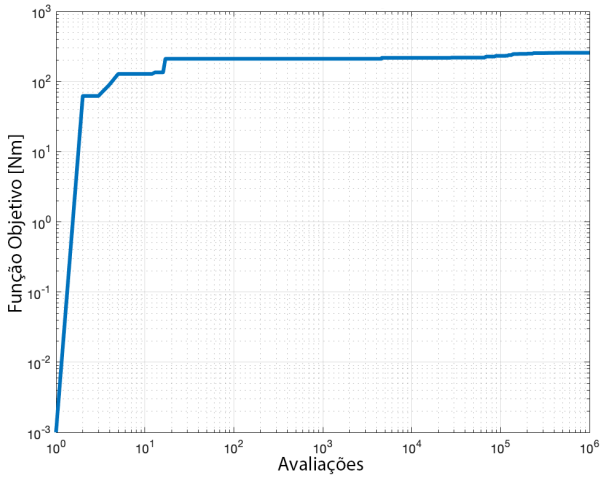

Figura 5. Progressão da função objetivo para o melhor resultado.

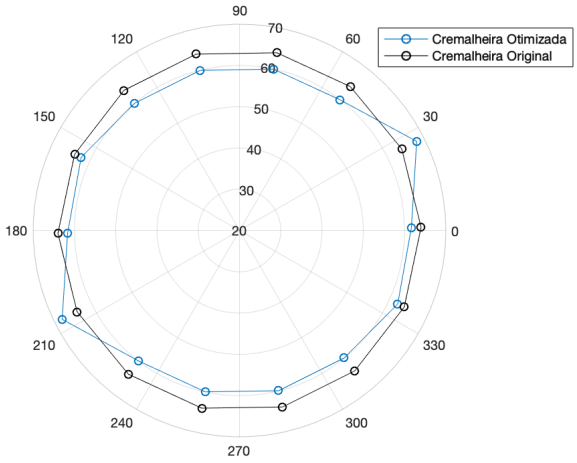

Figura 6. Cremalheiras Original e Otimizada.

Finalmente, o novo valor de Binário médio sofreu um aumento em 5%, valor superior ao teórico. Existe uma redução do valor de pico em cerca de 8 %, como é possível observar através Figura 7. O objetivo do projeto foi alcançado ainda que seja necessário uma interpolação adicional e, consequentemente, recalculado o valor médio final do binário.

O comportamento do algoritmo na procura de resultados demonstrou ser pouco preciso, flutuando entre soluções até às 900.000 avaliações, como é possível observar na Figura 5 e em mais detalhe no gráfico de evolução dos raios da Figura 11 do Anexo C.1. Mais ainda, em 15 ensaios nunca convergiu para a mesma solução e a tendência nem sempre mostrou ser a apresentada anteriormente.

Para efeitos de comparação, a forma da cremalheira obtida está de acordo com algumas das soluções que já existem para consumo dentro desta modalidade.

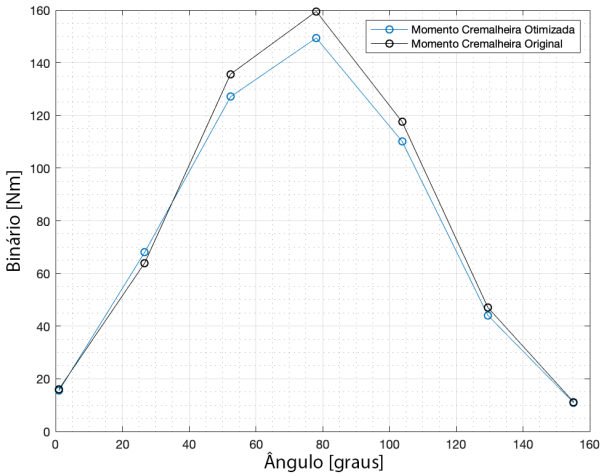

Figura 7. Curvas de Binário Original e Otimizada.

8 Conclusões

Conclui-se desta forma que o algoritmo mostrou-se robusto para a resolução dos problemas propostos pelos autores, com sucesso em ambos. O algoritmo tende rapidamente para a vizinhança da solução ótima, mas necessita de grande quantidade de dados para convergir completamente, uma vez que para poucas avaliações o valor da função-objetivo irá apresentar flutuações entre testes.

Os autores pretendem continuar a elaboração dos trabalhos, no primeiro caso desenvolvendo o realismo do simulador criado e no segundo caso a criação de um modelo físico para teste, e ainda a implementação do cálculo para obtenção de modelos físicos personalizados.

A Benchmark 2020

A.1 Enquadramento do benchmark

No contexto da Unidade Curricular de Otimização Não-Linear em Engenharia, foi proposta a realização de um *benchmark* para avaliar a robustez e a precisão do algoritmo selecionado para resolução dos problemas autopropostos pelos estudantes 1, permitindo originar discussão e comparação entre as diferentes técnicas de otimização não-linear. O *benchmark* do presente ano letivo 2020/2021 baseia-se na minimização do peso de um redutor de velocidade que será acoplado a um motor de uma pequena aeronave, como exemplificado na Figura 1. Este problema de Engenheria para otimização é bastante conhecido, alvo de implementação de diversos processos de melhoria pela sua relevância no estudo de construção mecânica, podendo ser consultado em [6].

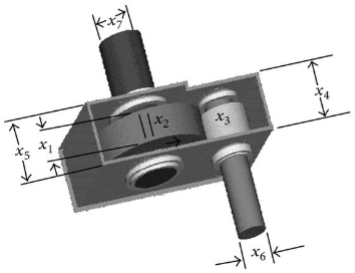

Figura 8. Diagrama do *benchmark* de minização de um redutor de velocidades [6].

Assim, torna-se fulcral a escolha de uma metodologia capaz de encontrar o mínimo global e, por conseguinte, ótimo, sem que deixe de obedecer às condições impostas pelo enunciado do problema. Com a sua formulação composta por 11 restrições de desigualdade e 7 variáveis, identificadas por: x_1, a largura da roda dentada; x_2, o número de dentes da roda dentada; x_3, o diâmetro do pinhão; x_4, a largura entre eixos do pinhão; x_5, a largura entre eixos da roda dentada; x_6, o diâmetro do eixo que faz girar o pinhão e x_7, o diâmetro do veio que suporta a roda dentada. Sendo de extrema complexidade prever a tendência no espaço de soluções. Para lidar com essas restrições, usamos o método de penalidades externas [6], que irá ser desenvolvido nos próximos capítulos.

O algoritmo selecionado foi o *Spiral Optimization*, conhecido pela sigla SPO ou SOA. Este pertence à família das metaheurísticas, baseado na formação de espirais orgânicas em torno de um ponto central, [7, 8]. O seu funcionamento pode ser analisado no fluxograma da Figura 9.

A.2 Implementação

O algoritmo é implementado em linguagem *Python*, com base num código pré-existente em [9], que contém as matrizes de rotação e contração para n dimensões, essenciais para o funcionamento do SPO. Os ciclos de rotação foram modificados de modo a ser mais intuitiva a programação das diferentes funções-objetivo e restrições através do uso de estruturas. Após estas modificações, foi programada a função-objetivo, as restrições através do método de penalidades externas [6],

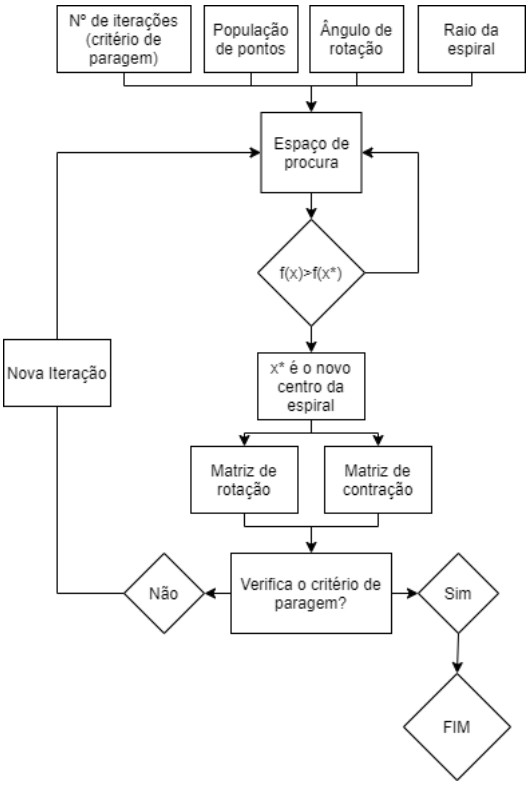

Figura 9. Diagrama de Fluxo do Algoritmo SPO.

a geração do espaço de procura e a função aumentada com os valores de α e β, valores contantes de penalidade positiva, com o valor de 10^{15} específicos ao SPO [8].

A.3 Resultados

Os melhores resultados foram obtidos com avaliações à função objetivo iguais ou superiores a 10^6. Repetimos o ensaio 10 vezes para testar a consistência de convergência deste método. O melhor resultado obtido foi: $f(x)$=3003,9, com $\mathbf{x}^*$=[3,5; 0,7; 17,0; 7,9; 7,9; 3,4; 5,3]. O resultado médio da família dos 10 ensaios foi de: $f(x)$=3050,6 e $\mathbf{x}^*$=[3,5; 0,7; 17,0; 7,9; 8,0; 3,4; 5,3]. Concluindo assim que houve alguma flutuação de valores, apesar de não ser significativa. O erro do valor médio para os resultados esperados é cerca de 1,8 %.

Analisando o melhor resultado, observou-se que $\mathbf{x}^*$ se afasta apenas um pouco do valor esperado; isto vem de um erro de programação que foi a gestão incorreta da avaliação do número de dentes (x_3), que deveria ser um número inteiro não sendo tratado como tal, obrigando o programa ser computacionalmente oneroso e piorando os resultados das variáveis que lhe seguiam, como o x_4, x_5, x_6, x_7 fazendo afastá-los dos resultados esperados.

Em discussão, é perceptível a maior desvantagem deste algoritmo: a sua exigência computacional, uma vez que, para obter soluções mais próximas da solução ótima deve-se trabalhar com grande quantidade de dados. De facto, o algortimo gera as matrizes com base num raio de procura e um ângulo de rotação, valores que calibrados para uma maior precisão obrigam ao cálculo minucioso. Este factor induz no aumento do número de avaliações incrementando a exigência computacional. Em suma, demonstrou ser um algoritmo pouco eficiente,

apesar dos resultados tenderem a aproximar-se da solução ótima.

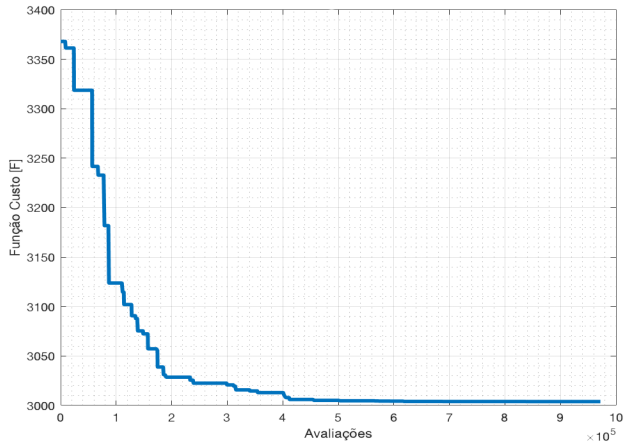

Figura 10. Evolução da Função de Custo para o melhor resultado.

Referências

1. Pirelli, *2019 Spanish Grand Prix - Qualifying* , https://press.pirelli.com/2019-spanish-grand-prix—qualifying/, 2019
2. C. Sulsters and R. Bekker, *Simulating Formula One Strategies*, Vrije Universiteit Amsterdam, Amesterdão, Holanda, 2018
3. Wikipédia, *Método dos mínimos quadrados*, https://pt.wikipedia.org/wiki/M%C3%A9todo-dos-m%C3%ADnimos-quadrados (25/11/2020), 2017.
4. JackWare, *mtbr*, https://www.mtbr.com/threads/diameter-of-oval-chainring.1058531/ (25/11/2020), 2017.
5. Carpes. F, et al., Aplicação de força no pedal em prova de ciclismo 40 km contra-relógio simulada, Vol(**19**)::p105–13, abril/junho, 2005.
6. A. Andrade-Campos and J.Dias-de-Oliveira, *Benchmark 2020: Weight minimisation of a speed reducer*, Universidade de Aveiro, Aveiro, Portugal, 2020
7. Tamura K. and Yasuda K., Spiral Dynamics. Inspired Optimization , *Journal of Advanced Computational Intelligence and Intelligent Informatics*, Vol(**15**).8, 2011.
8. Kuntjoro Adji Sidarto and Adhe Kania, *Spiral Optimization Algorithm*, em Industrial and Financial Mathematics Research Group, Faculty of Mathematics and Natural Sciences, Bandung, Indonesia, December, 2017.
9. lestarisusi, *Spiral Optimization Algorithm in Python*, https://github.com/lestarisusi/Spiral-Optimization-Algorithm/ (10 de Dezembro de 2020), 2019.

B Maximização da velocidade máxima de uma corrida automóvel

B.1 Formulação

Nesta secção do apêndice são apresentadas as fórmulas de modelação de consumo de combustível e degradação do pneu, usadas para calcular as penalidades da Equação 3. As suas variáveis serão a são Q, quantidade de combustível no tanque para cada volta em litro, L, o número da volta do *stint* em que o piloto se encontra e W, que corresponde à percentagem do desgaste do pneu. Para o pneu macio, as funções são:

$$\begin{aligned} p_f(Q) &= -8{,}887\cdot(\tfrac{Q}{100})^3+6{,}948\cdot(\tfrac{Q}{100})^2-0{,}9224\cdot(\tfrac{Q}{100})-0{,}344 \\ W(L) &= -0{,}04283\cdot L+1 \\ p_t(W) &= 17{,}8\cdot W^5-19{,}8\cdot W^4-0{,}385\cdot W^3+3{,}64\cdot W^2-2{,}69\cdot W+3{,}27. \end{aligned} \quad (5)$$

O pneu médio é definido por:

$$\begin{aligned} p_f(Q) &= -0{,}09297\cdot(\tfrac{Q}{100})^3+0{,}08361\cdot(\tfrac{Q}{100})^2+0{,}7618\cdot(\tfrac{Q}{100})-0{,}744 \\ W(L) &= -0{,}02175\cdot L+1 \\ p_t(W) &= 24\cdot W^4-41{,}54\cdot W^3+21{,}86\cdot W^2-5{,}843\cdot W+2{,}8. \end{aligned} \quad (6)$$

Por fim o pneu duro define-se como:

$$\begin{aligned} p_f(f) &= 0{,}1763\cdot(\tfrac{Q}{100})^3-0{,}3253\cdot(\tfrac{Q}{100})^2+0{,}3028\cdot(\tfrac{Q}{100})+1{,}188 \\ W(L) &= \tfrac{2\times10^{-7}\cdot L^5-4\times10^{-5}\cdot L^4+0{,}003\cdot L^3-0{,}098\cdot L^2-0{,}098\cdot L+100}{100} \\ p_t(W) &= 134{,}8\cdot W^5-292{,}6\cdot W^4+213{,}2\cdot W^3-58{,}56\cdot W^2+1{,}975\cdot W+2{,}688. \end{aligned} \quad (7)$$

C Minimização dos picos de binário na roda traseira da bicicleta

C.1 Resultados

Em seguida é possível observar a evolução dos valores de raios da cremalheira para as 7 posições do vetor de **R**, Figura 11.

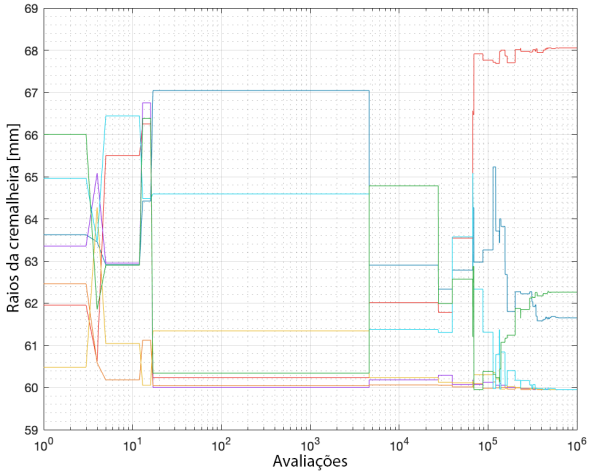

Figura 11. Evolução dos raios da cremalheira.

Otimização Não-Linear em Engenharia | MIEM.2021
(artigo/relatório de projecto)

Resolução de problemas utilizando o algoritmo *Enhanced Firefly Algorithm*

Minimização dos custos de material de um implante cranial obtido por impressão 3D e minimização do material utilizado numa embalagem metálica

Pedro Santos · Bruno Rodrigues

Submetido: 03/03/2021

Resumo Este relatório apresenta a aplicação do algoritmo *Enhanced Firefly Algorithm* a dois problemas de otimização. O primeiro problema aborda a minimização de custos de material de um implante cranial e o segundo a minimização de material utilizado numa embalagem metálica. É também abordado um problema clássico de otimização, o *Benchmark*, com o intuito de testar e validar o algoritmo.

Palavras-Chave Otimização Não-Linear · *Enhanced Firefly Algorithm* · Implante cranial · Embalagem metálica

1 Introdução

A otimização é um processo fulcral em engenharia pois permite a obtenção de resultados otimizados de uma forma mais elegante quando comparado com técnicas de força bruta, levando a poupanças de tempo e custos e com garantias de viabilidade de soluções.

Nesse seguimento, foi objetivo aplicar um algoritmo de otimização já desenvolvido na resolução de dois problemas de engenharia. O primeiro problema, minimização de custos de material de um implante cranial obtido por impressão 3D, está relacionado com o setor da biomecânica e surge no seguimento da procura da obtenção de componentes de forma mais rápida e barata, utilizando outros materiais e técnicas de fabrico. Já o segundo, que tem como objetivo a minimização de material numa embalagem metálica, situado no setor do embalamento, propõe a redução do consumo de matéria prima sem colocar em causa a integridade mecânica do produto.

De forma a obter maior confiança nos resultados fornecidos pelo algoritmo, este foi testado num problema de *benchmark* com soluções conhecidas.

Pedro Santos
n.º 84747
E-mail: pedroosantos2@ua.pt

Bruno Rodrigues
n.º 88625
E-mail: bru.seabra@ua.pt

2 Definição de problemas de engenharia

2.1 Minimização dos custos de material de um implante cranial

Os avanços que recentemente se têm desenvolvido permitiram que a cranioplastia tenha ao seu dispor componentes mais simples e de obtenção mais rápida e barata, nomeadamente implantes craniais obtidos por impressão 3D [1]. Estes componentes são especiais em termos de engenharia e *design*, uma vez que o seu projeto está dependente de cada situação e paciente e requerem especificações biomecânicas importantes, como a biocompatibilidade dos materiais usados e a existência de furos para a drenagem de fluídos e integração de tecidos.

Numa lógica de simplificação, e uma vez que não há um *design* padronizado, foi tomada a geometria de uma calote esférica como sendo a aproximação do implante, com altura e raio definidos. O objetivo passa por minimizar o custo de material tomando como referência o custo por quilograma do filamento para impressão 3D fazendo variar propriedades geométricas (espessura, número e raio dos furos) e o material do implante. Relativamente aos furos, optou-se por fazer uma distribuição radial em quatro camadas, com o número de furos e o seu raio variáveis (Fig 1(a)). Por seu lado, o conjunto

de materiais considerados para a análise pertencem à família dos polímeros.

De modo a testar a viabilidade do implante, foi efetuado um teste de impacto com recurso ao *software Abaqus*, aplicando uma força de 100 N (valor de referência em alguns testes relacionados com esta matéria [2]) na zona central do implante, com a zona lateral encastrada (simplificação do método de fixação). Assim sendo, será possível retirar os valores máximos da tensão gerada e do deslocamento.

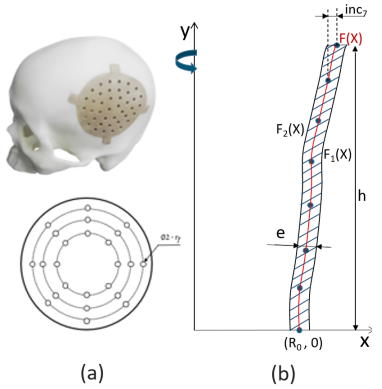

Figura 1.: lustrações dos problemas: (a) exemplo de implante cranial e distribuição de furos a realizar (b) forma do corpo da embalagem.

2.2 Minimização do material utilizado numa embalagem metálica

As embalagens metálicas estão presentes no nosso dia-a-dia, desde os desodorizantes a refrigerantes. Este tipo de embalagens é crucial no armazenamento de bens nos dias de hoje.

O caso especifico em análise neste problema será o de uma embalagem metálica para transporte de produtos, que usualmente possui uma geometria cilíndrica. Sendo este produto relativamente simples e o seu fabrico de baixa complexidade, o material utilizado representa a maioria dos custos a si associados, pelo que é imperativo a sua minimização.

Para a realização deste problema apenas foi considerada a face lateral da embalagem metálica, não considerando assim o tampo ou o fundo, por serem componentes com formas pré-definidas. Por uma questão de simplificação do problema e por ser também mais usual, a geometria cilíndrica foi a escolhida para este problema, tornando-o assim axissimétrico.

A quantidade de material utilizado é dada em função da espessura da face, da altura e da forma que esta possui. A forma da face é representada através de uma *spline* cúbica definida por nove pontos ao longo da altura, suficientes para representar qualquer forma requerida uma vez que não são necessárias, nem desejadas, variações bruscas.

3 Formulação dos problemas de otimização

3.1 Minimização dos custos de material de um implante cranial

A função-objetivo passa, de uma forma muito simplista, pela multiplicação do custo por quilograma do filamento pela massa do implante. A massa do implante é dada pela multiplicação entre a densidade e o volume, sendo que o volume é dado pela subtração do volume dos furos no volume da casca da calote. O custo e a densidade são propriedades dependentes da variável da escolha de material (x_4), o cálculo do volume da casca da calote requer a utilização da espessura (x_1) e o cálculo do volume dos furos utiliza o raio dos furos (x_2) e o número de furos (x_3). As variáveis escolha do material e número de furos, por serem de natureza discreta, são descritas pelo seu índice, sendo que as correspondências para a escolha do material estão descritas na Tabela 2, Anexo C, ao passo que os índices do número de furos correspondem a 48, 64 e 80, respetivamente. Por seu lado, as restrições têm que ver com a limitação das tensões geradas e do deslocamento de maneira a que sejam inferiores à tensão limite de elasticidade e deslocamento admissível, respetivamente.

A formulação pode ser descrita como

$$
\begin{aligned}
&\text{Procurar } \mathbf{x} \text{ de modo a :} && (1)\\
&\quad \text{minimizar } f(\mathbf{x}) = \text{Custo}(x_4) \times \rho(x_4) \times \pi^2 \\
&\qquad\qquad \times x_2^2 \times x_3 \times \left(\frac{10800x_1^2 + x_1^4}{6000}\right)\\
&\quad \text{sujeito a } g_1(\mathbf{x}) = \sigma_{\max} - \sigma_{\lim}(x_4) < 0,\\
&\qquad\qquad g_2(\mathbf{x}) = U_{\max} - 0{,}35 < 0,\\
&\qquad\qquad 3 \leq x_1 \leq 4{,}5,\\
&\qquad\qquad 1 \leq x_2 \leq 1{,}5 \quad \text{e}\\
&\qquad\qquad 1 \leq x_i \leq 3, \quad i = 3{,}4.
\end{aligned}
$$

O problema é de natureza não-linear com restrições, apresentando um misto de variáveis contínuas e discretas.

3.2 Minimização do material utilizado numa embalagem metálica

O consumo de matéria-prima numa embalagem metálica é minimizado, sujeito a restrições de volume, $g_1(\mathbf{x})$, de resistência mecânica, $g_2(\mathbf{x})$ e de empilhamento.

A restrição de volume é de igualdade, uma vez que representa a capacidade do produto, que para este problema se definiu como vinte litros, por ser a embalagem mais comum no mercado. A tensão máxima a que o produto esta sujeito, $S_{\max}$, é obtida diretamente do software *Abaqus CAE* tendo de ser igual ou inferior à tensão limite elástica do material, de forma a cumprir a restrição $g_2(\mathbf{x})$, que garante a viabilidade física da embalagem quando sujeita a uma força de compressão axial, imposta por um eventual empilhamento, e por uma pressão interna, aplicada pelo fluído a transportar. A restrição de empilhamento é garantida através do domínio de x_i com i=2,..,8. Se cumprido este domínio, existe sempre um raio crescente ao longo da altura, o que permite o empilhamento destas embalagens vazias (situação em que são empilhadas umas no interior das outras).

Este problema contém dez variáveis, sendo estas, o raio inicial (x_1), que define a coordenada x do primeiro ponto da *spline*, os sete incrementos em relação ao ponto anterior (x_{i-1}), que definem as coordenadas x dos restantes pontos da *spline* (x_i, i=2,..,8), a altura, (x_9), com o propósito de definir todas as coordenadas y dos pontos da *spline* através de uma divisão em 9 intervalos equidistantes, e, por último, a espessura do corpo (x_{10}). As variáveis apresentadas encontram-se representadas graficamente na Fig.1(b).

Procurar $\mathbf{x}$ de modo a : (2)

$$\begin{aligned}
\text{minimizar } f(\mathbf{x}) &= \int_0^h \pi((F_1(x) - F_2(x))^2 dx, \\
\text{sujeito a } g_1(\mathbf{x}) &= \int_0^h \pi(F_2(x))^2 dx = 20 \times 10^6, \\
g_2(\mathbf{x}) &= S_{\max} \leq \sigma_{\text{lim,elast}} \\
100 &\leq x_1 \leq 150, \\
0 &\leq x_i \leq 5, \quad i = 2,...,8 \\
300 &\leq x_9 \leq 400 \quad \text{e} \\
0.1 &\leq x_1 0 \leq 1.
\end{aligned}$$

Todas as variáveis envolvidas neste problema são de natureza contínua e trata-se de um problema de programação não-linear.

4 Análise de sensibilidade

4.1 Minimização dos custos de material de um implante cranial

No que concerne à análise de sensibilidade, foi feito um estudo, recorrendo ao *Matlab*, da influência da variação de cada uma das variáveis no valor da função-objetivo. Assim, variando o valor de cada uma das variáveis dentro do seu domínio, é possível tirar conclusões acerca do seu impacto na função-objetivo através do declive das retas, observável no gráfico da Fig. 2(a).

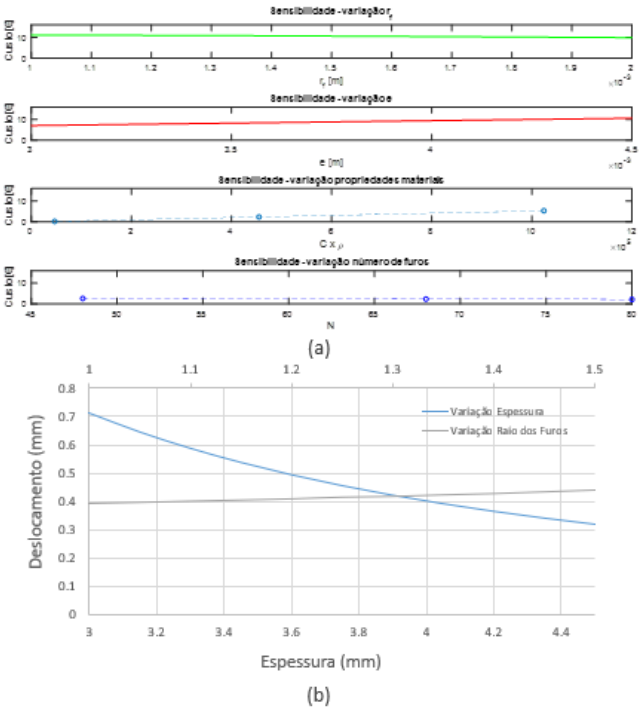

Figura 2.: Análise de sensibilidade do primeiro problema: (a) função-objetivo e (b) restrições.

De seguida procedeu-se a um estudo análogo mas, desta feita, para perceber o impacto das variáveis espessura e raio dos furos no valor da restrição de tensão, cujos resultados estão apresentados na Fig. 2(b). É assim percetível, por leitura das curvas, a não-linearidade do problema na restrição de tensão.

4.2 Minimização do material utilizado numa embalagem metálica

Com o objetivo de obter um maior conhecimento relativamente ao comportamento da função-objetivo e das restrições, realizou-se uma análise de sensibilidade para

cada um destes, usando o mesmo método do primeiro problema.

Começando pela função-objetivo, ao consultar a Fig. 3(a) é de fácil perceção que a espessura é, naturalmente, a variável com o maior impacto, sendo a relação diretamente proporcional. Todas as restantes variáveis têm um impacto consideravelmente menor, porém em nenhuma ocasião a função-objetivo permanece constante.

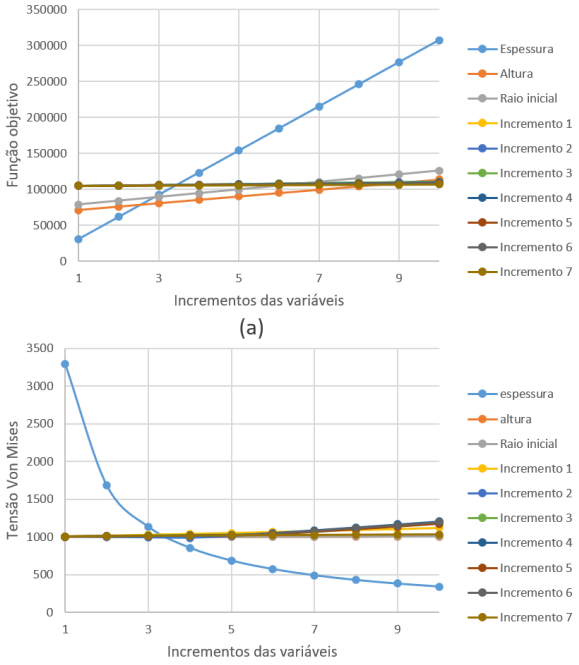

Figura 3.: Análise de sensibilidade do segundo problema: (a) função-objetivo e (b) restrições.

Por uma questão de espaço no presente relatório, optou-se por não representar os dados relativos à restrição de volume, uma vez que, os resultados não apresentaram nenhuma particularidade relevante para o problema em questão.

Por último, analisando a Fig. 3(b), contendo os resultados relativos à restrição g_2(x), é possível concluir que apenas a espessura tem um impacto positivo nesta restrição, uma vez que quanto maior for a espessura menor serão as tensões geradas no produto. Através da curva relativa à espessura é também possível comprovar, de uma forma mais explícita, a não-linearidade do problema.

5 Avaliação

Dada a similaridade dos dois problemas quanto ao sistema de engenharia implementado, nomeadamente na utilização de um software de simulação numérica como programa externo, a avaliação foi idêntica para ambos. Para este efeito foi escolhido o software *Abaqus CAE*.

A interface deste sistema é um *script* realizado recorrendo à linguagem de programação *Python* que, uma vez recebidas as variáveis, as envia para o *Abaqus Python Script* [3], a *blackbox* do sistema, e executa um comando para correr a simulação numérica. Depois deste passo, a interface recolhe os valores das saídas de cada problema, tensões geradas e deslocamentos para o primeiro e tensões geradas e coordenadas do modelo para o segundo, escritos num ficheiro texto pela *blackbox*, e devolve os valores da função-objetivo e das restrições.

6 Algoritmos de otimização

No momento de escolha do algoritmo, teve-se em conta a natureza dos problemas e das variáveis dos mesmos, sendo que se fez um esforço no sentido de encontrar um algoritmo que nos permitisse a aplicação aos dois problemas de forma eficiente e robusta. Assim a escolha recaiu sobre o *Enhanced Firefly Algorithm* [4], uma meta-heurística inspirada na atração entre os pirilampos.

Este algoritmo é uma derivação do *Firefly Algorithm* (FA), adaptado de forma a permitir a introdução de variáveis discretas, existentes no primeiro problema, e de restrições, estas últimas que são aplicadas através de penalização seguindo as três regras de *Deb* [4]. Por sua vez, as variáveis discretas são introduzidas no problema através do arredondamento antes de cada avaliação da função-objetivo. A implementação e descrição do algoritmo podem ser consultadas com maior detalhe no Anexo A, Secção A.2.

7 Resultados

7.1 Minimização do custos de material de um implante cranial

Como última fase da análise aplicou-se o algoritmo ao problema, sendo que a curva média de convergência é apresentada no gráfico da Fig. 4(a). É importante dar nota de que a análise foi efetuada dez vezes para mitigar possíveis erros relacionados com a aleatoriedade do algoritmo.

É observável que, nas primeiras iterações, ocorre uma subida do valor da função-objetivo, que é explicado pelo facto do algoritmo eliminar as soluções que violam as restrições, com recurso às regras de Deb, na fase mais precoce da análise. É também notório que a partir sensivelmente das 150 avaliações há uma convergência clara dos valores da função-objetivo.

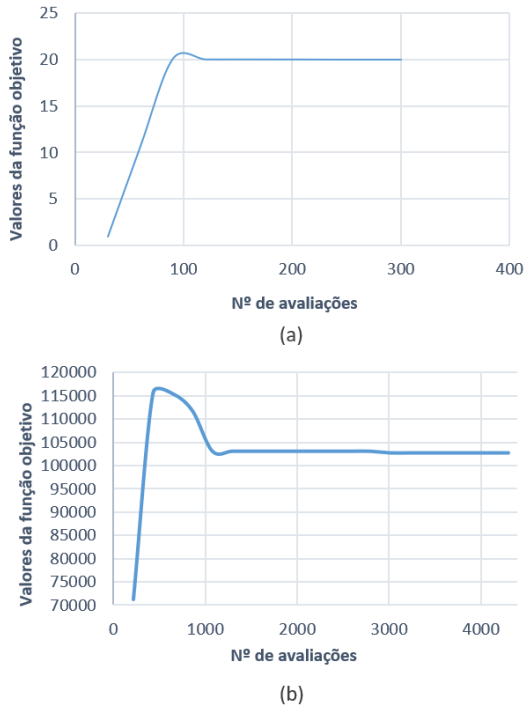

Figura 4.: Curvas médias de convergência: (a) primeiro problema e (b) segundo problema.

Na Fig. 5(a) está apresentado o aspeto da melhor solução obtida. Esta apresenta um custo de 19,9404 €, que corresponde ao material PEEK com uma espessura de 4,0333 milímetros e com 48 furos com um raio de 1,3141 milímetros, ou seja **x**=[4,0333;1,3141;1;1].

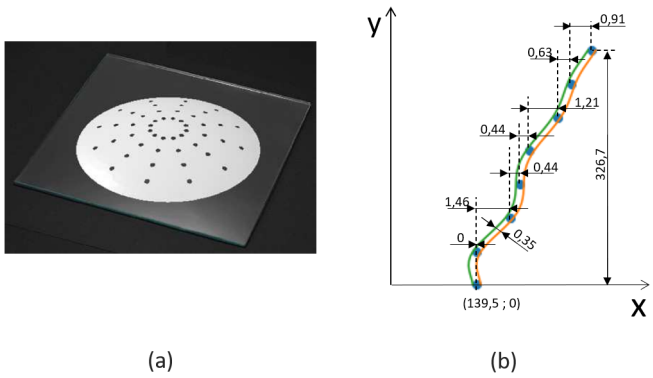

Figura 5.: Representação das soluções ótimas: (a) aspeto do implante cranial correspondente ao melhor resultado obtido e (b) aspeto da parede da embalagem correspondente ao melhor resultado obtido.

7.2 Minimização do material utilizado numa embalagem metálica

Aplicando o algoritmo ao segundo problema e realizando a sua análise dez vezes, para suprimir possíveis erros, obteve-se a curva média de convergência, representada na Fig. 4(b). À imagem do primeiro problema, a subida de valores até sensivelmente às 500 avaliações é devido à natureza do algoritmo, como explicado anteriormente. Analisando o resto da curva é possível concluir que existiu uma convergência.

A solução ótima para este problema corresponde ao menor valor encontrado em todas as simulações realizadas, que foi de 102766 mm^3. Obtiveram-se os valores das variáveis **x**=[139,5 ; 0; 1,46; 0,435; 0,43968; 1,20; 0,626; 0,907; 326,7; 0,347] que se encontram representadas na Fig. 5(b). De referir que esta imagem não se encontra à escala com o intuito de promover uma melhor compreensão e representação das variáveis. Analisando a mesma, é possível perceber que este caso se apresenta como um caso viável para aplicação deste tipo de embalagem por não apresentar variações bruscas de raios. Embora o primeiro incremento (x_2) seja zero, existe uma diminuição do raio neste intervalo devido à interpolação, algo que não inviabiliza o empilhamento, por ser possível a realização do entrave através da restante forma do corpo. Esta figura demonstra também que a variação da forma, ao longo da altura, ajuda de facto na redução do uso de material numa embalagem.

8 Conclusões

Dado por terminado o trabalho, é notória a evolução ao nível do conhecimento na área da otimização, das competências de programação bem como na formulação de um problema de engenharia.

O algoritmo utilizado mostrou-se bastante robusto e com resultados competitivos, como verificado na aplicação ao *benchmark*, revelando-se bastante versátil dadas as diferenças de todos os problemas em análise.

Denota-se que os problemas, no sentido da sua viabilidade prática no mundo real, necessitam de melhorias que não são compatíveis com o enquadramento da Unidade Curricular. Neste campo destaca-se, no primeiro problema, a necessidade de simular numericamente o comportamento do implante enquadrado no modelo da cabeça humana. Relativamente ao segundo, a introdução do sistema de fecho seria preponderante para aumentar o realismo.

A Benchmark 2020

A.1 Enquadramento do benchmark

No âmbito da Unidade Curricular de Otimização Não-Linear em Engenharia foi proposto como *benchmark* o problema de minimização do peso de um redutor de velocidade, um problema de *benchmark* bastante utilizado em otimização.

Este problema concreto surge na sequência da necessidade da utilização de técnicas de otimização aquando do projeto de componentes mecânicos deste tipo tendo em vista a minimização do seu peso, sem que sejam postas em causa a integridade e funcionalidade destes. [5]

No problema há que ter em conta sete variáveis que estão relacionadas com o projeto mecânico deste tipo de componentes, como por exemplo o número de dentes do pinhão (variável inteira), a largura e módulo da roda e diâmetro e comprimento entre rolamentos dos veios (variáveis contínuas). [5]

As restrições do problema centram-se essencialmente com limitação de tensões na roda e limite de tensões e de deflexão nos veios, sendo que as equações que descrevem o *benchmark* (função-objetivo e restrições) estão descritas em [5].

A.2 Implementação

De forma a resolver este problema foi implementado o algoritmo *Enhanced-Firefly*(E-FA), através da linguagem de programação *python*.

Este algoritmo é uma derivação do *Firefly Algorithm* (FA) [4], sendo adaptado de forma a permitir a introdução de variáveis discretas e de restrições, que são aplicadas através de penalização, seguindo as três regras de Deb. Uma outra melhoria introduzida pelo E-FA foi a aplicação de uma redução da progressão geométrica ao fator de escala, o que demonstrou melhorar a qualidade da solução, tal como, a convergência do algoritmo.

Como é possível concluir através do seu nome, as *fireflies* são a inspiração deste algoritmo, mais concretamente a sua reprodução. Este ser vivo uni-sexo é atraído, maioritariamente, através do brilho que outros indivíduos da mesma espécie emitem. Esta relação é a base do E-FA [4].

Sendo o E-FA uma meta-heurística, a sua inicialização é realizada através da obtenção da população inicial, de uma forma aleatória. Cada população, contendo o valor associado a cada uma das variáveis do problema, corresponderá à posição de uma *firefly*.

No mundo das *fireflies*, a mais desejada é aquela que mais brilho tem, sendo que no contexto deste problema, a posição mais desejada é aquela para a qual se obtém o valor mais reduzido da função objectivo. Logicamente, neste algoritmo, o brilho de cada *firefly* corresponderá ao valor da função-objetivo, calculado através da sua posição.

Obtido o brilho para cada *firefly*, é possível a comparação entre as mesmas. Esta comparação é realizada com o intuito de aproximar as *fireflies* menos brilhantes (com pior valor da função-objetivo) das mais brilhantes. Cada *firefly* é então comparada com todas as restantes. Se em alguma destas comparações o seu brilho não for o maior então uma nova posição é calculada para a mesma, tendo em conta a posição da *firefly* mais brilhante, bem como outros parâmetros intrínsecos do algoritmo [4]. Atribuída essa nova posição à *firefly*, é então calculado o seu novo brilho, tendo em conta a sua posição atualizada. Se respeitar as três regras de Deb, a sua posição e brilho são atualizados na população do problema e usado nas seguintes iterações. Se tal não se verificar, então a *firefly* volta à sua posição inicial [4].

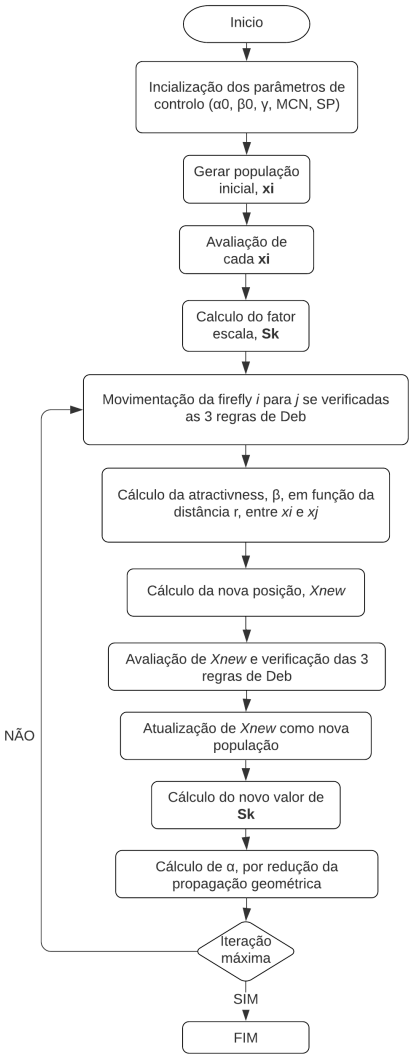

Figura 6.: Fluxograma.

Este processo é repetido um determinado número de iterações com o intuito de chegar à posição ideal para a qual a *firefly* terá o maior brilho possível.

Através do fluxograma representado na Fig. 6 é possível obter uma melhor representação de todo este processo, de uma forma mais pormenorizada.

A.3 Resultados

De seguida, e depois de implementado o algoritmo, foram efetuados diversos testes. Foi tomado como como variante o número de avaliações, com o objetivo de aferir acerca da convergência do valor da função-objetivo em função de parâmetros como o número máximo de iterações e a dimensão da população.

Assim, variando estes parâmetros, avaliamos dez vezes, cada uma das configurações, de modo a evitar um resultado desproporcional, devido à aleatoriedade intrínseca ao algo-

ritmo. Os resultados destas operações estão apresentados na

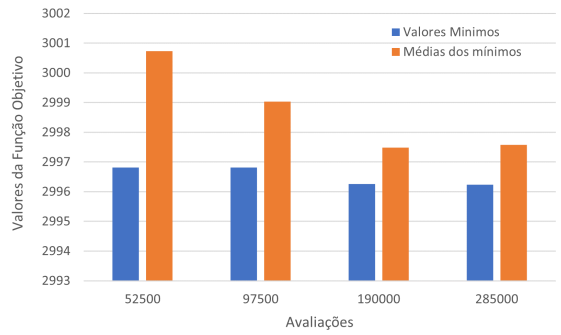

Figura 7.: Histograma de médias e mínimos consoante as avaliações realizadas.

Fig. 7, onde é possível consultar o valor mínimo e a média para cada configuração. Estes resultados mostram claramente uma convergência a partir das 190 mil avaliações para ambos os valores.

Por sua vez, na Fig. 8 é possível observar a curva média de convergência para 285 mil avaliações. Analisando este gráfico, denota-se uma elevada robustez do algoritmo, devido ao facto da curva convergir para um número de avaliações que implicam um baixo custo computacional.

Tabela 1.: Melhor solução obtida.

Mínimo FO	Melhor solução(x_1,x_2,x_3,x_4,x_5,x_6,x_7)
2996,24	(3,50002;0,7;17;7,3;7,8001;3,35023;5,28668)

Por leitura da tabela 1, é observável que o valor mínimo obtido para a função objectivo é idêntico àqueles encontrados em várias soluções presentes na literatura [6]. Importa também realçar que nenhuma destas soluções violam as restrições por terem sido usadas as regras de Deb como critério de seleção.

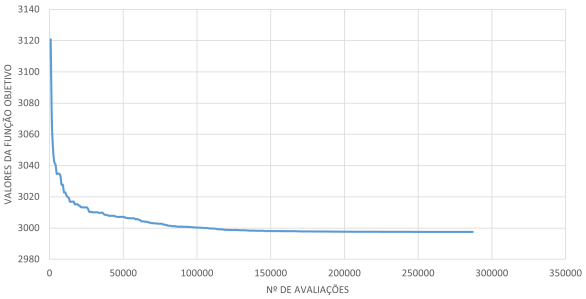

Figura 8.: Gráfico representativo da curva média de convergência.

Referências

1. David J. Bonda; Sunil Manjila; Warren R. Selman; David Dean , *The Recent Revolution in the Design and Manufacture of Cranial Implants: Modern Advancements and Future Directions*, Neurosurgery, Volume 77, Issue 5, Sérbia, 2015.
2. Tsouknidas A.; Maropoulos S.; Savvakis S.; Michailidis N. , *FEM assisted evaluation of PMMA and Ti6Al4V as materials for cranioplasty resulting mechanical behaviour and the neurocranial protection*, Bio-Medical Materials and Engineering, IOS Press, 2011.
3. Gautam Puri, *Python Scripts For Abaqus*, Estados Unidos da América, 2011.
4. I. Brajevic; J. Ignjatovic, *An Enhanced Firefly Algorithm For Mixed Variable Structural Optimization Problems*, Facta Universitatis, Ser. Math. Inform. Vol. 30 , Novembro 2015.
5. A. Andrade-Campos, J. Dias-de-Oliveira, *Benchmark 2020: Weight minimisation of a speed reducer*, Universidade de Aveiro, Aveiro, 2020.
6. S. Elizabeth Amudhini Stephen, Christu Nesam David D., Ajay Joe, *Weight minimization of Speed Reducer design problem using PSO, SA, PS, GODLIKE, CUCKOO, FF, FP, ALO, GSA and MVO*, International Journal Of Scientific Engineering Research, Volume 7, Karunya University, 2016.

B Código em Python do *Enhanced Firefly Algorithm*

```
# -*- coding: utf-8 -*-

#%% Packages
import numpy as np
from math import *
from MainProg import *
# from sphere import *
import matplotlib.pyplot as plt

#%% Parametros de inicializacao

D=4

lb=[3 , 1 , 1 , 1]
ub=[4.5 , 1.5 , 3 , 3]

C=[0,0,1,1] #avaliacao das variaveis (0 se continua e 1 se discreta)
fun=Prog_calote

#%% Parametros do algoritmo

N=7
alpha=0.6
beta0=1
gamma=1
iter_max=10
theta=(1-(10 **(-4)/ 0.9)**(1/iter_max))

# Geracao de populacao inicial

pop=np.zeros((N,D))

for i in range(0 , N):
    for j in range (0 , D):
        pop[i,j]=np.random.uniform(lb[j],ub[j])
        if C[j]==1:
            pop[i,j]=round(pop[i,j])

# avaliacao da funcao-objetivo

fx=np.zeros((N,2))
for i in range(0,N):
    fx[i]=fun(pop[i,:])

Sk=np.zeros((1,D))
for k in range(0,D):
    Sk[0,k]=abs(ub[k]-lb[k])

Bestf=np.zeros((iter_max,1))
BestX=np.zeros((iter_max,D))


cnt=0

# Main EFA Program

for it in range(0 , iter_max):

    for i in range(0,N):

        for j in range(0,N):

            if (fx[i,0]<fx[j,0] and fx[i,1]==0) or (fx[i,1]<fx[j,1] and fx[i,1]!=0 and fx[j,1]!=0) or (fx[i,1]==0 and fx[i,1]!=0) :      # compara o valor calculado antes(que nao muda no for de j) com os valores de j um a um
                pop[i,:]=pop[i,:]  #se o xi for o melhor entao mantem-se como estava...

            else:
                r=0
                for k in range(0,D): # ciclo for para calculo do r, para obter o somatorio k=1:D
                    Xi=pop[i,k]
                    Xj=pop[j,k]
                    r=r+(Xi-Xj)**2
                r=sqrt(r)
                Xnew=np.zeros((1,D))

                for k in range(0,D):
                    Xi=pop[i,k]
                    Xj=pop[j,k]

                    beta = beta0*exp(-gamma*r**2)

                    steps=alpha*(np.random.uniform(0,1)-0.5)*Sk[0,k]

                    Xnew[0,k]=Xi+beta*(Xj-Xi)+steps

                    if Xnew[0,k]<lb[k]:         # Verificar os limites

                        Xnew[0,k]=lb[k]

                    if Xnew[0,k]>ub[k]:

                        Xnew[0,k]=ub[k]

                    if C[k]==1:
                        Xnew[0,k]=round(Xnew[0,k])

                fnew=np.zeros((1,2))
                fnew[0,:]=fun(Xnew[0,:])
                cnt=cnt+1

                if (fnew[0,0]<fx[i,0] and fnew[0,1]==0) or (fx[i,1]!=0 and fnew[0,1]!=0 and fnew[0,1]<fx[i,1]) or (fnew[0,1]==0 and fx[i,1]!=0):

                    fx[i,:]=fnew[0,:]
                    pop[i,:]=Xnew

    for k in range(0,D):
        Sk[0,k]=Sk[0,k]*theta**(1/iter_max)

    alpha=alpha*theta**(1/iter_max)

    fbest=min(fx[:,0])
    fx_list=fx[:,0].tolist()

    Xbest=fx_list.index(fbest)


    Bestf[it]=fbest
    BestX[it]=pop[Xbest,:]
```

C Informação adicional do problema de minimização dos custos de material de um implante cranial obtido por impressão 3D

```
# -*- coding: utf-8 -*-

#%% Packages
from math import *
import numpy as np
import re

# Funcao replace para substituir valores das variaveis no script abaqus
def replace(file, pattern, subst):
    # Read contents from file as a single string
    file_handle = open(file, 'r')
    file_string = file_handle.read()
    file_handle.close()

    # Use RE package to allow for replacement (also allowing for (multiline) REGEX)
    file_string = (re.sub(pattern, subst, file_string))

    # Write contents to file.
    # Using mode 'w' truncates the file.
    file_handle = open(file, 'w')
    file_handle.write(file_string)
    file_handle.close()

# Main Program
def Prog_calote(X):

    # Atribuicao das variaveis

    espe=X[0] #espessura

    rf=X[1] #raio dos furos

    if X[2]==1: #num furos
        N=48
    elif X[2]==2:
        N=64
    elif X[2]==3:
        N=80

    if X[3]==1: #material_peek
        E=3500
        cp=0.38
        p=1300
        tens_lim=100
        C=700
    elif X[3]==2: #material_pmma
        E=2900
        cp=0.36
        p=1170
        tens_lim=70
        C=40
    elif X[3]==3: #material_pekk
        E=3500
        cp=0.40
        p=1280
        tens_lim=110
        C=800

    # definicao do angulo do posicionamento dos furos

    alpha=1440/N

    # envio dos dados para o programa AbaqExec (funcao replace)

    variableE = """E=%s""" % (E)  #para E
    replace("C:/Users/Pedro/Desktop/Universidade/5ano/1semestre/ONLE/Algoritmos/AbaqExec.py", "E=[0-9]*" , variableE)

    variablecp = """cp=%s""" % (cp) #para cp
    replace("C:/Users/Pedro/Desktop/Universidade/5ano/1semestre/ONLE/Algoritmos/AbaqExec.py", "cp=[0-9].+" , variablecp)

    variableespe = """espe=%s""" % (espe) #para es (inteiro)
    replace("C:/Users/Pedro/Desktop/Universidade/5ano/1semestre/ONLE/Algoritmos/AbaqExec.py", "espe=[0-9]" , variableespe)

    variableespe = """espe=%s""" % (espe) #para es (decimal)
    replace("C:/Users/Pedro/Desktop/Universidade/5ano/1semestre/ONLE/Algoritmos/AbaqExec.py", "espe=[0-9].+" , variableespe)

    variablerf = """rf=%s""" % (rf) #para rf (inteiro)
    replace("C:/Users/Pedro/Desktop/Universidade/5ano/1semestre/ONLE/Algoritmos/AbaqExec.py", "rf=[0-9]" , variablerf)

    variablerf = """rf=%s""" % (rf) #para rf (decimal)
    replace("C:/Users/Pedro/Desktop/Universidade/5ano/1semestre/ONLE/Algoritmos/AbaqExec.py", "rf=[0-9].+" , variablerf)

    variablealpha = """alpha=%s""" % (alpha) #para alpha (inteiro)
    replace("C:/Users/Pedro/Desktop/Universidade/5ano/1semestre/ONLE/Algoritmos/AbaqExec.py", "alpha=[0-9]" , variablealpha)

    variablealpha = """alpha=%s""" % (alpha) #para alpha (decimal)
    replace("C:/Users/Pedro/Desktop/Universidade/5ano/1semestre/ONLE/Algoritmos/AbaqExec.py", "alpha=[0-9].+" , variablealpha)

    # definir dados

    r0=0.05
    h=0.01

    # funcao objetivo

    fobj=C*p*(((1/6)*pi*h*(3*(r0+(espe*10**-3))**2+h**2))-((1/6)*pi*(h-(espe*10**-3))*(3*r0**2+(h-(espe*10**-3))**2))-(N*(pi*(rf*10**-3)**2*(espe*10**-3))))

    # execucao abaqus script

    import os
    if os.path.exists("sim.lck"):
        os.remove("sim.lck")

    os.system('abq2019se cae noGUI="AbaqExec.py"')

    # ler valores maximos de tensao e deslocamento

    S = np.genfromtxt('sim_Stresses.txt', unpack=True)
    S_max=max(S)

    D = np.genfromtxt('sim_Displacements.txt', unpack=True)
    D_max=abs(max(D))

    #violacao das restricoes

    g=np.zeros((2,1))
    g[0]=S_max-tens_lim
    g[1]=D_max-0.39
    CV=0
    for i in range(0,2):
        CV=CV+max(0,g[i])

    # return
    return fobj,CV
```

Tabela 2.: Materiais e propriedades para o problema de minimização de custos de material de um implante cranial.

Índice	Material	Custo (€/kg)	ρ(kg/m^3)	E (GPa)	ν
1	PEEK	700	1300	3,5	0,38
2	PMMA	40	1170	2,9	0,36
3	PEKK	800	1280	3,5	0,4

D Código de interação com o Abaqus

```
# -*- coding: utf-8 -*-


#%% VARIABLES SET (REPLACE FUNCTION)
E=3500
cp=0.4
espe=4
rf=1.25
alpha=22.5


#%%
# -*- coding: mbcs -*-
# Do not delete the following import lines

from abaqus import *
from abaqusConstants import *
import __main__
from odbAccess import *
from types import IntType
import numpy as np

import section
import regionToolset
import displayGroupMdbToolset as dgm
import part
import material
import assembly
import step
import interaction
import load
import mesh
import optimization
import job
import sketch
import visualization
import xyPlot
import displayGroupOdbToolset as dgo
import connectorBehavior

# MACRO_INITIALIZE #
s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
    sheetSize=200.0)
g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
s.setPrimaryObject(option=STANDALONE)
s.ConstructionLine(point1=(0.0, -100.0), point2=(0.0, 100.0))
s.FixedConstraint(entity=g[2])
s.ArcByCenterEnds(center=(0.0, 0.0), point1=(46.25, 0.0), point2=(0.0, 42.5),
    direction=COUNTERCLOCKWISE)
s.CoincidentConstraint(entity1=v[2], entity2=g[2], addUndoState=False)
s.CoincidentConstraint(entity1=v[1], entity2=g[2], addUndoState=False)
session.viewports['Viewport: 1'].view.setValues(nearPlane=37.9677,
    farPlane=339.156, width=1159.07, height=575.769, cameraPosition=(
    -39.926, -213.521, 188.562), cameraTarget=(-39.926, -213.521, 0))
s.RadialDimension(curve=g[3], textPoint=(25.3269462585449, 73.2544250488281),
    radius=130.0)
s.Line(point1=(0.0, 95.0), point2=(88.7411967464723, 95.0))
s.HorizontalConstraint(entity=g[4], addUndoState=False)
s.PerpendicularConstraint(entity1=g[2], entity2=g[4], addUndoState=False)
s.CoincidentConstraint(entity1=v[3], entity2=g[2], addUndoState=False)
s.CoincidentConstraint(entity1=v[4], entity2=g[3], addUndoState=False)
s.VerticalDimension(vertex1=v[1], vertex2=v[3], textPoint=(-11.9621467590332,
    111.012710571289), value=10.0)
s.autoTrimCurve(curve1=g[3], point1=(75.8763427734375, 104.135955810547))
s.autoTrimCurve(curve1=g[4], point1=(24.8871574401855, 118.749069213867))
p = mdb.models['Model-1'].Part(name='calote', dimensionality=THREE_D,
    type=DEFORMABLE_BODY)
p = mdb.models['Model-1'].parts['calote']
p.BaseShellRevolve(sketch=s, angle=90.0, flipRevolveDirection=OFF)
s.unsetPrimaryObject()
p = mdb.models['Model-1'].parts['calote']
session.viewports['Viewport: 1'].setValues(displayedObject=p)
del mdb.models['Model-1'].sketches['__profile__']
session.viewports['Viewport: 1'].partDisplay.setValues(sectionAssignments=ON,
    engineeringFeatures=ON)
session.viewports['Viewport: 1'].partDisplay.geometryOptions.setValues(
    referenceRepresentation=OFF)
mdb.models['Model-1'].Material(name='Material')
mdb.models['Model-1'].materials['Material'].Elastic(table=((E, cp), ))
mdb.models['Model-1'].HomogeneousShellSection(name='Section-1',
    preIntegrate=OFF, material='Material', thicknessType=UNIFORM,
    thickness=espe, thicknessField='', nodalThicknessField='',
    idealization=NO_IDEALIZATION, poissonDefinition=DEFAULT,
    thicknessModulus=None, temperature=GRADIENT, useDensity=OFF,
    integrationRule=SIMPSON, numIntPts=5)
p = mdb.models['Model-1'].parts['calote']
f = p.faces
faces = f.getSequenceFromMask(mask=('[#1 ]', ), )
region = p.Set(faces=faces, name='Set-1')
p = mdb.models['Model-1'].parts['calote']
p.SectionAssignment(region=region, sectionName='Section-1', offset=0.0,
    offsetType=MIDDLE_SURFACE, offsetField='',
    thicknessAssignment=FROM_SECTION)
a = mdb.models['Model-1'].rootAssembly
session.viewports['Viewport: 1'].setValues(displayedObject=a)
session.viewports['Viewport: 1'].assemblyDisplay.setValues(
    optimizationTasks=OFF, geometricRestrictions=OFF,
    stopConditions=OFF)
a = mdb.models['Model-1'].rootAssembly
a.DatumCsysByDefault(CARTESIAN)
p = mdb.models['Model-1'].parts['calote']
a.Instance(name='calote-1', part=p, dependent=ON)
session.viewports['Viewport: 1'].assemblyDisplay.setValues(
    adaptiveMeshConstraints=ON)
mdb.models['Model-1'].StaticStep(name='load', previous='Initial',
    maxNumInc=1000, initialInc=0.001, minInc=1e-10, maxInc=0.01)
session.viewports['Viewport: 1'].assemblyDisplay.setValues(step='load')
session.viewports['Viewport: 1'].assemblyDisplay.setValues(loads=ON, bcs=ON,
    predefinedFields=ON, connectors=ON, adaptiveMeshConstraints=OFF)
a = mdb.models['Model-1'].rootAssembly
v1 = a.instances['calote-1'].vertices
verts1 = v1.getSequenceFromMask(mask=('[#1 ]', ), )
region = a.Set(vertices=verts1, name='Set-1')
mdb.models['Model-1'].ConcentratedForce(name='load', createStepName='load',
    region=region, cf2=-100.0, distributionType=UNIFORM, field='',
    localCsys=None)
a = mdb.models['Model-1'].rootAssembly
e1 = a.instances['calote-1'].edges
edges1 = e1.getSequenceFromMask(mask=('[#2 ]', ), )
region = a.Set(edges=edges1, name='Set-2')
```

```
mdb.models['Model-1'].EncastreBC(name='encastre', createStepName='load',
    region=region, localCsys=None)
a = mdb.models['Model-1'].rootAssembly
e1 = a.instances['calote-1'].edges
edges1 = e1.getSequenceFromMask(mask=('[#1 ]', ), )
region = a.Set(edges=edges1, name='Set-3')
mdb.models['Model-1'].XsymmBC(name='sim_x', createStepName='load',
    region=region, localCsys=None)
a = mdb.models['Model-1'].rootAssembly
e1 = a.instances['calote-1'].edges
edges1 = e1.getSequenceFromMask(mask=('[#4 ]', ), )
region = a.Set(edges=edges1, name='Set-4')
mdb.models['Model-1'].ZsymmBC(name='sim_z', createStepName='load',
    region=region, localCsys=None)
session.viewports['Viewport: 1'].partDisplay.setValues(sectionAssignments=OFF,
    engineeringFeatures=OFF)
session.viewports['Viewport: 1'].partDisplay.geometryOptions.setValues(
    referenceRepresentation=ON)
p = mdb.models['Model-1'].parts['calote']
session.viewports['Viewport: 1'].setValues(displayedObject=p)
p = mdb.models['Model-1'].parts['calote']
v1, d1 = p.vertices, p.datums
p.DatumPlaneByPointNormal(point=v1[0], normal=d1[1])
p = mdb.models['Model-1'].parts['calote']
e, d2 = p.edges, p.datums
t = p.MakeSketchTransform(sketchPlane=d2[3], sketchUpEdge=e[2],
    sketchPlaneSide=SIDE1, sketchOrientation=RIGHT, origin=(25.0, 130.0,
    25.0))
s1 = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
    sheetSize=304.7, gridSpacing=7.61, transform=t)
g, v, d, c = s1.geometry, s1.vertices, s1.dimensions, s1.constraints
s1.setPrimaryObject(option=SUPERIMPOSE)
p = mdb.models['Model-1'].parts['calote']
p.projectReferencesOntoSketch(sketch=s1, filter=COPLANAR_EDGES)
p = mdb.models['Model-1'].parts['calote']
e1, v2, n, p1 = p.edges, p.vertices, p.nodes, p.elemEdges
p.projectReferencesOntoSketch(sketch=s1, vertices=(v2[0], v2[1], v2[2]))
s1.ConstructionLine(point1=(25.0, -25.0), point2=(25.0, 25.0))
s1.VerticalConstraint(entity=g[2], addUndoState=False)
s1.CoincidentConstraint(entity1=v[0], entity2=g[2], addUndoState=False)
s1.CoincidentConstraint(entity1=v[2], entity2=g[2], addUndoState=False)
s1.ConstructionLine(point1=(25.0, -25.0), point2=(11.415, 11.415))
s1.CoincidentConstraint(entity1=v[0], entity2=g[3], addUndoState=False)
s1.ConstructionLine(point1=(25.0, -25.0), point2=(7.61, -3.805))
s1.CoincidentConstraint(entity1=v[0], entity2=g[4], addUndoState=False)
s1.ConstructionLine(point1=(25.0, -25.0), point2=(1.9025, -11.415))
s1.CoincidentConstraint(entity1=v[0], entity2=g[5], addUndoState=False)
s1.ConstructionLine(point1=(25.0, -25.0), point2=(-3.805, -19.025))
s1.CoincidentConstraint(entity1=v[0], entity2=g[6], addUndoState=False)
s1.AngularDimension(line1=g[2], line2=g[3], textPoint=(19.0800857543945,
    8.73482513427734), value=alpha)
s1.AngularDimension(line1=g[2], line2=g[4], textPoint=(6.21212005615234,
    12.8518753051758), value=2*alpha)
s1.AngularDimension(line1=g[2], line2=g[5], textPoint=(-8.31980133056641,
    6.2868480682373), value=3*alpha)
s1.AngularDimension(line1=g[2], line2=g[6], textPoint=(-22.7407913208008,
    -3.28251457214355), value=4*alpha)
s1.CircleByCenterPerimeter(center=(21.633239056915, -14.6381752723828),
    point1=(21.4117705083918, -13.9565651658922))
s1.CoincidentConstraint(entity1=v[4], entity2=g[3], addUndoState=False)
s1.CoincidentConstraint(entity1=v[3], entity2=g[3], addUndoState=False)
s1.CircleByCenterPerimeter(center=(18.1133270383443, -3.80499999985099),
    point1=(16.7361888885498, -4.89089393615723))
s1.CoincidentConstraint(entity1=v[5], entity2=g[3], addUndoState=False)
session.viewports['Viewport: 1'].view.setValues(nearPlane=141.494,
    farPlane=173.314, width=61.7829, height=30.6907, cameraPosition=(
    18.2673, -32.4039, 7.51584), cameraTarget=(18.2673, 125, 7.51584))
s1.CircleByCenterPerimeter(center=(15.6406881501734, 3.80499999985099),
    point1=(17.1893949508667, 4.70979690551758))
s1.CoincidentConstraint(entity1=v[7], entity2=g[3], addUndoState=False)
s1.CircleByCenterPerimeter(center=(12.5498895396472, 13.3175000004098),
    point1=(13.1680492617716, 11.4150000000186))
s1.CoincidentConstraint(entity1=v[10], entity2=g[3], addUndoState=False)
s1.CoincidentConstraint(entity1=v[9], entity2=g[3], addUndoState=False)
s1.ObliqueDimension(vertex1=v[3], vertex2=v[0], textPoint=(29.1761212348938,
    -13.7511463165283), value=10.0)
s1.ObliqueDimension(vertex1=v[0], vertex2=v[5], textPoint=(32.8601307868958,
    -2.30461311340332), value=20.0)
s1.ObliqueDimension(vertex1=v[0], vertex2=v[7], textPoint=(39.9586029052734,
    6.43801307678223), value=30.0)
s1.ObliqueDimension(vertex1=v[0], vertex2=v[9], textPoint=(44.6310119628906,
    14.5497283935547), value=40.0)
s1.CircleByCenterPerimeter(center=(19.0250000000699, -16.7761180250714),
    point1=(18.0166087150574, -17.5534858703613))
s1.CoincidentConstraint(entity1=v[11], entity2=g[4], addUndoState=False)
s1.CircleByCenterPerimeter(center=(12.3654254382041, -7.61000000016764),
    point1=(11.415, -9.5125))
s1.CoincidentConstraint(entity1=v[13], entity2=g[4], addUndoState=False)
s1.CircleByCenterPerimeter(center=(6.22185470885597, 0.845899678766727),
    point1=(7.61, 1.9025))
s1.CoincidentConstraint(entity1=v[15], entity2=g[4], addUndoState=False)
s1.CircleByCenterPerimeter(center=(0.698462271830067, 8.44819716876373),
    point1=(1.47795753879473, 7.37531397631392))
s1.CoincidentConstraint(entity1=v[18], entity2=g[4], addUndoState=False)
s1.CoincidentConstraint(entity1=v[17], entity2=g[4], addUndoState=False)
s1.ObliqueDimension(vertex1=v[0], vertex2=v[11], textPoint=(28.287166595459,
    -15.2898788452148), value=10.0)
s1.ObliqueDimension(vertex1=v[0], vertex2=v[13], textPoint=(25.0,
    -4.13840293884277), value=20.0)
s1.ObliqueDimension(vertex1=v[0], vertex2=v[15], textPoint=(32.695143699646,
    3.48172950744629), value=30.0)
s1.ObliqueDimension(vertex1=v[0], vertex2=v[17], textPoint=(22.4601926803589,
    5.12942314147949), value=40.0)
s1.CircleByCenterPerimeter(center=(16.7761180237942, -19.0249999994878),
```

```
    point1=(17.8101156218909, -19.7762432279997))
s1.CoincidentConstraint(entity1=v[20], entity2=g[5],
    addUndoState=False)
s1.CoincidentConstraint(entity1=v[19], entity2=g[5],
    addUndoState=False)
s1.CircleByCenterPerimeter(center=(7.60999999970198,
    -12.3654254375163),
    point1=(9.5125, -11.415))
s1.CoincidentConstraint(entity1=v[21], entity2=g[5],
    addUndoState=False)
s1.CircleByCenterPerimeter(center=(-3.80500000054948,
    -4.07194248002406),
    point1=(-1.99626743979752, -5.3860636074096))
s1.CoincidentConstraint(entity1=v[24], entity2=g[5],
    addUndoState=False)
s1.CoincidentConstraint(entity1=v[23], entity2=g[5],
    addUndoState=False)
s1.CircleByCenterPerimeter(center=(-9.40954801243743,
    0.0), point1=(
    -7.95791382482275, -1.05467397184111))
s1.CoincidentConstraint(entity1=v[26], entity2=g[5],
    addUndoState=False)
s1.CoincidentConstraint(entity1=v[25], entity2=g[5],
    addUndoState=False)
s1.ObliqueDimension(vertex1=v[19], vertex2=v[0],
    textPoint=(15.1891288757324,
    -28.4796485900879), value=10.0)
s1.ObliqueDimension(vertex1=v[21], vertex2=v[0],
    textPoint=(5.95965576171875,
    -36.4381408691406), value=20.0)
s1.ObliqueDimension(vertex1=v[23], vertex2=v[0],
    textPoint=(-6.83206558227539,
    -40.9858512878418), value=30.0)
s1.ObliqueDimension(vertex1=v[0], vertex2=v[25],
    textPoint=(-14.766170501709,
    -48.6195068359375), value=40.0)
s1.CircleByCenterPerimeter(center=(15.2199999998696,
    -21.8222853707427),
    point1=(16.2730250661261, -22.1644339554477))
s1.CoincidentConstraint(entity1=v[28], entity2=g[6],
    addUndoState=False)
s1.CoincidentConstraint(entity1=v[27], entity2=g[6],
    addUndoState=False)
s1.CircleByCenterPerimeter(center=(5.70749999977648,
    -18.7314867602443),
    point1=(6.61084086137612, -19.0249999994878))
s1.CoincidentConstraint(entity1=v[30], entity2=g[6],
    addUndoState=False)
s1.CoincidentConstraint(entity1=v[29], entity2=g[6],
    addUndoState=False)
s1.CircleByCenterPerimeter(center=(-3.80500000008382,
    -15.6406881499931),
    point1=(-3.805, -17.1225))
s1.CoincidentConstraint(entity1=v[31], entity2=g[6],
    addUndoState=False)
s1.CircleByCenterPerimeter(center=(-13.317500000177,
    -12.5498895394815),
    point1=(-11.4150000004843, -13.1680492614364))
s1.CoincidentConstraint(entity1=v[34], entity2=g[6],
    addUndoState=False)
s1.CoincidentConstraint(entity1=v[33], entity2=g[6],
    addUndoState=False)
s1.ObliqueDimension(vertex1=v[27], vertex2=v[0],
    textPoint=(9.49579811096191,
    -28.9665288925171), value=10.0)
s1.ObliqueDimension(vertex1=v[0], vertex2=v[29],
    textPoint=(5.36438941955566,
    -30.1505575180054), value=20.0)
s1.ObliqueDimension(vertex1=v[0], vertex2=v[31],
    textPoint=(-7.38395690917969,
    -38.2019643783569), value=30.0)
s1.ObliqueDimension(vertex1=v[0], vertex2=v[33],
    textPoint=(-19.8962173461914,
    -35.9523077011108), value=40.0)
s1.RadialDimension(curve=g[7], textPoint
    =(23.0030345693617, -16.673772894563),
    radius=rf)
s1.RadialDimension(curve=g[11], textPoint
    =(21.4197564125061,
    -18.1202735900879), radius=rf)
s1.RadialDimension(curve=g[15], textPoint
    =(18.9651074409485,
    -19.6073408126831), radius=rf)
s1.RadialDimension(curve=g[19], textPoint
    =(17.968665599823, -21.8013746738434),
    radius=rf)
s1.RadialDimension(curve=g[8], textPoint
    =(21.7615938186646, -8.03036880493164),
    radius=rf)
s1.RadialDimension(curve=g[12], textPoint
    =(17.9099292755127,
    -10.8055610656738), radius=rf)
s1.RadialDimension(curve=g[16], textPoint
    =(13.515775680542, -14.1793241500854),
    radius=rf)
s1.RadialDimension(curve=g[20], textPoint
    =(6.97856140136719, -20.884250164032),
    radius=rf)
s1.RadialDimension(curve=g[9], textPoint
    =(19.6680107116699, 1.16119384765625),
    radius=rf)
s1.RadialDimension(curve=g[13], textPoint
    =(12.8869018554688,
    -2.83495330810547), radius=rf)
s1.RadialDimension(curve=g[17], textPoint
    =(3.04802703857422,
    -13.7182102203369), radius=rf)
s1.RadialDimension(curve=g[21], textPoint
    =(-1.52922248840332,
    -19.4148464202881), radius=rf)
s1.RadialDimension(curve=g[22], textPoint
    =(-11.2770690917969,
    -17.289234161377), radius=rf)
s1.RadialDimension(curve=g[18], textPoint
    =(-6.95410919189453,
    -6.74620819091797), radius=rf)
s1.RadialDimension(curve=g[14], textPoint
    =(1.44103622436523, 7.26113510131836),
    radius=rf)
session.viewports['Viewport: 1'].view.setValues(
    nearPlane=141.332,
    farPlane=173.476, width=77.5225, height=38.5094,
    cameraPosition=(
    20.0582, -32.4039, 16.5563), cameraTarget
    =(20.0582, 125, 16.5563))
s1.RadialDimension(curve=g[10], textPoint
    =(9.4793529510498, 7.2639274597168),
    radius=rf)
s1.CircleByCenterPerimeter(center=(25.0,
    -15.2199999998696), point1=(
    25.8699359893799, -14.476996421814))
s1.CoincidentConstraint(entity1=v[35], entity2=g[2],
    addUndoState=False)
s1.CircleByCenterPerimeter(center=(25.0,
    -5.70749999977648), point1=(25.0,
    -4.71339797973633))
s1.CoincidentConstraint(entity1=v[38], entity2=g[2],
    addUndoState=False)
s1.CoincidentConstraint(entity1=v[37], entity2=g[2],
    addUndoState=False)
s1.CircleByCenterPerimeter(center=(25.0,
    7.61000000016764), point1=(25.0,
    9.32235717773438))
s1.CoincidentConstraint(entity1=v[40], entity2=g[2],
    addUndoState=False)
s1.CoincidentConstraint(entity1=v[39], entity2=g[2],
    addUndoState=False)
s1.CircleByCenterPerimeter(center=(25.0,
    17.1224999997951), point1=(25.0,
    18.944465637207))
s1.CoincidentConstraint(entity1=v[42], entity2=g[2],
    addUndoState=False)
s1.CoincidentConstraint(entity1=v[41], entity2=g[2],
    addUndoState=False)
s1.RadialDimension(curve=g[23], textPoint
    =(26.8801689147949,
    -17.6498184204102), radius=rf)
s1.RadialDimension(curve=g[24], textPoint
    =(26.3674001693726,
    -8.23289489746094), radius=rf)
s1.RadialDimension(curve=g[25], textPoint
    =(26.3477714061737, 3.89763450622559),
    radius=rf)
```

```
s1.RadialDimension(curve=g[26], textPoint
    =(27.5652561187744, 19.7017517089844),
    radius=rf)
s1.VerticalDimension(vertex1=v[35], vertex2=v[0],
    textPoint=(26.7524995803833,
    -21.670193195343), value=10.0)
s1.VerticalDimension(vertex1=v[0], vertex2=v[37],
    textPoint=(34.8119440078735,
    -15.656364440918), value=20.0)
s1.VerticalDimension(vertex1=v[40], vertex2=v[0],
    textPoint=(43.7215194702148,
    -20.3767070770264), value=30.0)
s1.VerticalDimension(vertex1=v[41], vertex2=v[0],
    textPoint=(71.5422286987305,
    -17.452127456665), value=40.0)
s1.ConstructionLine(point1=(25.0, -25.0), point2
    =(-34.245, -19.025))
s1.CoincidentConstraint(entity1=v[0], entity2=g[27],
    addUndoState=False)
s1.AngularDimension(line1=g[2], line2=g[27],
    textPoint=(-55.8354339599609,
    -4.4422607421875), value=5*alpha)
s1.CircleByCenterPerimeter(center=(15.2199999998696,
    -25.0), point1=(
        15.9274883270264, -25.0))
s1.CoincidentConstraint(entity1=v[44], entity2=g[27],
    addUndoState=False)
s1.CoincidentConstraint(entity1=v[43], entity2=g[27],
    addUndoState=False)
s1.CircleByCenterPerimeter(center=(5.70749999977648,
    -25.0), point1=(
    6.6686840057373, -23.5239441394806))
s1.CoincidentConstraint(entity1=v[45], entity2=g[27],
    addUndoState=False)
s1.CircleByCenterPerimeter(center=(-5.70749999977648,
    -25.0), point1=(
    -4.69608116149902, -25.0))
s1.CoincidentConstraint(entity1=v[48], entity2=g[27],
    addUndoState=False)
s1.CoincidentConstraint(entity1=v[47], entity2=g[27],
    addUndoState=False)
s1.CircleByCenterPerimeter(center=(-15.2199999998696,
    -25.0), point1=(
    -13.7825050354004, -25.0))
s1.CoincidentConstraint(entity1=v[50], entity2=g[27],
    addUndoState=False)
s1.CoincidentConstraint(entity1=v[49], entity2=g[27],
    addUndoState=False)
s1.RadialDimension(curve=g[31], textPoint
    =(-13.6495132446289,
    -19.1148881912231), radius=rf)
s1.RadialDimension(curve=g[30], textPoint
    =(-5.80292510986328,
    -31.0965967178345), radius=rf)
s1.RadialDimension(curve=g[29], textPoint
    =(8.69330978393555,
    -29.6321649551392), radius=rf)
s1.RadialDimension(curve=g[28], textPoint
    =(15.8320503234863,
    -26.9058499336243), radius=rf)
s1.ObliqueDimension(vertex1=v[43], vertex2=v[0],
    textPoint=(
    22.3124756813049, -27.8195266723633), value=10.0)
s1.ObliqueDimension(vertex1=v[0], vertex2=v[45],
    textPoint=(
    13.5630264282227, -31.4871168136597), value=20.0)
s1.ObliqueDimension(vertex1=v[0], vertex2=v[47],
    textPoint=(
    5.53610801696777, -39.5342473983765), value=30.0)
s1.ObliqueDimension(vertex1=v[0], vertex2=v[49],
    textPoint=(
    -6.37760353088379, -46.8363189697266), value
    =40.0)
p = mdb.models['Model-1'].parts['calote']
e, d1 = p.edges, p.datums
p.CutExtrude(sketchPlane=d1[3], sketchUpEdge=e[2],
    sketchPlaneSide=SIDE1,
    sketchOrientation=RIGHT, sketch=s1,
    flipExtrudeDirection=ON)
s1.unsetPrimaryObject()
del mdb.models['Model-1'].sketches['__profile__']
session.viewports['Viewport: 1'].view.setValues(
    nearPlane=102.21,
    farPlane=205.589, width=124.963, height=55.3777,
    viewOffsetX=5.30595,
    viewOffsetY=-5.11455)
a1 = mdb.models['Model-1'].rootAssembly
a1.regenerate()
a = mdb.models['Model-1'].rootAssembly
session.viewports['Viewport: 1'].setValues(
    displayedObject=a)
p = mdb.models['Model-1'].parts['calote']
session.viewports['Viewport: 1'].setValues(
    displayedObject=p)
session.viewports['Viewport: 1'].partDisplay.
    setValues(mesh=ON)
session.viewports['Viewport: 1'].partDisplay.
    meshOptions.setValues(
    meshTechnique=ON)
session.viewports['Viewport: 1'].partDisplay.
    geometryOptions.setValues(
    referenceRepresentation=OFF)
elemType1 = mesh.ElemType(elemCode=S4R, elemLibrary=
    STANDARD,
    secondOrderAccuracy=OFF, hourglassControl=DEFAULT
    )
elemType2 = mesh.ElemType(elemCode=S3, elemLibrary=
    STANDARD)
p = mdb.models['Model-1'].parts['calote']
f = p.faces
faces = f.getSequenceFromMask(mask=('[#1 ]', ), )
pickedRegions =(faces, )
p.setElementType(regions=pickedRegions, elemTypes=(
    elemType1, elemType2))
elemType1 = mesh.ElemType(elemCode=S4R, elemLibrary=
    STANDARD,
    secondOrderAccuracy=OFF, hourglassControl=DEFAULT
    )
elemType2 = mesh.ElemType(elemCode=S3, elemLibrary=
    STANDARD)
p = mdb.models['Model-1'].parts['calote']
f = p.faces
faces = f.getSequenceFromMask(mask=('[#1 ]', ), )
pickedRegions =(faces, )
p.setElementType(regions=pickedRegions, elemTypes=(
    elemType1, elemType2))
p = mdb.models['Model-1'].parts['calote']
f = p.faces
pickedRegions = f.getSequenceFromMask(mask=('[#1 ]',
    ), )
p.setMeshControls(regions=pickedRegions, elemShape=
    QUAD, algorithm=MEDIAL_AXIS)
p = mdb.models['Model-1'].parts['calote']
p.seedPart(size=17.0, deviationFactor=0.1,
    minSizeFactor=0.1)
p = mdb.models['Model-1'].parts['calote']
p.generateMesh()
a = mdb.models['Model-1'].rootAssembly
a.regenerate()
session.viewports['Viewport: 1'].setValues(
    displayedObject=a)
session.viewports['Viewport: 1'].assemblyDisplay.
    setValues(mesh=OFF)
session.viewports['Viewport: 1'].assemblyDisplay.
    meshOptions.setValues(
    meshTechnique=OFF)
mdb.Job(name='sim', model='Model-1', description='',
    type=ANALYSIS,
    atTime=None, waitMinutes=0, waitHours=0, queue=
    None, memory=90,
    memoryUnits=PERCENTAGE, getMemoryFromAnalysis=
    True,
    explicitPrecision=SINGLE, nodalOutputPrecision=
    SINGLE, echoPrint=OFF,
    modelPrint=OFF, contactPrint=OFF, historyPrint=
    OFF, userSubroutine='',
    scratch='', resultsFormat=ODB)
mdb.jobs['sim'].submit(consistencyChecking=OFF)
mdb.jobs['sim'].waitForCompletion()
# MACRO_FINISH #

# GET THE STRESS AND DISPLACEMENT VALUES #
```

```
path='C:/Users/Pedro/Desktop/Universidade/5ano/1
    semestre/ONLE/Algoritmos/'
odbName = path+'sim'
odb = openOdb(odbName + '.odb')

#stresses
file_name_to_save='sim_Stresses.txt'

#this reads the last frame [-1] of the first step [0]


steplast=odb.steps['load'].frames[-1]
stress_outputs_number=len(steplast.fieldOutputs['S'].
    values)

nodes_number=len(steplast.fieldOutputs['S'].values
    [0].instance.nodes)
elements_number=len(steplast.fieldOutputs['S'].values
    [0].instance.elements)

mises=[]

for i in range(elements_number):
    mises.append(steplast.fieldOutputs['S'].values[i
    ].mises)

mises=np.array(mises)
mises=np.reshape(mises,[mises.size,1])



form='%11.7e'
file_name_to_save=path+file_name_to_save
np.savetxt(file_name_to_save, mises,fmt=form)



#displacement
file_name_to_save='sim_Displacements.txt'

#this reads the last frame [-1] of the first step [0]
steplast=odb.steps['load'].frames[-1]
disp_outputs_number=len(steplast.fieldOutputs['U'].
    values)
nodes_number=len(steplast.fieldOutputs['U'].values
    [0].instance.nodes)
elements_number=len(steplast.fieldOutputs['U'].values
    [0].instance.elements)

disp=[]

for i in range(elements_number):
    disp.append(steplast.fieldOutputs['U'].values[i].
    magnitude)


disp=np.array(disp)
disp=np.reshape(disp,[disp.size,1])

form='%11.7e'
file_name_to_save=path+file_name_to_save
np.savetxt(file_name_to_save, disp,fmt=form)
```

Otimização Não-Linear em Engenharia | MIEM.2021
(artigo/relatório de projecto)

Otimização de problemas com recurso ao *Harmony Algorithm*

Otimização da asa traseira do carro *Formula Student* e distribuição de centros de manutenção de uma companhia aérea

João Sarmento · Pedro Novais

Submetido: 20/01/2021

Resumo Neste relatório, é apresentado o processo de otimização de dois problemas propostos pelos autores no âmbito da cadeira de Otimização não linear em engenharia. São mencionadas tanto a formulação matemática de cada problema como a análise de sensibilidade do mesmo. Em seguida, procede-se à explicação do algoritmo em análise e à conclusão dos resultados que este permitiu obter. Estes resultados revelam que o algoritmo permite obter soluções viáveis para diversos tipos de problemas. No entanto, foram adicionadas duas variações do algoritmo original de forma a melhorar a sua performance.

Palavras-Chave *Harmony Search Algorithm* · *Airfoil* · Centro de manutenção

1 Introdução

Através de algoritmos de otimização, é possível obter soluções óptimas para problemas não-lineares de diversas áreas, tais como a aerodinâmica, transportes ou gestão operacional. O primeiro problema apresentado na Secção 2.1 pode ser inserido na área da aerodinâmica, com vista à maximização da performance da asa traseira do carro. Já no caso da gestão operacional e transportes pode-se associar o segundo tema, abordado na Secção 3.1, em que se pretende minimizar o custo despendido em manutenção numa companhia aérea.

J.Sarmento
n.º 84827
E-mail: joaomiguelsarmento@ua.pt

P.Novais
n.º 84825
E-mail: pedrogasparnovais@ua.pt

2 Otimização da asa traseira do carro *Formula Student*

2.1 Definição do problema

Actualmente, a performance aerodinâmica tem um papel fundamental nos desportos automóveis de competição. Na competição *Formula Student* o percurso é essencialmente constituído por curvas, o que faz com que o universo de velocidades se encontre entre os 50 e os 100 km/h. Por este motivo o uso de aparelhos aerodinâmicos é essencial de forma a garantir uma maior aderência do carro através de um peso virtual (*Downforce*), mantendo a força de arrasto (*Drag*) limitada.

A performance aerodinâmica é influenciada por vários fatores, entre eles: a velocidade do carro, que foi mantida constante ao longo da análise, o ângulo de ataque e a forma do *airfoil*, sendo esta última definida por três variáveis (*Thickness*, *Max camber*, *Point of max camber*).

2.2 Formulação e classificação do problema

Em seguida, apresenta-se a primeira formulação atribuída ao problema.

$$\begin{aligned} &\text{Procurar } \{T,P,M,\alpha\} \text{ de modo a:} \qquad (1)\\ &\text{minimizar } \frac{C_\text{l}}{C_\text{d}} = f([X_i,\, Y_i(T,\, P,\, M)],\, \alpha),\\ &\text{com } Y_i = g(T,P,M,X_i),\, i = 1,\ldots,N\\ &\text{sujeito a } 0 \leq X_i \leq 1,\\ &\qquad 0 \leq \alpha \leq 20,\\ &\qquad 0 \leq T \leq 0.5,\\ &\qquad 0 \leq M \leq 0.2 \; e\\ &\qquad 0 \leq P \leq 0.8\ . \end{aligned}$$

em que X_i e Y_i correspondem ao conjunto de coordenadas que definem o *airfoil*. X_i é um valor normalizado e constante ao longo das avaliações, sendo este delineado por um *linspace*, e N o número de pontos que representa o *airfoil*. O parâmetro α corresponde ao ângulo de ataque do *airfoil* em relação á direcção do vetor velocidade, **U**. Os parâmetros T, M e P correspondem, respectivamente, à *Thickness*, espessura máxima do *airfoil*, à *Max camber*, altura máxima da linha que define a curvatura do *airfoil* e ao *Point of max camber* que corresponde à abcissa onde ocorre a *Max camber*. Todos estes parâmetros podem ser identificados na Figura 1.

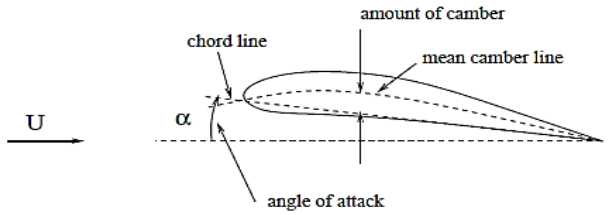

Figura 1. Propriedades do *airfoil* (www.wikipedia.org/NACA-airfoil)

A função objetivo é não-linear, contínua e adimensional. Em relação às variáveis da função, estas são contínuas, com exceção de α que é tratada como uma variável discreta, e todas estas devem respeitar os intervalos apresentados.

É importante acrescentar que este problema é do tipo *blackbox*, em que a função objetivo está dependente de um programa auxiliar para obter resultados, neste caso o *Xfoil*.

2.3 Análise de sensibilidade

Para uma maior perceção do comportamento da função objetivo, procedeu-se a uma análise de sensibilidade da mesma. Para isso, inicialmente variou-se cada uma das variáveis (T, M, P) individualmente, mantendo as restantes com valor constante, como se pode observar na Figura 2, em que é visível a variação do valor da função objetivo ao longo do aumento da *Max camber*. É de notar um valor *outlier*, próximo de 0 (0,404), na Figura 2, que corresponde ao caso em que simulação executada no *Xfoil* não converge. Este valor foi escolhido propositadamente para ser facilmente distinguido dos restantes.

Este processo foi repetido para as restantes variáveis, no entanto este tipo de análise revelou-se insuficiente para este problema, pois a variação singular de um dos parâmetros não permite avaliar o *airfoil* de forma geral. Para isso foi criado um espaço de dados 3D de

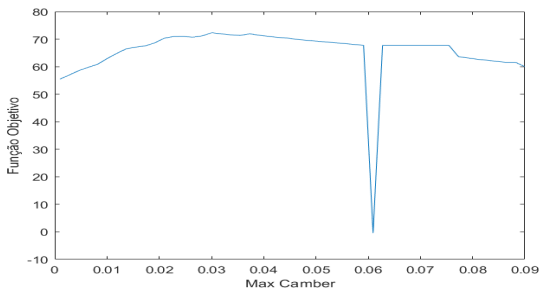

Figura 2. Análise de sensibilidade á componente *Max camber*

forma a tentar encontrar alguma linearidade na função, este espaço de dados encontra-se representado na Figura 3, em que se pode observar a variação do valor da função objetivo, representado na barra de cores, para todo o espaço de dados possível. Com isto é possível confirmar, a partir das representações gráficas, a existência de vários mínimos locais e a não existência de nenhuma linearidade visível.

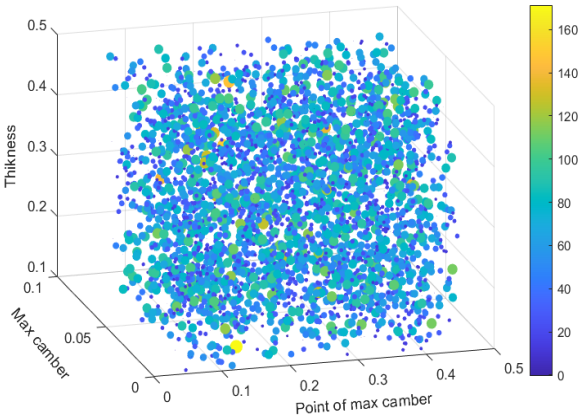

Figura 3. Análise de sensibilidade com valores *random* em todos os parâmetros

2.4 Avaliação

A programação do problema foi realizada em *Matlab*, onde foram criadas duas funções. A primeira prepara um documento do tipo .txt com as coordenadas do *airfoil* e efetua o *plot* do mesmo. A restante função tem como objetivo o envio do *script*, que irá servir de guia para a simulação. Adicionalmente, a função irá ler e armazenar os resultados provenientes do *Xfoil*.

Devido ao *Xfoil* por vezes demorar cerca de 10 minutos a convergir numa avaliação da função objetivo foi adicionada uma *toolbox*, *Parallel Computing Toolbox*, de

forma a que o *Matlab* pudesse ter acesso ao tempo decorrido em cada iteração a meio do processo. Esta *toolbox* possibilitou o fecho automático do *Xfoil* através da linha de comandos e, simultaneamente, a passagem para a próxima avaliação ao fim de um tempo estipulado.

2.5 Algoritmos de otimização

Relativamente ao algoritmo, teve-se em especial atenção a escolha de um que permitisse obter bons resultados, tanto no *benchmark*, como nos problemas propostos pelo grupo. Assim, optou-se por usar o *Harmony Search Algorithm* (HSA) adaptado de [2], que se baseia nos princípios musicais de improvisação de harmonias. Este algoritmo metaheurístico, através de alguns parâmetros de carácter aleatório, tem a capacidade de realizar buscas tanto em mínimos locais como em mínimos globais nos problemas não lineares.

O HSA começa por gerar aleatoriamente uma população/Memória de Harmonias (HM). Em seguida, é improvisada uma Harmonia em cada iteração, cada harmonia será constituída por X_j, ou notas, com $j = 1, \cdots, max$, como se encontra representado na Figura 4. Esta pode ser uma cópia parcial de uma Harmonia já predefinida na HM, ou um valor aleatório dentro do domínio, ou ainda uma afinação. Estas possibilidades são regidas pelos parâmetros *PAR*, *bw* e *HMCR*, onde *HMCR* é a probabilidade de gerar uma nota aleatória, *PAR* define a probabilidade de afinação de uma nota copiada e *bw* é a *bandwith* da afinação.

No sentido de assegurar que a afinação era mantida no domínio de X_j, a afinação foi corrigida através da equação seguinte:

$$X_{\text{new},\,j} = X_{\text{new},\,j} + rand(-bw(-x_{\min} + X_{\text{new},\,j}), bw(x_{\max} - X_{\text{new},\,j}))\ e\ X_{\text{new},\,j} \in [x_{\min}, x_{\max}] \quad (2)$$

onde $X_{\text{new},\,j}$ é uma nova nota da variável tipo j, com domínio $[x_{\min}, x_{\max}]$. Apesar desta restrição ao domínio não ser necessário para o *benchmark*, revelou-se necessária para o problema na Secção 2.1, onde variáveis fora do domínio não convergem para uma solução.

De forma a otimizar o HSA foram testadas duas variantes do mesmo. Após uma análise inicial à formulação base do HSA, onde os parâmetros *PAR*, *bw* e *HMCR* são estáticos, verificou-se a necessidade de adaptar estes parâmetros ao longo das iterações. Assim, seguimos a formulação proposta por [2] através das equações 3 e 4 e criou-se a variante 2, onde os parâmetros

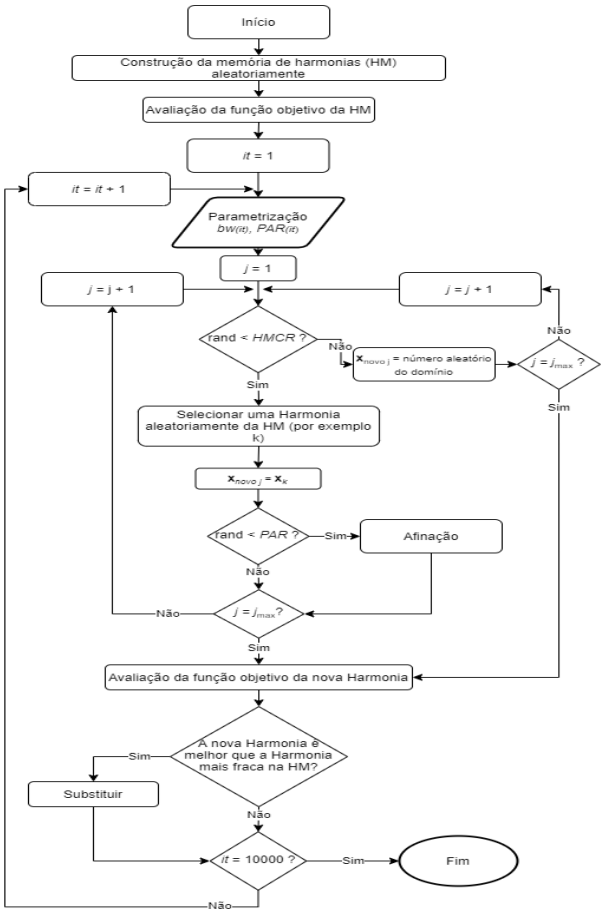

Figura 4. Fluxograma do algoritmo

PAR e *bw* são dinâmicos e dados por,

$$PAR(it) = PAR_{\min} + \frac{PAR_{\max} - PAR_{\min}}{it_{\max}} it, \quad (3)$$

$$bw(it) = bw_{\max} \times e^{(Ln(\frac{bw_{\min}}{bw_{\max}})\frac{it}{it_{\max}})}, \quad (4)$$

onde it corresponde à iteração do HSA e $PAR_{\min}$, $PAR_{\max}$, $bw_{\min}$ e $bw_{\max}$ são parâmetros estáticos. No sentido de os calibrar aplicou-se um novo algoritmo HSA em série com a variante 2. Com estes resultados criou-se a variante 3.

2.6 Resultados

Numa primeira avaliação deste problema, foi usada uma Memória de 20 Harmonias e foram alcançadas as 1000 avaliações. Através da figura 5 é possível observar a não convergência total dos valores da função objetivo ao fim das 1000 avaliações. No entanto, devido ao custo computacional deste problema não foi possível gerar mais avaliações.

Por este motivo, não foram ajustados os parâmetros do algoritmo para este problema em especifico. Em vez disso foi aproveitada a otimização dos parâmetros feita para o benchmark. A partir da variante 2 do algoritmo seleccionado na Secção 2.5 e da função objetivo 2, foram então possíveis obter os resultados representados na Figura 5, em que na parte superior, tanto é possível observar a evolução dos resultados da função objetivo das harmonias presentes na memória (Figura à esquerda), como também as variáveis das harmonias presentes na memória obtida no fim das 1000 avaliações (Figura à direita). Na parte inferior estão também representados os *airfoils* ótimos obtidos ao longo da simulação. A vermelho encontra-se o *airfoil* com melhor performance e que pode ser obtido através dos seguintes parâmetros (0,0667; 0,0794; 0,5192), para um ângulo de ataque α de -3º, obtendo assim um $\frac{C_l}{C_d}$ igual a
-157,97 e um C_l de apenas -1,2912.

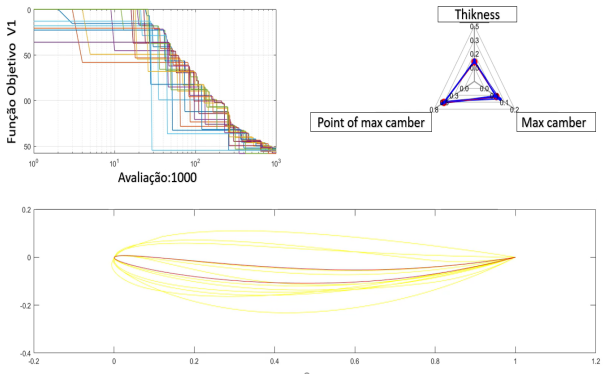

Figura 5. Evolução da memória de harmonias e forma do *airfoil* ao longo das avaliações

Como foi referido na Secção 2.1, para as competições do tipo *formula student* o objetivo principal de um kit aerodinâmico é gerar uma *downforce* mantendo a componente do *drag* com valores razoáveis. No entanto, para a função objetivo usada, o peso atribuído a cada componente encontra-se igualmente distribuído, o que se revelou não ser o mais indicado.

Por este motivo, procedeu-se à concretização de duas novas propostas que se ajustassem mais ao problema em análise, ambas tomando como base a seguinte equação,

$$\begin{aligned} \text{minimizar } h(C_l,C_d) &= \beta\frac{C_l}{C_d} + (1-\beta)NC_l, \\ \text{onde : } C_l &= s_1(T,M,P,\alpha) \ e \\ C_d &= s_2(T,M,P,\alpha) \ . \end{aligned} \tag{5}$$

Numa primeira abordagem de forma a aumentar a relevância da componente C_l adotaram-se os seguintes valores, $N = 100$ e $\beta = 0{,}45$, sendo N o termo responsável pela equalização de grandezas e β responsável pelo controlo da relevância de cada termo. Com isto foi possível obter uma importância de 55 % para o C_l, obtendo um *airfoil* ótimo com capacidade de gerar um $C_l = -2{,}158$ e um $\frac{C_l}{C_d} = -119{,}445$, para um α igual a -4º.

Considerando os resultados insuficiente, optou-se por dar ainda uma maior importância ao termo de C_l, através da terceira versão da função objetivo com os valores, $N = 100$ e $\beta = 0{,}25$, atribuindo uma importância de 75% para o C_l, onde foi possível obter o seguinte *airfoil*, da Figura 6 em que para um $\alpha = -13º$ e $(T,M,P) = (0{,}1318; 0{,}1605; 0{,}4885)$ foi possível obter um $C_l = -2{,}357$ e um $\frac{C_l}{C_d} = -92{,}130$.

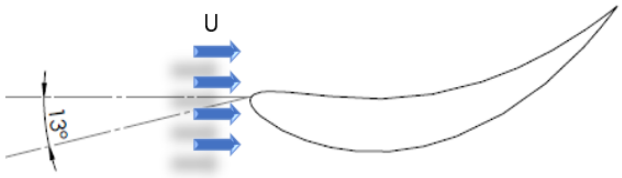

Figura 6. *Airfoil* com melhor performance aerodinâmica

2.7 Conclusões

Após a elaboração deste trabalho, pode-se afirmar que a solução ótima obtida alcançou níveis de desempenho elevados, especialmente comparando com os valores apresentados por outras equipas de *Formula Student*, em que os valores de C_l rondam os -4 para combinações de 3 *airfoils*. É também de notar que os valores obtidos resultam de uma simulação 2D, por isso não incluem uma das componentes do *Drag*, o *lift induced drag*. Ou seja, a relação $\frac{C_l}{C_d}$ apresentada não corresponde totalmente aos valores que se obteria na realidade.

Como trabalho futuro, poder-se-á aplicar este algoritmo a combinações de *airfoils* e também efectuar a otimização para diferentes velocidades, pois por exemplo no caso apresentado como solução ótima, devido ao elevado ângulo de ataque, para velocidades mais baixas existe a possibilidade do sistema aerodinâmico entrar em *stall*, perda completa do contacto com o ar na parte inferior do *airfoil*.

3 Distribuição de centros de manutenção de uma companhia aérea

3.1 Definição do problema

Uma companhia aérea pretende construir cinco centros de manutenção (CM) para dar suporte a 12 aeroportos situados na Ásia, Oceania, África e Europa. Em cada aeroporto o número de aviões que requerem manutenção varia, sendo que cada centro de manutenção só poderá servir um máximo de 60 aviões. O custo de construção dos CM dependerá da sua longitude e do tipo de plataforma, podendo esta ser marítima ou terrestre, de acordo com a Tabela 1. A este custo, acresce o custo de deslocação entre os aeroportos e os CM.

3.2 Formulação e classificação do problema

Em seguida, apresenta-se a primeira formulação atribuída ao problema.

Procurar $\mathbf{X}_i$ de modo a:

$$\begin{aligned} \text{minimizar} \quad f(\mathbf{X}_i) = \sum_{i=1}^{5} \Bigg[& C_i^{\text{Construção}} M_i \\ & + \alpha \sum_{j=1}^{n} (W_{i,j} d_j(\mathbf{X}_i)) \Bigg] \quad (€), \end{aligned} \tag{6}$$

sujeito a

$$0 \leq \sum_{j=1}^{\text{n}} W_{i,j} \leq 60, \quad i = 1, \cdots, 5 \quad j = 1, \cdots, 12.$$

Onde:

$$\mathbf{X}_i = [\theta_i, \phi_i], \tag{7}$$

$$\theta_i \in [-40,80],$$

$$\phi i \in [-20,160],$$

$$\begin{aligned} d_j(\mathbf{X}_i) = 2r \arcsin \Bigg(& \sin^2 \frac{\theta_i - \theta_j}{2} \\ & + \cos\theta_i \cos\theta_j \\ & \sin^2 \frac{\phi_i - \phi_j}{2} \Bigg)^{1/2} (\text{km}), \end{aligned} \tag{8}$$

$$\alpha = 50 \quad \left(\frac{€}{\text{km}}\right), \tag{9}$$

$$M_i = \begin{cases} 1 & \text{se} \quad \mathbf{X}_i \notin \text{ Cidade}, \\ 2 & \text{se} \quad \mathbf{X}_i \in \text{ Cidade}, \end{cases} \tag{10}$$

$$C_i^{\text{Construção}} = \begin{cases} 300000000 & \text{se} \quad \phi_i \in [-20,40], \\ 150000000 & \text{se} \quad \phi_i \in \,]40,160], \\ 1000000000 & \text{se} \quad \mathbf{X}_i \in \text{Mar}, \quad (€) \end{cases} \tag{11}$$

em que $\mathbf{X}_i$ representa as coordenadas, latitude θ_i e longitude ϕ_i, do centro de manutenção i. O parâmetro α corresponde à despesa de deslocação por km. Esta despesa é calculada através da distância percorrida $d_j(\mathbf{X}_i)$ pelos aviões $W_{i,j}$, do aeroporto j até ao centro de manutenção i. O parâmetro $C_i^{\text{Construção}}$ representa o custo de construção do centro de manutenção. Este custo pode sofrer uma penalização M se $\mathbf{X}_i$ se encontrar num raio de 30 km de uma cidade.

Tabela 1. Localização *vs* Custo

Plataforma	Longitude [ºE]	Custo [€]
Terrestre	[-20, 40]	300 000 000
Terrestre	] 40, 160]	150 000 000
Marítima	[-20, 160]	1 000 000 000

A função objetivo deste problema é não-linear, sendo que as variáveis de otimização, $\mathbf{X}_i$ são linearmente independentes e contínuas. A função objetivo tem uma restrição, porém esta será transformada numa regra heurística dentro do programa *Matlab*, a fim de evitar penalizações na função objetivo.

3.3 Análise de sensibilidade

No sentido de perceber melhor o comportamento da função objetivo foi efetuado uma análise de sensibilidade. Nesta análise começou-se por decompor a função objetivo em vários elementos, M, $C_i^{\text{Construção}}$ e custo de deslocação dado por $\alpha \sum_{j=1}^{n} (W_{i,j} d_j(\mathbf{X}_i))$, apresentadas nas Figuras 11, 12, 13, respetivamente. A análise revelou que o $C_i^{\text{Construção}}$ tem um maior impacto na função objetivo face ao custo de deslocação e poderá comprometer a eficácia do algoritmo. Assim criou-se um fator N para que o impacto do custo de deslocamento seja mais significativo. Simultaneamente adicionou-se um peso β. A equação seguinte é o resultado desse ajuste:

$$\begin{aligned} f(\mathbf{X}_i) = \sum_{i=1}^{5} \Bigg(& (1-\beta) C_i^{\text{Construção}} M_i \\ & + \beta N \alpha \sum_{j=1}^{n} (W_{i,j} d_j(\mathbf{X}_i)) \Bigg) \quad (€), \end{aligned} \tag{12}$$

$$\beta \in [0,1],$$

$$N = 33.5285,$$

onde N é um elemento de normalização e foi obtido após uma primeira convergência da equação 6.

3.4 Avaliação

A função objetivo e o algorítmo de otimização foram implementados em *Matlab*. O código começa por gerar

cinco centros de manutenção dispersos no domínio. Em seguida, o programa averigua a existência de cidades próximas dos CM recorrendo à base de dados da *Mapping Toolbox*. Depois, recorrendo a funções internas da *Mapping Toolbox* os CM são categorizados como plataformas terrestres ou marítimas.

Por fim, calcula-se a distância percorrida dos aviões entre o aeroporto e o centro de manutenção. Para a atribuição de um avião a um centro de manutenção é usado o critério da menor distância percorrida, que exceptuando pequenas exceções, traduz-se na menor distância possível percorrida por todos os aviões com necessidades de manutenção. Aliado a este critério existe uma condição que restringe o número de aviões por centro de manutenção.

3.5 Algoritmos de otimização

O algoritmo usado para a resolução deste problema foi o HSA e a sua implementação foi similar à descrita na Secção 2.5. Porém, a existência de uma restrição levou à necessidade de criar uma regra heurística dentro do código *Matlab* que assegurasse o seu cumprimento.

3.6 Resultados

Na minimização da equação 6 usou-se uma HM composta por 20 harmonias. Os parâmetros necessários para o algoritmo coincidem com a variante 2 definida na Tabela 2. Após 10000 avaliações da função objetivo obteve-se a curva de convergência, Figura 7, onde cada linha representa uma harmonia na HM. Foi atingido um mínimo de 772 733 377 €, composto em 750 000 000€ de custo de construção em centros de manutenção em plataformas terrestre na Ásia e 22 733 377 € em custos associados ao deslocamento dos aviões.

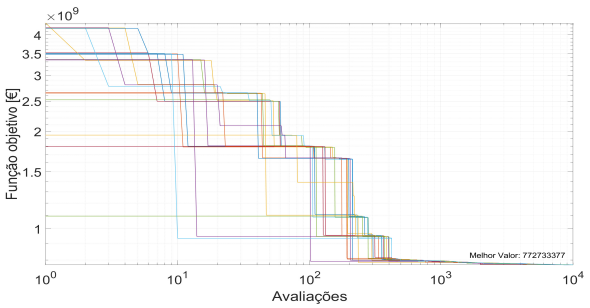

Figura 7. Curva de convergência da função objectivo 6

A Figura 7 revela ainda uma alta eficiência do algoritmo nas primeiras 500 iterações. Este comportamento era esperado, dado que a função objectivo possui duas componentes com grandezas diferentes, portanto, numa fase inicial foi dada uma prioridade à grandeza de maior escala. Este tipo de comportamento conduz a mínimos locais e assim, reduz a exatidão do algoritmo.

Para normalizar a função objetivo procurou-se minimizar a Equação 12 com os parâmetros anteriormente estabelecidos. Realizou-se uma nova curva de convergência, Figura 14, onde $\beta = 0{,}5$, e um gráfico pareto representado pela Figura 8, onde $\beta_i \in [0{,}1]$, com $i = 0, \cdots, 20$. Neste pareto verifica-se dois pontos de utopia. O primeiro dá-se para $\beta = 0$ e corresponde à minimização do custo de construção. Por sua vez, quando $\beta = 1$, o custo de deslocação é mínimo em prol do aumento do custo de construção.

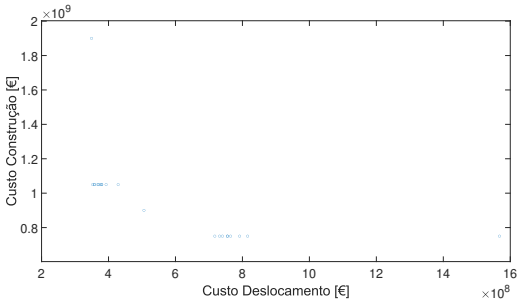

Figura 8. Pareto multiobjetivo: Custo deslomento *Vs* Custo de construção

A tendência hiperbólica do pareto faz nos concluir que as duas componentes da função objetivo são conflituantes, ou seja, a minimização de um dos objetivos leva ao detrimento do segundo, o que implica a existência de um espaço de soluções.

3.7 Conclusões

O algorítmo HSA foi capaz de convergir para o problema exposto e chegou a um mínimo de 72 733 377 €. Porém, numa análise multiobjetivo observou-se um conflito entre o custo de construção.

Para trabalhos futuros, seria interessante analisar um novo critério na distribuição dos aviões pelos aeroportos uma vez que o critério de menor distância nem sempre é o mais eficiente. Outro sugestão passaria pelo uso de mais variáveis no cálculo da função objetivo para que o problema se aproxime o máximo do contexto real.

A Benchmark 2020

A.1 Enquadramento do benchmark

O *benchmark* proposto corresponde a um problema de minimização de peso de um sistema redutor. A natureza do problema em causa é de otimização combinatória, em que 6 dos termos usados para determinar o peso final do sistema são variáveis do tipo continuas e o restante do tipo discreto. A formulação do problema encontra-se descrita em [1].

A.2 Implementação

O algoritmo proposto - Figura 4, Secção 2.5 - foi implementado em *Matlab*. Para o benchmark, foram testadas todas as variações apresentadas na secção 2.5 de forma a obtermos resultados o mais próximos possível dos valores mencionados em [1].

A.3 Resultados

O HSA para cada variante foi executado 100 vezes, com um critério de paragem de 10000 iterações, com uma HM de 20 harmonias, perfazendo um total de 10020 avaliações por execução. Assim, traçou-se um gráfico dos valores médios e medianos da evolução da função objetivo ao longo das iterações, Figura 9. Os valores usados na parametrização do HSA encontram-se na Tabela 2.

Devido ao desenvolvimento logarítmico de V1, V2 e V3 a precisão pode ser aproximada à diferença entre a média, sensível a extremos, e a mediana. A Figura 9 revela que a a variante 3 obtém o erro mínimo, em detrimento de uma baixa precisão. Coloca-se a hipótese de este comportamento ser causado pelo valor alto *HMCR*, parâmetro responsável pela geração de notas aleatórias. Denota-se que este tipo de comportamento não é compatível com problemas onde o número de avaliações é um fator limitante. A variante V2 consegue auxiliar uma boa exatidão, onde o erro, aquando comparado com os valores fornecidos em [1] e ao fim de 10000 iterações é apenas de 0.53%, a uma boa precisão, uma vez que a média e a mediana convergem para o mesmo valor em menos de 2000 iterações. O uso de parâmetros adaptativos revala-se ser uma metodologia eficiente quando comparado com o algoritmo V1, onde os parâmetros são estáticos.

Todas as variantes dos algoritmos apresentam duas fases distintas em termos de eficiência, uma inicial em que o ritmo de convergência tem um comportamento logarítmico e uma segunda em que o valor quase estabiliza. A dificuldade de convergência, quando existe uma aproximação ao mínimo global, pode ser justificada pela falta de precisão do parâmetro *bw* na vizinhança do mínimo, gerando um salto entre harmonias superior ao necessário para obter uma convergência para o mínimo.

Após o sucesso em diversos problemas é possível validar o algoritmo em termos de robustez. Porém, para trabalhos futuros sugere-se que seja desenvolvido um novo mecanismo de calibração dos parâmetros e que o número de harmonias na HM e o *HMCR* sejam incluídos na calibração.

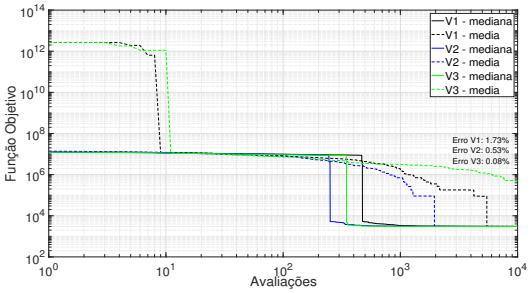

Figura 9. Evolução da função objetivo recorrento às variantes 1, 2 e 3 do HSA

	HCMR	PARmin	PARmax	Bwmin	Bwmax
V1	0.5	0.5	0.5	0.5	0.5
V2	0.5	0.1	0.8	0.01	0.5
V3	0.820	0.088	0.382	0.003	0.534

Tabela 2. Tabela das variáveis de controlo nas várias versões.

Referências

1. A. Andrade-Campos, J. Dias-de-Oliveira, *Benchmark 2020: Weight minimisation of a speed reducer*
2. Alireza Askarzadeh e Esmat Rashedi, *Harmony Search Algorithm:Basic Concepts and Engineering Applications*,Iran.

B Anexo - Informação complementar ao problema 2

Os aviões descritos no problema dos centros de manutenção, Secção 3.1, encontram-se dispersos por 12 aeroportos de acordo com a Figura 10.

Figura 10. Número de aviões que aguardam manutenção em cada aeroporto

Ao decompor a função objetivo nos seus membros, foi possível criar várias figuras que poderão facilitar simultaneamente a compreensão e sensibilidade ao problema. A Figura 11 corresponde a uma matriz máscara, onde os pontos a branco são as principais cidades do globo, ou seja, onde existe uma penalização M.

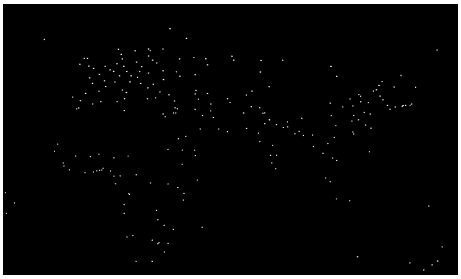

Figura 11. Cidades da *Mapping Toolbox* do *Matlab*

A Figura 12 resulta de uma análise ao custo de deslocamento. Desta análise verificou-se que a Europa central poderá ser uma boa candidata para a construção de pelo menos um centro de manutenção. A forma elítica das isolinhas poderá ser explicada pela elevada concentração de aviões a Este, no Japão (Tokyo), na Rússia (Vladivostok) e na Austrália (Sydney).

Da análise ao custo de construção resultou a Figura13. A cor amarela corresponde aos locais onde será necessário a construção de uma plataforma marítima, e consequentemente, em que o custo de construção será mais elevado. Já os continentes foram divididos em dois, um predominantemente Asiático (Azul escuro), com o custo mais reduzido, e um segundo correspondente à Europa mais África, com o dobro do preço da área azul escura.

Para normalizar a função objetivo procurou-se minimizar a Equação 12, dada na Secção 3.3 com os parâmetros estabelecidos na Secção 3.2. Realizou-se uma nova curva de convergência, Figura 14, para $\beta = 0{,}5$.

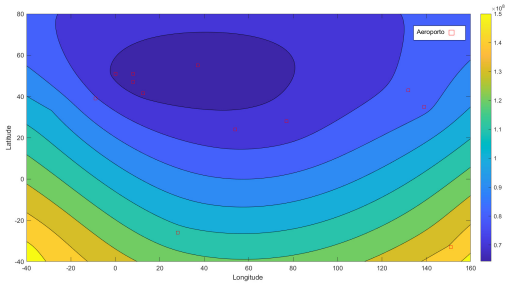

Figura 12. Análise à sensibilidade do custo de deslocamento

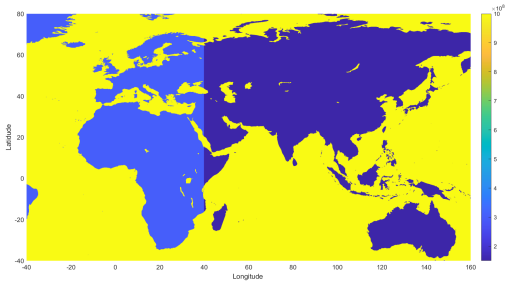

Figura 13. Análise à sensibilidade do custo de construção

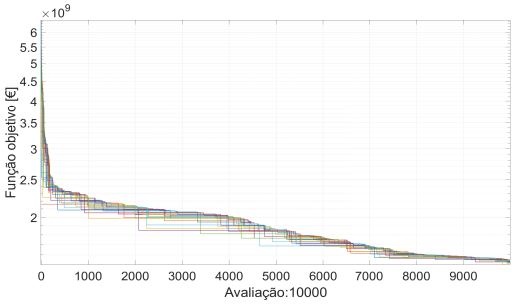

Figura 14. Curva de convergência da equação 12

Otimização Não-Linear em Engenharia | MIEM.2021
(artigo/relatório de projecto)

Otimização da disposição de peças para impressão 3D e torres de internet 5G

Resolução de dois problemas de engenharia usando a heurística *Firefly*

Francisco Power · Pedro Rolo

Submetido: 19/1/2021

Resumo Resolução de dois problemas reais e atuais de engenharia, a disposição de pecas na cama quente de uma impressora 3D e de torres de *internet* 5G num determinado espaço, procurando soluções ótimas para cada problema, usando como ferramenta de otimização, o algoritmo *Firefly*. Avaliação do desempenho deste algoritmo resolvendo o problema *Benchmark* 2020 (ver anexo A). O algoritmo demonstrou ser bastante robusto, produzindo resultados muito satisfatórios.

Palavras-Chave Otimização · *Firefly* · Heuristicas · *internet* 5G · impressão 3D

1 Introdução

A *internet* 5G e a impressão 3D são duas tecnologias que apresentam inúmeras vantagens face a tecnologias semelhantes, mas também têm desvantagens. É possível atenuar os efeitos desses problemas traduzindo-os em problemas de otimização e resolvendo-os como tal.

Neste trabalho foram abordados dois desafios, formulados como problemas de otimização e resolvidos usando uma meta-heurística, o algoritmo *Firefly*. Estudou-se, também, o desempenho e robustez deste algoritmo através da resolução do problema *Benchmark* 2020, o problema da minimização do peso de um redutor.

Francisco Power
n.º 84706
E-mail: franciscopower66@ua.pt

Pedro Rolo
n.º 84803
E-mail: pmrolo@ua.pt

2 Apresentação dos problemas de otimização

2.1 Impressão 3D

A impressão 3D não só é extremamente útil para prototipagem rápida, como também já é usada para produções de peças de geometria complexa de pequena e média escala. No entanto, ainda é um processo relativamente lento.

O objetivo deste trabalho é desenvolver um programa que minimize o tempo de impressão, minimizando o percurso da extrusora em vazio. Conhecendo o formato das peças a imprimir e as dimensões da cama quente, o programa deverá dispor as peças na cama quente de forma ótima, aplicando uma determinada translação e rotação a cada peça. O programa terá que ter em consideração o percurso da extrusora em todas as camadas da impressão e não apenas a forma da base.

2.2 Internet 5G

A internet 5G traz inúmeras vantagens, nomeadamente a velocidade de transmissão de dados, o que permitirá grandes avanços na área de *internet of things*, condução autónoma, entre outras áreas. No entanto, as ondas de comprimento de onda milimétrico têm um alcance bastante baixo, sendo também facilmente bloqueadas por edifícios e outras obstruções de tamanhos similares. O objetivo deste trabalho é descobrir, para uma certa área e um certo numero de torres 5G, a distribuição que ofereça a máxima cobertura do espaço, tendo em conta o efeito das obstruções mencionado. Será, posteriormente, avaliado também o melhor número de torres 5G.

A área que se pretende afetar é obtida através de uma imagem de satélite, a qual é depois convertida num

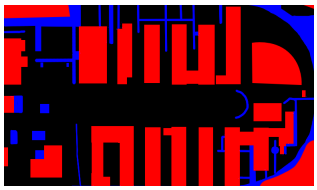

Figura 1. Mapa de obstruções (vermelho) e zonas restritas (vermelho + azul).

mapa de obstruções e restrições, como se pode ver na Figura 1.

3 Formulação dos problemas de otimização

3.1 impressão 3D

Neste problema, o objetivo é minimizar a distância que a extrusora percorre em vazio, ou seja, o somatório em todas as camadas da distância no plano entre o ponto de saída da extrusora de uma peça, $[a_n,b_n]$, e o ponto de entrada da peça seguinte, $[c_{n+1},d_{n+1}]$. Estes relacionam-se com o vetor de translação, $[x_{(t,n)},y_{(t,n)}]$, e o ângulo de rotação de cada peça, θ_n, substituindo $[x_n,y_n]$ por $[a,b]$ ou $[c,d]$ em:

$$\begin{bmatrix} x_n \\ y_n \\ f(x_n,y_n) \\ 1 \end{bmatrix} = \begin{bmatrix} cos(\theta_n) & -sin(\theta_n) & 0 & x_{(t,n)} \\ sin(\theta_n) & cos(\theta_n) & 0 & y_{(t,n)} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{n_0} \\ y_{n_0} \\ f(x_{n_0},y_{n_0}) \\ 1 \end{bmatrix} \tag{1}$$

sendo f a função de forma de cada peça, $[x_{n_0},y_{n_0}]$ as coordenadas x e y de um ponto da peça n estando esta centrada na origem do referencial e $[x_n,y_n]$ as coordenadas x e y do mesmo ponto após a transformação. Assim sendo, e considerando N o número de peças a imprimir e C o número máximo de camadas de impressão, pode-se formular o problema como apresentado seguidamente:

Procurar $\mathbf{x} = \left[x_{(t,n)},\, y_{(t,n)},\, \theta_n\right]^{\mathrm{T}}$ de forma a:

$$\begin{aligned} \text{minimizar } F(\mathbf{x}) = \sum_{c=1}^{C} \Bigg(\sum_{n=1}^{N-1} & \left(\sqrt{(c_{n+1}-a_n)^2 + (d_{n+1}-b_n)^2} \right) \\ & + \sqrt{(c_1-a_N)^2 + (d_1-b_N)^2} \Bigg)_c \end{aligned} \tag{2}$$

$$\begin{aligned} \text{sujeito a } g_1(\mathbf{x}) &= -x_n - \mathrm{Cc}/2 \leq 0, \\ g_2(\mathbf{x}) &= x_n - \mathrm{Cc}/2 \leq 0, \\ g_3(\mathbf{x}) &= -y_n - \mathrm{Lc}/2 \leq 0, \\ g_4(\mathbf{x}) &= y_n - \mathrm{Lc}/2 \leq 0, \\ g_5(\mathbf{x}) &= -\sqrt{(c_{n+1}-a_n)^2 + (d_{n+1}-b_n)^2} \\ &\quad + 3 \leq 0, \\ h(\mathbf{x}) &= A(f_n(x,y) \wedge f_{n+1}(x,y)) = 0, \end{aligned} \tag{3}$$

sendo Cc o comprimento da cama quente e Lc a largura da cama quente e estando n a variar entre 0 e o número de peças.

3.2 Internet 5G

Para um determinado número de torres, o objetivo é minimizar a zona não coberta, o que pode ser feito calculando a intensidade de sinal em cada ponto, seguido do cálculo da razão entre a área com uma intensidade de sinal inferior a uma intensidade mínima e a área total.

A intensidade de sinal num determinado ponto é dada por:

$$I(x_n,y_n) = \frac{P}{4\pi\left((x-x_p)^2 + (y-y_p)^2\right)}\ , \tag{4}$$

sendo P a potência do sinal, $[x,y]$ as coordenadas do ponto em análise e $[x_p,y_p]$ as coordenadas da torre 5G mais próxima e sem obstruções entre o ponto e a torre). Assim sendo, e considerando A a área do espaço excluindo as obstruções e considerando $I_{\min}$ a intensidade mínima de sinal admissível, pode-se definir o problema como apresentado seguidamente.

Procurar $\mathbf{x} = \left[x_n,\, y_n\right]^T$ de forma a:

$$\begin{aligned} \text{minimizar } F(\mathbf{x}) &= \frac{\sum_{x=0}^{Ce}\sum_{y=0}^{Le} a}{A} \cdot 100\ , \\ a &= \begin{cases} 0, \text{ se } I(x_p,y_p) > I_{\min} \\ 1, \text{ se } I(x_p,y_p) \leq I_{\min} \end{cases} \end{aligned} \tag{5}$$

$$\text{sujeito a } R(\mathbf{x}) = (x_n,y_n) \in A_{\text{permitida}} \tag{6}$$

$$\begin{aligned} \text{e com } 0 \leq x_n \leq Ce, \\ 0 \leq y_n \leq Le, \end{aligned} \tag{7}$$

sendo Ce o comprimento e Le a largura do espaço a cobrir.

É importante notar que os mapas de obstruções e áreas restritas para colocação de torres são imagens. Assim sendo, as coordenadas das posições das torres 5G são as coordenadas de determinados pixeis da imagem, logo, discretas inteiras.

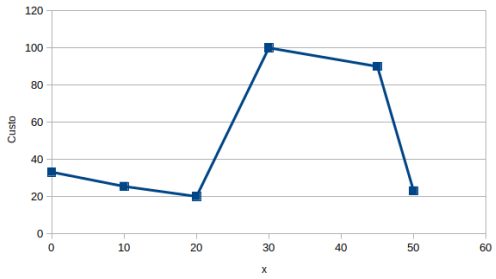

Figura 2. Variação do custo variando a coordenada x de uma torre 5G

4 Análise de sensibilidade

A análise de sensibilidade tem como principal objectivo verificar o efeito da alteração de uma variável na função objectivo, mostrando assim o quão sensível é a função objetivo a mudanças dessa variável.

4.1 Impressão 3D

A análise de sensibilidade deste problema fez-se variando as coordenadas dos pontos de chegada e saída de objeto. Alterando a posição dos objetos altera-se o comprimento do percurso da extrusora, que é directamente proporcional ao tempo de impressão.

Para isso usou-se o programa Cura da Ultimaker® de modo a simular os tempos de impressão de um conjunto de peças diferentes em posições diferentes. Foi possível confirmar que, quanto maior é a distancia entre os objetos, maior e o tempo de impressão, e que a orientação também influencia o seu tempo de impressão.

4.2 Internet 5G

Após programar o problema, foi possível observar que o valor resultado da função objetivo é menor no caso em que as torres estão muito juntas num espaço confinado, abrangendo uma pequena área, e significativamente maior no caso em que as torres estão mais afastadas e abrangendo uma área maior.

É também visível a existência de degraus na função objetivo, devido às obstruções, como se pode ver na Figura 2, em que os valores foram obtidos alterando apenas uma coordenada da posição de uma torre. Isto é algo de elevada importância na escolha do algoritmo de otimização.

5 Avaliação

5.1 Impressão 3D

Para poder fazer a avaliação do problema, foi, primeiro, necessário obter as coordenadas de todos os pontos do percurso da extrusora. Para tal, foi gerado o código CN no programa Cura, com 4 peças arbitrarias centradas na origem da cama quente. Um programa em python lê os documentos *.gcode* e cria uma matriz de pontos, sendo possível extrair dessa matriz os pontos de inicio e fim do percurso da extrusora em cada camada. O algoritmo de otimização Firefly, descrito na Secção 6, gera os valores de translação e rotação para cada peça e são aplicadas as transformações a todos os pontos de cada peça. Por fim, é calculado o valor da função objetivo, como descrito na Secção 3.1.

5.2 Internet 5G

A avaliação do problema começa com a geração das coordenadas das torres pelo algoritmo Firefly, descrito em 6, e introdução das mesmas num programa python que, tendo também como argumentos de entrada o mapa de obstruções e restrições (ver Figura 1), percorre todos os pontos do mapa, verifica quais as torres sem obstruções entre a elas e o ponto e, de entre essas torres, calcula o valor da função objetivo, como descrito na Secção 3.2.

6 Algoritmo de otimização

6.1 O algoritmo *Firefly*

A otimização deste problema pode ser feita com recurso a muitos algoritmos de otimização diferentes. Tendo em consideração que uma das funções objetivo apresenta degraus, optou-se por não utilizar métodos baseados no gradiente da função. Métodos heurísticos, apesar de nem sempre convergirem com exatidão no mínimo da função, permitem encontrar o ótimo global mesmo em funções com degraus, não contínuas.

O algoritmo *Firefly* é um método heurístico baseado na ideia de inteligência de enxames (*Swarm Intelligence*) [3], em particular, de pirilampos. O método assume que cada pirilampo é atraído por todos os outros, sendo essa atracão proporcional ao seu brilho e inversamente proporcional à distância entre os pirilampos. No contexto de otimização, o brilho do pirilampo é determinado pela função objetivo [4].

As posições iniciais dos pirilampos são geradas aleatoriamente e a deslocação de um pirilampo gera novas soluções.

Tabela 1. Pseudo-código do Algoritmo *Firefly*

```
x = rand(x_min, x_max) #inicializar populacao
I = f(x)
for t in range(0, iteracao_max):
    for i in range(1, populacao):
        for j in range(1, populacao):
            if I[j] < I[i]:
                x_novo = ... # mover pirilampo i na
                     direcao j segundo a equacao 8
                I[i] = f(x_novo)
    # guardar o otimo global
    alpha = alpha * damp
```

6.2 Pseudo-código do algoritmo

O pseudo-código do algoritmo encontra-se na Tabela 1 [4, 5].

Em cada avaliação, a nova posição do pirilampo i, x_i^{novo}, é dada por:

$$x_i^{\text{novo}} = x_i^{\text{anterior}} + \beta_0(x_j - x_i)e^{-\gamma d^2} + \alpha L \cdot \text{rand}(-1,1) \quad (8)$$

onde β_0 é a atração entre pirilampos à distância 0, γ é o coeficiente de absorção, $d = ||x_j - x_i||$ é a distância entre os pirilampos, α é o fator de aleatoriedade e L é a escala do problema [2, 4].

6.3 Penalidades externas

O algoritmo *Firefly* assume que o problema não tem restrições. Uma vez que os problemas em estudo têm restrições, é necessário transformá-los em problemas sem restrições. Isto pode ser feito aplicando penalizações às funções objetivo, como se pode ver em:

$$\begin{aligned} &\text{Procurar } \mathbf{x} \text{ de modo a :} \\ &\quad \text{minimizar } F(\mathbf{x},r_g) = f(\mathbf{x}) + P(\mathbf{x},r_g,r_h), \\ &\quad \text{sujeito a } x_i^{\min} \le x_i \le x_i^{\max}, \end{aligned} \quad (9)$$

sendo $P(\mathbf{x},r_g,r_h)$ a penalização da função objetivo e $f(\mathbf{x})$ a função objetivo original.

No problema da impressão 3D, as penalizações podem ser calculadas de acordo com :

$$P(\mathbf{x},r_g,r_h) = r_g \cdot \sum_{j=1}^{m} [\max\{0,g_j(\mathbf{x})\}]^2 + r_h \cdot (h(\mathbf{x}))^2 \quad (10)$$

sendo r_g e r_h os fatores de penalização. No caso do problema da *internet* 5G, o valor de penalização é fixo para cada torre que não seja posicionada numa zona permitida, ou seja:

$$P(\mathbf{x},r_g) = r_g \cdot n((x_n,y_n) \notin A_{\text{permitida}}) \quad (11)$$

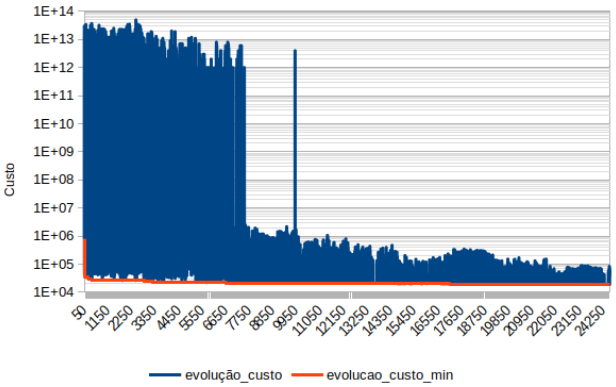

Figura 3. Evolução do custo mínimo da função objetivo da otimização de impressão 3D

Tabela 2. Valores ótimos para as translações e rotações das peças.

Peça	x_t	y_t	θ
1	-63,27	-35,56	3,78
2	-36,23	-37,49	3,59
3	-46,37	-8,73	5,60
4	-21,17	1,99	2,11

7 Resultados

Uma vez que os resultados destes problemas são desconhecidos, os parâmetros do algoritmo *Firefly* tiveram que ser ajustados e estudados com um problema com um mínimo conhecido, o Benchmark 2020, apresentado no apêndice A.

7.1 Impressão 3D

Os resultados foram obtidos com 4 objetos distintos uma cama quente de 200×200 mm, tendo os testes sido efetuados com uma população de 75 e 10 iterações.

Na Figura 3 é possível observar a evolução do custo e custo mínimo ao longo das avaliações, verificando-se a estabilização do custo mínimo. Observa-se também que nas avaliações há varias posições com custos extremamente elevados, o que se deve ao facto de essas posições não cumprirem as restrições impostas e serem penalizadas por tal.

Após 24528 avaliações da função objetivo, obteve-se o valor ótimo final de 18888,55, estando os valores finais de translação e rotação de cada peça na Tabela 2.

7.2 Internet 5G

Após efetuadas 37863 avaliações da função objetivo, com uma população de 100 e 10 iterações, obteve-se o gráfico da Figura 5. O valor do custo ao longo das

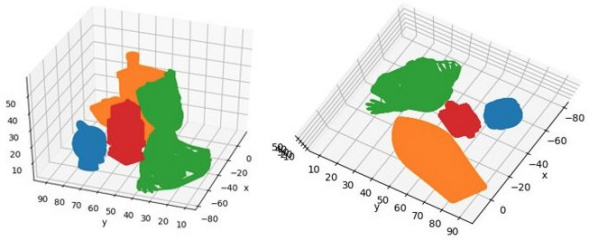

Figura 4. Peças dispostas de forma ótima.

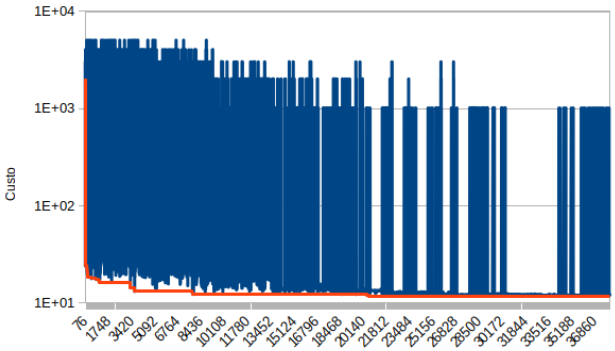

Figura 5. Evolução do custo mínimo da função objetivo da otimização da distribuição de torres 5G

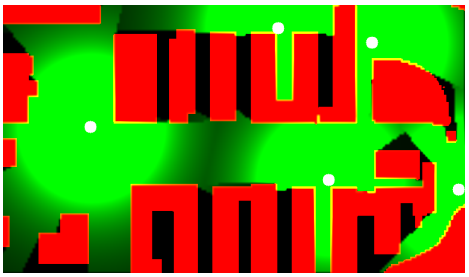

Figura 6. Disposição ótima de 5 torres 5G no espaço.

avaliações apresenta elevadas oscilações. Efetivamente, sendo a função objetivo extremamente sensível a mudanças das variáveis de otimização, a variação de uma delas por apenas um fator de 1 pode colocar a torre numa zona restrita, penalizando a função objetivo, o que provoca as oscilações observadas. No entanto, pode-se observar uma diminuição e estabilização do valor mínimo da função objetivo.

Efetuados vários ensaios, para 5 torres, para o mapa da Figura 1, uma potência de -33 dBW [7] e uma intensidade mínima de -67 dBm [6], obteve-se, após 37863 avaliações, o valor de menor custo de 11,73, associado à melhor posição apresentada na Tabela 3. A Figura 6 mostra, de forma mais visual, o resultado obtido.

Efetuando os mesmos testes, com a mesma população e número de iterações, para os casos de 3, 7 e 9 torres, os valores de custo obtidos são os apresentados na Tabela 4.

Tabela 3. Valores ótimos para as posições das torres 5G.

Torre	x_t	y_t
1	38	51
2	155	16
3	190	77
4	137	74
5	116	11

Tabela 4. Valor da função objetivo em função do número de torres 5G.

nº Torre	custo
3	18,41
5	11,73
7	9,39
9	8,83

8 Conclusões

Os resultados obtidos após a resolução destes dois problema de otimização não-linear de escala intermédia foram bastantes satisfatórios.

No problema da impressão 3D, observa-se, como esperado, uma aproximação das peças, com as restrições cumpridas. No entanto, o gráfico da Figura 3 mostra que os resultados não convergiram totalmente. Para tal acontecer, seria necessário que fossem feitas mais iterações. Outro aspeto a considerar é o facto que o valor em análise foi obtido com uma população de 75 indivíduos, em vez de 100, uma vez que os recursos computacionais disponíveis não permitiram um número de avaliações suficientemente elevado, com uma população de 100, o que permitiria obter um resultado melhor.

Para os vários valores obtidos na Tabela 4, pode-se visualizar que o aumento número de torres aumenta a área coberta. No entanto, como a diferença entre a área coberta por 5 torres e por 9 torres é de apenas 2,9%, não compensaria investir em mais de 5 torres, daí terem sido apresentados resultados mais detalhados para 5 torres. Estes resultados foram bastante satisfatórios, sendo a área com cobertura 5G cerca de 90% da área total.

A Benchmark 2020/2021

A.1 Enquadramento do benchmark

O problema da minimização do peso de um redutor de engrenagens paralelas é um problema benchmark conhecido em otimização não-linear. Sendo um problema não linear de elevado grau de complexidade e com restrições, e sendo conhecida uma solução ótima, é ideal para testar o desempenho de novos algoritmos de otimização.

Com a resolução do Benchmark 2020, pretende-se estudar o desempenho do algoritmo *Firefly*, um algoritmo inspirado na natureza,baseado no conhecido Particle Swarm Optimization.

O problema em questão tem como objetivo minimizar o peso de um redutor de engrenagens paralelas de um andar, modificando sete variáveis [1], estando estas sujeitas a determinadas restrições.

A.1.1 Formulação do problema de otimização

O problema de otimização apresentado, pode ser formulado como [1]:

Procurar $\mathbf{x}$ de modo a:

$$\begin{aligned}\text{minimizar } f(\mathbf{x}) &= 0.7854x_1x_2^2 \\ &\times (3.3333x_3^2 + 14.9334x_3 - 43.0934) \\ &- 1.508x_1(x_6^2 + x_7^2) \\ &+ 7.477(x_6^3 + x_7^3) \\ &+ 0.7854 * (x_4x_6^2 + x_5x_7^2), \end{aligned} \tag{12}$$

$$\begin{aligned}\text{sujeito a } g_1(\mathbf{x}) &= \frac{27}{x_1x_2^2x_3} - 1 \leq 0, \\ g_2(\mathbf{x}) &= \frac{397.5}{x_1x_2^2x_3} - 1 \leq 0, \\ g_3(\mathbf{x}) &= \frac{1.93x_4^4}{x_2x_3x_6^4} - 1 \leq 0, \\ g_4(\mathbf{x}) &= \frac{1.93x_4^4}{x_2x_3x_6^4} - 1 \leq 0, \\ g_5(\mathbf{x}) &= \frac{\sqrt{\left(\frac{745x_4}{x_2x_3}\right)^2 + 16.9E6}}{110x_6^3} - 1 \leq 0, \\ g_6(\mathbf{x}) &= \frac{\sqrt{\left(\frac{745x_5}{x_2x_3}\right)^2 + 157.5E6}}{85x_7^3} - 1 \leq 0, \\ g_7(\mathbf{x}) &= \frac{x_2x_3}{40} - 1 \leq 0, \\ g_8(\mathbf{x}) &= \frac{5x_2}{x_1} - 1 \leq 0, \\ g_9(\mathbf{x}) &= \frac{x_1}{12x_2} - 1 \leq 0, \\ g_{10}(\mathbf{x}) &= \frac{1.5x_6 + 1.9}{x_4} - 1 \leq 0, \\ g_{11}(\mathbf{x}) &= \frac{1.1x_7 + 1.9}{x_5} - 1 \leq 0, \end{aligned} \tag{13}$$

e com $2.6 \leq x_1 \leq 3.6$,
$0.7 \leq x_2 \leq 0.8$,
$17.0 \leq x_3 \leq 28.0$,
$7.3 \leq x_4 \leq 8.3$,
$7.8 \leq x_5 \leq 8.3$,
$2.9 \leq x_6 \leq 3.9$,
$5.0 \leq x_7 \leq 5.5$, (14)

Tanto a função objetivo como as restrições são funções não-lineares. Assim sendo, este problema é classificado com um problema de otimização não linear, de escala intermédia (uma vez que há mais de 5 variáveis de otimização), com variáveis de natureza determinística, seis continuas e uma inteira.

A.1.2 Análise de sensibilidade

De modo a melhor compreender o problema, efetuou-se uma análise de sensibilidade. Com uma análise de sensibilidade "um-de-cada-vez", foi possível observar que a função objetivo é contínua, o que pode ajudar a seleccionar o método de otimização mais apropriado. Neste caso, seria possível a utilização de métodos baseados no gradiente da função, uma vez que a análise de sensibilidade mostra que a função é diferenciavel.

A.2 Método de otimização e Implementação

A otimização deste problema pode ser feita com recurso a muitos algoritmos de otimização diferentes. O algoritmo em questão foi escolhido tendo em consideração os dois problemas referidos anteriormente, cuja solução é desconhecida e se pretende otimizar. Como analisado na Secção 4, a função objetivo do problema da otimização das torre 5G apresenta degraus, não sendo possível calcular o seu gradiente.

Assim sendo, optou-se por não utilizar métodos baseados no gradiente da função. Métodos de procura direta também não são uma possibilidade, uma vez que são muito pouco eficientes em problemas com elevado grau de complexidade.

O método escolhido foi a heurísitca *Firefly*, já descrita na Secção 6. Tanto o algoritmo de otimização como o problema foram programados em linguagem de programação *Python*.

A.2.1 Penalizações externas

O algoritmo *Firefly* assume que o problema não tem restrições. Uma vez que o problema em estudo tem restrições de desigualdade, é necessário transformá-lo num problema sem restrições. Isto pode ser feito aplicando penalizações à função objetivo, a qual passa a ser descrita pela equação 9, sendo $r_\mathrm{h} = 0$.

A.3 Resultados

Para obter resultados fiáveis usando o algoritmo escolhido, foi necessário ajustar os parâmetros do algoritmo.

Como este algoritmo, para este problema, converge extremamente rápido, usou-se apenas 10 iterações para uma população de 100 indivíduos, efetuando, em média, 42 000 avaliações, uma vez que em cada iteração há um número de

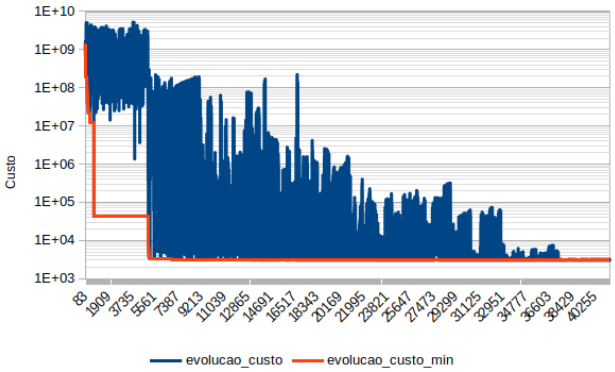

Figura 7. Custo em cada avaliação da função objetivo

iterações aproximadamente igual a metade do quadrado do tamanho da população.

Os parâmetros escolhidos para o melhor resultado obtido foram: $\gamma = 1$ de forma a convergir mais depressa, menos iterações, compensado assim o valor do damp = 0,4, cuja redução diminui também a velocidade de convergência, $\alpha = 0{,}1$, de forma a que os valores aleatórios da distribuição normal não sejam demasiado distantes dos valores calculados na iteração anterior e, por fim, tem-se o valor de $\beta = 1{,}8$, que, como os outros valores, foi obtido através de vários testes.

Na Figura 7, pode-se ver que, ao fim de 25000 a 30000 avaliações, já se tem valores relativamente próximos do valor ótimo e pode ser útil ou não fazer mais avaliações, dependendo da precisão requerida.

O melhor custo obtido foi de 2996,349406 com uma posição de x =[3,500001; 0,7; 17;7,3;7,8;3,350216; 5,286684] para 41353 avaliações, bastante próximo da literatura que é de 2996,348165 com a posição de x = [3,499999; 0,7; 17; 7,3; 7,8;3,350215; 5,286683], com 10^5 avaliações [1].

Utilizando o algoritmo *Firefly* na resolução de um problema de otimização *benchmark*, foi possível avaliar o seu desempenho e exatidão de resultados, além de ganhar uma maior compreensão dos parâmetros dos quais o algoritmo depende. Em comparação com o melhor valor obtido noutros estudos deste problema, o resultado obtido nestes ensaios apresenta um erro de apenas 0,0000416%. Trata-se de um algoritmo com uma muito boa exatidão de resultado para este problema. Além disso, sendo um algoritmo que converge depressa, caso seja necessário reduzir o custo computacional, será possível diminuir o número de avaliações (diminuindo a população ou o número de iterações) sem comprometer demasiado o resultado final.

Após efetuados 50 testes, sempre com os mesmos parâmetros, o resultado ótimo médio foi de 3008,16, 0,394% acima do resultado mínimo obtido, sendo o desvio padrão deste conjunto de resultados de 80,4787, ou seja, 2,920% do melhor resultado. Com estes valores, pode-se concluir que se trata de um algoritmo de elevada robustez.

Referências

1. A. Andrade-Campos, J. Dias-de-Oliveira, Benchmark 2020: Weight minimisation of a speed reducer, *Non-Linear Optimisation in Enginering / MIEM.2020*, 2020.
2. Yang, Xin-She & Deb, Suash. (2010). Eagle Strategy Using Lévy Walk and Firefly Algorithms for Stochastic Optimization. Studies in Computational Intelligence.
3. Chakraborty, Amrita & Kar, Arpan. (2017). Swarm Intelligence: A Review of Algorithms
4. Firefly Algorithm - an overview | ScienceDirect Topics. (2020). Retrieved 14 December 2020, from https://www.sciencedirect.com/topics/computer-science/firefly-algorithm
5. WiraDKP/Firefly-Algorithm. (2020). Retrieved 17 December 2020, from https://github.com/WiraDKP/Firefly-Algorithm
6. Welkom bij Random Solutions Support, "Best dBm Values for Wifi," Randomsolutions.nl, Retrieved 14 December 2020, from https://support.randomsolutions.nl/827069-Best-dBm-Values-for-Wifi
7. Global 5G wireless deal threatens weather forecasts. (2019). Retrieved 14 January 2021, from https://www.nature.com/articles/d41586-019-03609-x

Otimização Não-Linear em Engenharia | MIEM.2021
(artigo/relatório de projecto)

Otimização de um sistema de bombagem e de um processo vinícola

Resolução de dois problemas lineares de engenharia

Mariana Lopes · Teresa Gonçalves

Submetido: 04/03/2021

Resumo Dois problemas de engenharia são apresentados e pretende-se otimizar um sistema de bombagem com objetivo de minimizar o custo associado para um consumo fixo e otimizar as percentagens de mercado para os produtos de uma empresa vinícola. Para além da resolução destes dois problemas, um caso de benchmark fornecido pelos docentes também foi resolvido.

Palavras-Chave Otimização linear · Otimização não linear · PSO · Sistema de bombagem · Processo Vinícola

1 Introdução

No âmbito da unidade curricular de Otimização Não-Linear em Engenharia, foi proposto pelos docentes a elaboração de dois problemas de engenharia e a sua resolução através de métodos lecionados. O trabalho foi dividido em várias etapas e apresentações. Começou-se por definir o problema, depois as variáveis do problema, as várias de decisão, as restrições e a função objetivo. O objetivo é a elaboração completa do problema e a aplicação de este num algoritmo matemático que o possa resolver.

Mariana Lopes
n.º 80575
E-mail: marianahlopes@ua.pt

Teresa Gonçalves
n.º 80512
E-mail: teresagoncalves@ua.pt

2 Apresentação dos problemas de otimização

2.1 Sistema de bombagem

O primeiro problema visa a minimização do custo de um sistema de 3 bombas. Estas 3 bombas encarregam-se da distribuição da água de um furo (bomba 1) e de um poço (bomba 2) para que um consumo de água na casa e na rega seja satisfeito. Existem 2 reservatórios de armazenamento. Todas os dados relativamente às bombas, ao reservatório e ao custo da eletricidade são conhecidas.

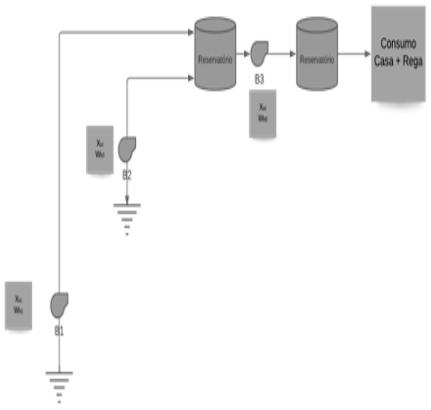

Figura 1. Esquema do reservatório

Os reservatórios têm uma altura de 2,59 m e uma área de 4.05405 m^2. Entre a bomba 1 e o reservatório 1 existe uma diferença de altura de 128 m, entre a bomba 2 e o reservatório 1 existe uma variação de altura de 18m. A diferença de alturas entre o reservatório e a bomba 3 é de apenas 2,59m. Por razões de segurança, o

reservatório exige valores de altura mínimo e máximo, sendo estes de 0,02m e 2,59m, respetivamente. O reservatório 2 abastece um consumo de água de uma casa e de um sistema de rega. Este consumo está referenciado na seguinte tabela.

Tabela 1. Consumo do sistema

T(h)	Consumo (m^3/h)
0	4,8
1	2,88
3	2,4
4	1,28
5	1,28
6	1,28
7	0,24
8	0,24
9	0,16
10	0,16
11	2,24
12	1,84
13	3,44
14	2,16
15	0,88
16	0,48
17	0,56
18	2,4
19	6
20	5,2
21	3,6
22	3,2
23	4

As três bombas, todas com eficiência de 75%, respeitam as curvas hidráulicas seguintes

$$\begin{cases} h_{1,i} = -8{,}6904Q_{1,i}^2 - 0{,}4687Q_{1,i} + 132{,}86[m] \\ h_{2,i} = -1{,}5711Q_{2,i}^2 - 1{,}4088Q_{2,i} + 88{,}899[m] \\ h_{3,i} = -1{,}0462Q_{1,i}^2 - 1{,}0462Q_{3,i} + 63{,}32[m] \end{cases}$$

Os caudais estão em m3/h. De referenciar que 5% de perdas foram consideradas por parte das condutas. O tarifário varia ao longo do dia e é enunciado na tabela seguinte.

Tabela 2. Consumo do sistema

T(h)	Consumo (euros/kwh)
0 a 2	0,0737
2 a 6	0,06618
6 a 7	0,0737
7 a 9	0,10094
9 a 12	0,18581
12 a 24	0,10094

Considerando as curvas hidráulicas das 3 bombas, e sabendo a altura manométrica de operação, calculou-se o caudal de operação.

Se $h_{j,i} = h_{operacao,j} + 0{,}05 * h_{operacao,j}$, então

$$\begin{cases} h_{1,i} = 128{,}28[m] \\ h_{2,i} = 20{,}7959[m] \\ h_{3,i} = 1{,}01[m] \end{cases}$$

Visto isto, e através das curvas hidráulicas,

$$\begin{cases} Q_{1,i} = 0{,}0021[m^3/h] \\ Q_{2,i} = 6{,}15[m^3/h] \\ Q_{3,i} = 7{,}24[m^3/h] \end{cases}$$

Admitindo que a potência das bombas é constante e, sabendo que é calculada através de $W_{j,i} = \rho * g * Q_{j,i} * h_{j,i}$, temos que

$$\begin{cases} W_{1,i} = 0{,}000737423[kW] \\ W_{2,i} = 0{,}3473912[kW] \\ W_{3,i} = 0{,}01986214[kW] \end{cases}$$

2.2 Processo vinícola

O primeiro problema visa aumentar os lucros de uma empresa vinícola, otimizando as as vendas de mercado para seus produtos (vinho, sumo e uva). Este processo é o cálculo para um ano de rendimentos. Têm-se como objetivo de produção no mínimo 11000 euros com o mínimo de 1000 garrafas de vinho vendidas.

Temos como constantes do problema:

- Quantidade de uvas: 10000 kg
- Preço de cada um dos produtos: 2,5 euros para o vinho, 1 euro para o sumo e 1,5 euro/quilo a uva)
- Quantidade de uvas que existem em cada garrafa de vinho e sumo: 1,2 kg por garrafa; 0,5 kg por sumo
- Preço de cada uma das suas embalagens: 0,65 euros para vinho; 0,1 euros para sumo
- Custo da apanha por quilograma: 0,4 euros/kg
- Custo da mão-de-obra: 4 euros/h/pessoa
- Número de pessoas: 10 pessoas

3 Formulação dos problemas de otimização

3.1 Sistema de bombagem

O objetivo é encontrar a operação das 3 bombas que minimizem o custo do sistema de bombagem. Admitindo que as bombas são bombas hidráulicas simples, sem variador de velocidade, a operação das bombas resume-se

a encontrar os períodos de atividade da bomba. Assim, a variável de decisão x toma a seguinte configuração:

3.1.1 Variável de decisão

Variável binária que representa o estado da bomba (on/off) num intervalo de tempo [0, 24] h.

$$x_{i,j} = \begin{bmatrix} x_{1,1} & x_{2,1} & x_{3,1} \\ ... & ... & ... \\ x_{1,n} & ... & x_{3,n} \end{bmatrix}$$

3.1.2 Função objetivo

O custo de operação da bomba é função do tarifário energético, das potências das bombas, das suas eficiências, do estado e tempo de operação da bomba. Temos então que o custo é dado por:

$$C(x) = \sum_{j=1}^{3} \sum_{i=1}^{n} \frac{W_{j,i}}{n_j} * x_{j,i} * \Delta t_i * Tarifario$$

Onde Wij representa a potência fixa da bomba j no incremento de tempo i, nj a eficiência da bomba j, ti variação de tempo i e Tarifario(t) o custo energético ao longo do tempo.

3.1.3 Restrições

Para além das restrições do universo das variáveis, as restrições do problema resumem-se à limitação das alturas dos dois depósitos:

$$0{,}02 \leq N1_i \leq 2{,}59$$

$$0{,}02 \leq N2_i \leq 2{,}59$$

Estas restrições têm de ser satisfeitas ao longo de todo o dia. Para um qualquer intervalo de tempo i,

$$\text{gi}= \begin{bmatrix} g_{1,i} \\ g_{2,i} \\ g_{3,i} \\ g_{4,i} \end{bmatrix} = \begin{bmatrix} 0{,}02 - N_{1,i} \\ N_{1,i} - 2{,}59 \\ 0{,}02 - N_{2,i} \\ N_{2,i} - 2{,}59 \end{bmatrix} \leq \begin{matrix} 0 \\ 0 \\ 0 \\ 0 \end{matrix}$$

Para calcular os níveis do depósito, através do balanço mássico, temos:

$$Q_{Deposito_1} = Q_{in} - Q_{out}$$

$$\frac{dV_{Deposito_1}}{dt} = Q_1 + Q_2 - Q_3$$

$$A_{Deposito_1} * \frac{dN1}{dt} = Q_1 + Q_2 - Q_3$$

Considerando que os incrementos de tempo

$$\Delta N1 = \frac{\Delta t}{A_{Deposito_1}}(Q_1 + Q_2 - Q_3)$$

$$N1_{1+i} = Ni \frac{\Delta ti}{A_{Deposito_1}}(Q_{1,i} + Q_{2,i} - Q_{3,i})$$

Pela mesma lógica, para o segundo depósito tem-se que:

$$N2_{1+i} = N_i \frac{\Delta t_i}{A_{Deposito_2}}(Q_{3,i} * x_{3,i} - Q_{consumo})$$

Assim sendo, o problema pode ser formulado como:
Procurar o estado/tempo x de operação das bombas de forma a

Minimizar C(X)

Sujeito a $g_i \leq 0$,*i=1,. . . ,24*

3.2 Processo vinícola

O objetivo é encontrar a solução dos três rendimentos de maneira a que a sua soma seja 1. Assim, a variável de decisão está exposta abaixo.

3.2.1 Variável de decisão

Variável que representa o valor total dos lucros, subtraindo os custos, L.

3.2.2 Função objetivo

$L(x)$=Total das vendas - Custo total associado a todo o processo vinícola

$$L(x) = (V_{uva} + V_{sumo} + V_{vinho}) - (C_{uva} + C_{sumo} + C_{vinho}$$

$$L(x) = ((P_{uva} * X_{uva}) + (P_{sumo} * X_s) + (P_{vinho} * X_g)) - ((Ca * (N - UG * X_g - US * X_s) + (X_s * (C_s + US * C_a) + (X_g * (C_g + UG * C_a))$$

$$L(x) = (((P_{uva} * (N - UG * X_g - US * X_s)) + (P_{sumo} * X_s) + (P_{vinho} * X_g)) - (((C_p * \frac{np}{np * U}) * (N - UG * X_g - US * X_s) - (X_s * (C_s + US * C_a)) - (X_g * (C_g + UG * C_a))$$

Onde:

- P_{uva}: Preço da uva [euros]
- N: Quantidade de uvas [kg]
- UG: Conversão de uvas para garrafa [kg/garrafa]
- X_g: Quantidade de vinho [garrafas]
- US: Conversão de uvas para sumo [kg/sumo]
- X_s: Quantidade de sumo [garrafas]
- P_{sumo}: Preço do sumo [euros]

– P_{vinho}: Preço do vinho [euros]
– Ca: Custo da apanha [euros/kg]
– Cp: Custo da mão de obra [euros/h/pessoa]
– Np: Número de pessoas
– U: Apanha por hora[kg/h/p]
– Cg: Custo da garrafa de vinho vazia [euros]

3.2.3 Restrições

Para além das restrições do universo das variáveis, as restrições do problema resumem-se às percentagens terem de ser:

$$X_g \geq 1000$$

$$X_g > 0$$

$$X_s > 0$$

$$X_{uva} > 0$$

Assim sendo, o problema pode ser formulado como:

Procurar as percentagens de venda dos produtos de modo a

Maximizar L(x)

4 Análise de sensibilidade

4.1 Sistema de bombagem

Uma análise de sensibilidade procura determinar o efeito de uma variação de uma variável na função objetivo. Pode ser realizado por métodos aproximados. Aqui, tendo em conta que as funções objetivo e de restrição podem ser escritas analiticamente, as sensibilidades podem ser encontradas através das suas derivadas. A derivada da função objetivo C, para uma variável de decisão alpha é dada por

$$\frac{\partial C}{\partial x_\alpha}(X) = \frac{\sum_1^3 W_{j_\alpha}}{n_j} * Tarifario_\alpha$$

Da mesma forma, as derivadas para as restrições de desigualdade, podem ser escritas como

$$\frac{d_{gi(X)}}{d_{x\alpha}} = \begin{bmatrix} g_{1,i} \\ g_{2,i} \\ g_{3,i} \\ g_{4,i} \end{bmatrix} = \begin{matrix} 0 \\ 0 \\ 0 \\ 0 \end{matrix}, \text{ para } i < \alpha$$

$$\frac{d_{gi(X)}}{d_{x\alpha}} = \begin{bmatrix} g_{1,i} \\ g_{2,i} \\ g_{3,i} \\ g_{4,i} \end{bmatrix} = \begin{matrix} -\frac{\Delta t_i (Q_{1,i}+Q_{2,i}-Q_{3,i}}{A_{Deposito_1}} \\ \frac{\Delta t_i (Q_{1,i}+Q_{2,i}-Q_{3,i}}{A_{Deposito_1}} \\ -\frac{\Delta t_i (Q_{3,i}-Q_{consumo}}{A_{Deposito_2}} \\ \frac{\Delta t_i (Q_{3,i}-Q_{consumo}}{A_{Deposito_2}} \end{matrix}, \text{ para } i \geq \alpha$$

4.2 Processo vinícola

Para a análise de sensibilidade avaliou-se a influência de uma das variáveis de entrada: preço dos produtos no lucro final.

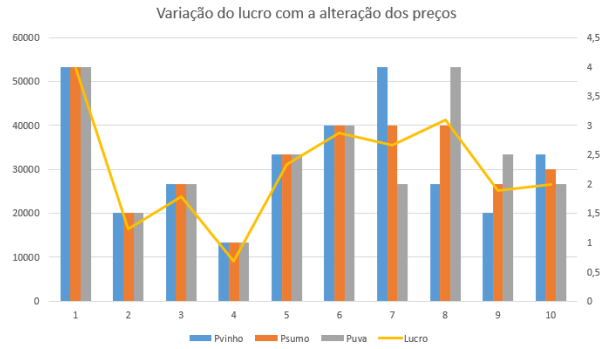

Figura 2. Análise de sensibilidade do problema 2

5 Resultados

5.1 Sistema de bombagem

Para chegar a uma solução ótima utilizou-se um modelo Python que utiliza o algoritmo simplex. Este algoritmo é uma boa escolha para problemas de natureza linear. A figura 4 mostra os resultados para as variáveis **x1, x2 e x3** e os respetivos níveis de água dos reservatórios 1 e 2.

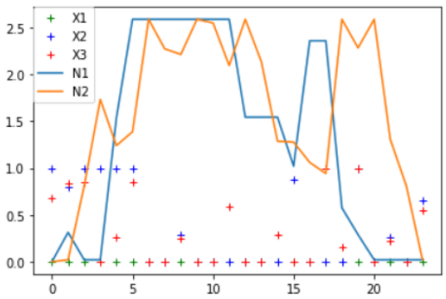

Figura 3. Solução das variáveis de decisão e níveis de depósitos respetivos

O valor ótimo da função objetivo é de 0.32413667897 31545 para um dia de operação. A seguinte figura mostra, para além do anteriormente mostrado, o caudal de consumo necessário a satisfazer e o preço energético.

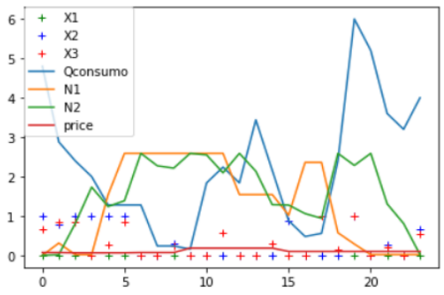

Figura 4. Solução das variáveis de decisão e níveis de depósitos respetivos, incluindo o consumo de água e o seu preço

5.2 Processo vinícola

Para chegar a uma solução ótima utilizou-se o Solver, em Excel, devido à natureza combinatória do trabalho e à sua adaptabilidade ao programa. A seguinte tabela mostra os resultados ótimos de x_g, x_s e x_{uva} mostrando as percentagens de venda associadas aos mesmos.

Tabela 3. Resultados ótimos de x_g, x_s e x_{uva}

Tipo de produto	x	Percentagens de vendas
Vinho	x_g=1000	0,12
Sumo	x_s=9720	0,49
Uva	x_{uva}=3940	0,39

A combinação ótima de percentagens é portanto 12% de vendas de vinho, 49% de venda de sumo e 39% de venda da própria uva.

6 Análise de resultados

6.1 Sistema de bombagem

Analisando o primeiro gráfico, conseguimos compreender que a primeira bomba não liga nenhuma vez ao longo do dia. Isto deve-se ao facto da bomba 2 ter um maior caudal e por isso ser a eleição de escolha no que toca ao enchimento do depósito 1. Apesar da bomba 2 ter uma potência maior e consequentemente ser mais dispendiosa, tem um caudal de funcionamento de 2928x menor o que faz com que não seja suficiente para a satisfação do consumo da casa e rega. Só a bomba 2 torna-se suficiente para o satisfazer. Tudo isto admitindo que não há restrições de volume de água nas fontes, o poço e o furo. Consegue-se concluir também que no período do dia com preço mais elevado de energia, as bombas estão, em maioria, quase ou sempre desligadas. No período inicial do dia, onde a eletricidade tem um custo mais baixo, as bombas estão em funcionamento parcial ou total. Em seguida apresentam-se a solução para a variável de decisão.

$$x_{j,i} = \begin{bmatrix} 0 & 1 & 0{,}674182 \\ 0 & 0{,}795119 & 0{,}839477 \\ 0 & 1 & 0{,}849448 \\ 0 & 1 & 0 \\ 0 & 1 & 0{,}259819 \\ 0 & 1 & 0{,}849448 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0{,}286179 & 0{,}243094 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0{,}585635 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0{,}292818 \\ 0 & 0{,}883104 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0{,}160221 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0{,}257088 & 0{,}218383 \\ 0 & 0 & 0 \\ 0 & 0{,}650406 & 0{,}552486 \end{bmatrix}$$

6.2 Processo vinícola

Analisando os resultados, consegue-se perceber que a maior venda é de sumo, seguindo-se a de uva e a de vinho. Os resultados são expectáveis visto que a produção de vinho é a que envolve mais custos e sendo assim, a menos lucrativa.

Referências

1. A. Andrade-Campos and J.Dias-de-Oliveira, *Benchmark 2020: Weight minimisation of a speed reducer,*, Universidade de Aveiro, 2020
2. A. Andrade-Campos and J.Dias-de-Oliveira *Técnicas modernas,implementação e aplicações*, Universidade de Aveiro, 2018/2019

A Benchmark 2020

Através da Unidade Curricular Opcional de Otimização Não-Linear em Engenharia, foi proposto um problema de Benchmark de maneira a desafiar os alunos a resolverem um problema de engenharia fora do contexto dos seus problemas anteriormente criados. O Benchmark 2020 contempla um problema clássico de Otimização: minimização do peso de um redutor de velocidades. Neste problema será utilizado o software de programação Python e o método de otimização será o continuous PSO com modificação no termo de fator de inércia com objetivo de melhorar a performance do PSO.

A.1 Enquadramento do benchmark

Segundo as guidelines, este problema de benchmark segue a seguinte formulação:

Procurar **x** de modo a:

minimizar f(x) $= 0{,}7854x_1x_2^2 * (3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 + x_7^2) + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2)$

suj. a $g_1(\mathbf{x}) = \frac{27}{x_1x_2^2x_3} - 1 \leq 0$

$g_2(\mathbf{x}) = \frac{397.5}{x_1x_2^2x_3} - 1 \leq 0$

$g_3(\mathbf{x}) = \frac{1.93x_34}{x_2x_3x_3x_64} - 1 \leq 0$

$g_4(\mathbf{x}) = \frac{1.93x_53}{x_2x_3x_74} - 1 \leq 0$

$g_5(\mathbf{x}) = \frac{\sqrt{(\frac{745x_5}{x_2x_3})^2+16.9E6}}{110x_6^3} - 1 \leq 0$

$g_6(\mathbf{x}) = \frac{\sqrt{(\frac{745x_5}{x_2x_3})^2+157.5E6}}{85x_7^3} - 1 \leq 0$

$g_7(\mathbf{x}) = \frac{x_2x_3}{40} - 1 \leq 0$

$g_8(\mathbf{x}) = \frac{5x_2}{x_1} - 1 \leq 0$

$g_9(\mathbf{x}) = \frac{x_1}{12x_2} - 1 \leq 0$

$g_{10}(\mathbf{x}) = \frac{1.5x_6+1.9}{x_4} - 1 \leq 0$

$g_{11}(\mathbf{x}) = \frac{11x_7+1.9}{x_5} - 1 \leq 0$

e com

$2.6 \leq x_1 \leq 3.6$
$0.7 \leq x_2 \leq 0.8$
$17.0 \leq x_3 \leq 28$
$7.3 \leq x_4 \leq 8.3$
$7.8 \leq x_5 \leq 8.3$
$2.9 \leq x_6 \leq 3.9$
$5.0 \leq x_7 \leq 5.5$

A.2 Implementação - PSO

Como foi referido na introdução, o algoritmo escolhido foi o continuous PSO com modificação no termo de fator de inércia. A otimização por aglomerado de partículas é uma técnica de computação evolutiva desenvolvida por Kennedy e Eberhart. Este algoritmo exibe atributos de computação evolutiva comum, incluindo inicialização com uma população de soluções. As soluções potenciais, chamadas de partículas, são então "transportadas"através do espaço do problema, seguindo as partículas ideais atuais. O conceito de aglomerado de partículas foi originado como uma simulação de um sistema social simplificado. Cada partícula tem o controlo da sua posição no espaço do problema, que estão associados à melhor solução alcançada até agora. Este valor é denominado 'pBest'. Outro valor "melhor" seguido pela versão global da otimização do aglomerado de partículas é o melhor valor geral e a sua localização obtida até agora por qualquer partícula na população. Este local é denominado 'gBest'. O conceito de otimização de enxame de partículas consiste em, em cada etapa, alterar a velocidade (ou seja, aceleração) de cada partícula em direção aos seus locais 'pBest' e 'gBest' (versão global do PSO). A aceleração é ponderada por um termo aleatório com números aleatórios separados sendo gerados para aceleração em direção aos locais 'pBest' e 'gBest'.

O seguinte esquema demonstra o raciocínio e implementação do PSO.

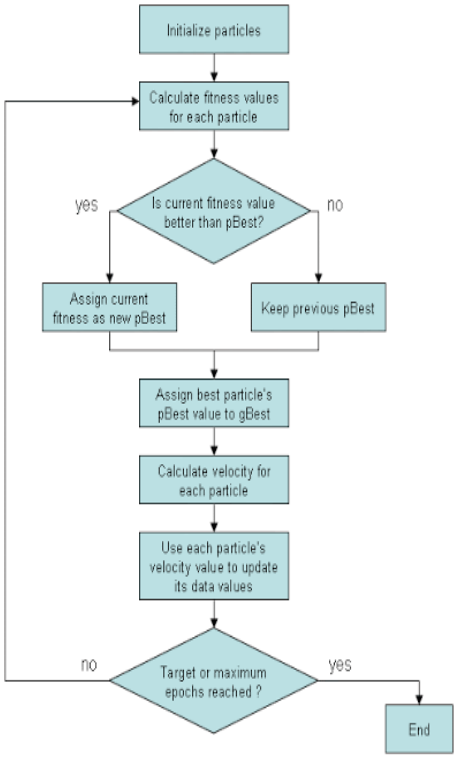

Figura 5. Algoritmo PSO

A.2.1 Equações de velocidade

Para auxiliar no código de programação, utiliza-se as seguintes equações de velocidade:

$v_i(t) = wv_i * (t-1) + r_1c_1(xpbest_i - x_i(t-1))$

$v_i(t) = wv_i*(t-1)+r_1c_1(xpbest_i-x_i(t-1))+r_2c_2(xgbest_i-x_i(t-1))$

$v_i(t) = wv_i*(t-1)+r_1c_1(zpbest_i-x_i(t-1))+r_2c_2(xlbest_k-x_i(t-1))$

A.2.2 Modificações no termo de fator de inércia

O PSO sugerido por Kennedy e Ebarhart não tinha termo de fator de inércia no algoritmo. A adição deste fator foi sugerida pela primeira vez por Shi e Eberhert e foi mostrado que o PSO tem um melhor desempenho com esta introdução. Estes, sugeriram a variação linear do fator de peso usando a seguinte expressão:

$w = ((maxw - minw) * (maxiter - curiter)/maxiter) + minw$

Onde maxw e minw são os valores máximo e mínimo do fator de peso (w), respetivamente. Maxiter é o número máximo de gerações e curiter é a iteração atual. Maxw e minw são considerados 0,9 e 0,4, respetivamente. Xiaohui sugeriu o fator de peso aleatório como:

$w = 0{,}5 + 0{,}5 * (rand)$

Onde rand é um número aleatório entre 0 e 1.

A.3 Resultados e Análise de Resultados

Neste capítulo apresentam-se os resultados variando o número de avaliações e a sua população. Em primeiro apresentou-se os resultados para o PSO continuous e em seguida os resultados para o PSO modificado apresentado no anterior capítulo. Estes programas foram realizados num total de 10 vezes cada um, para que uma estabilização do algoritmo fosse atingida. Apresentam-se também os gráficos da evolução da função objetivo com as avaliações para os dois melhores resultados.

Tabela 4. PSO

População	Avaliações	Função	Tempo
100	10000	3012,078898	0,591464877
200	20000	3001,643387	1,161938548
100	80000	2996,808218	4,535886765
1000	100000	2998,37085	5,476151328
200	160000	2996,547922	9,422259569
1000	800000	2996,402	47,53347015

Tabela 5. PSO modificado

População	Avaliações	Função	Tempo
100	10000	2996,348165	0,567530274
200	20000	3000,960822	1,230192304
100	80000	2996,348165	4,411230326
1000	100000	2996,348165	5,759610534
200	160000	2996,348165	8,711717486
1000	800000	2996,348165	43,11467465

O resultado ótimo em x é de x=[3,5, 0.7, 17, 7.3, 7.8, 3.35021467, 5.2868323] e o respetivo melhor valor da função objetivo é de 2996.34816496853. Analisando os resultados variando o máximo de iterações e de população, concluímos que, para o PSO continuous, como era de esperar, quanto maior o máximo de avaliações, melhor é o resultado. Contudo, mesmo com 800000 avaliações não chegou ao resultado esperado. A modificação no PSO teve um grande impacto para a sua performance. Mesmo com baixas avaliações o resultado convergiu de melhor forma do que o PSO continuous. O resultado da função objetivo é praticamente estável à medida que o número de avaliações aumenta. O melhor resultado da função objetivo é aqui atingido para 10000 avaliações com apenas 0,69 segundos. Sabendo o valor da função objetivo e o valor de 2997,116941, calculamos a oscilação que deu o valor de 0,2 %.

A.4 Conclusão

Após a realização do benchmark, podemos comparar os resultados com os obtidos pelos docentes. Comparando com os resultados obtidos na nossa simulação, podemos observar que a precisão é muito elevada o que nos leva a concluir que o algoritmo escolhido era o indicado para a resolução deste problema. A modificação no PSO conseguiu que houvesse uma melhor performance a nível de tempo de solução. Uma das grandes vantagens do algoritmo é a rapidez da sua convergência e o baixo número de avaliações. A modificação no algoritmo inicial melhorou a performance do algoritmo, visto que, o PSO original, não conseguiu convergir para poucas avaliações. Acreditamos que o objetivo tenha sido atingido e que um melhor conhecimento em algoritmos e formulação de problemas de engenharia tenha sido adquirido.

Referências

1. A. Andrade-Campos and J.Dias-de-Oliveira, *Benchmark 2020: Weight minimisation of a speed reducer,*, Universidade de Aveiro, 2020
2. A. Andrade-Campos and J.Dias-de-Oliveira *Técnicas modernas,implementação e aplicações*, Universidade de Aveiro, 2018/2019
3. I-Ling Lin, *Particle Swarm Optimization for Solving Constraint Satisfaction Problems*, Simon Fraser University, 2002
4. https://www.researchgate.net/publication/333314611/figure/fig2/AS:761735870939136@1558623392061/The-Particle-Swarm-Optimization-PSO-algorithm.png]

Capítulo 3

Comentários Finais

Os trabalhos compilados neste documento são fruto do esforço das/dos estudantes da unidade curricular de ONLE, do MIEM da Universidade de Aveiro. Os conteúdos apresentados foram submetidos para avaliação nesta UC. Com o acordo dos grupos que decidiram participar nesta iniciativa, são assim disponibilizados para referência futura por parte de outros colegas que irão frequentar esta disciplina.

Numa abordagem de partilha de conhecimento, pretende-se amadurecer este conceito. Esta abordagem mostrou o potencial para dar origem a uma série anual de compêndios, devidamente revistos e corrigidos, que possa ser partilhada fora do âmbito da unidade curricular de ONLE. Por mais este passo, nesta segunda edição desta compilação de trabalhos, os docentes agradecem aos estudantes do ano letivo de 2020/2021. Esta é efetivamente uma abordagem integradora de disseminação de trabalho e de conhecimento.

os docentes,

A. Gil Andrade-Campos
João Dias-de-Oliveira

Ficha técnica

Título:	Otimização Não-Linear em Engenharia – Trabalhos e Aplicações 2020/2021
Coordenadores:	A. Gil Andrade Campos, João Dias de Oliveira
Editora:	UA Editora – Universidade de Aveiro
	1ª edição – abril 2021
ISBN:	**978-972-789-690-5**
DOI:	https://doi.org/10.48528/46r9-gv78