Otimização Não-Linear em Engenharia

Trabalhos e Aplicações 2021/2022

universidade de aveiro
theoria poiesis praxis

GRIDS

Esta página foi intencionalmente deixada em branco.

Conteúdo

Esta página foi intencionalmente deixada em branco.

Capítulo 1

Introdução

1.1 Enquadramento

No contexto da Unidade Curricular (UC) de Otimização Não-Linear em Engenharia (ONLE), integrada no primeiro semestre do Mestrado em Engenharia Mecânica (MEM) 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.

Esta página foi intencionalmente deixada em branco.

Benchmark 2022: Problema de estrutura de viga soldada

Minimização do custo e da deflexão final de uma viga soldada

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

Resumo Uma viga sujeita a uma força F na sua extremidade necessita de ser soldada a um outro componente estrutural, de modo a satisfazer as condições de estabilidade e os requisitos de projeto, cujo problema consiste em minimizar o custo e a deflexão final. Ambos os objetivos são concorrentes, isto é, a redução do custo conduz a um maior deslocamento/flecha, levando a que, para diminuir o deslocamento, seja necessário aumentar o material e, consequentemente, o custo.

Existem cinco restrições intrínsecas ao problema, sendo que o desafio deste *benchmark* consiste na resolução da forma mais eficiente e precisa, do ponto de vista da otimização. Os resultados do problema deverão incluir uma análise da evolução iterativa das variáveis de decisão, da(s) função(ões) objetivo e das restrições. Devem ser discutidas as variáveis de projeto e as técnicas de otimização utilizadas.

Palavras-Chave Otimização multiobjetivo · Minimização de custos · Otimização não-linear · *Benchmark*

1 Introdução

Pretende-se neste *benchmark* avaliar o custo e a deflexão final de uma viga que necessita ser soldada a uma estrutura, sujeita a restrições de tensão de corte, tensão à deflexão e carga de encurvadura. Os objetivos passam por minimizar o custo e a deflexão final em simultâneo, sem preferência entre objetivos. O campo da Matemática abordado por este tipo de problema relaciona-se com a otimização multiobjetivo, em particular com o conceito de eficiência de Pareto (ou otimalidade de Pareto). Na otimização multiobjetivo, tal conceito é amplamente aplicado para a determinação da fronteira de Pareto, isto é, o conjunto de todas as alocações eficientes de Pareto, convencionalmente exibidas com recurso a gráficos.

Em qualquer problema de otimização multiobjetivo, na ausência de qualquer preferência entre os objetivos, o objetivo geral será chegar a um conjunto de soluções ótimas de Pareto [1]. Idealmente, o conjunto de soluções de Pareto deve manter uma dispersão uniforme ao longo da fronteira de Pareto, mantendo a diversidade no espaço paramétrico. Um conjunto de soluções diversas no espaço paramétrico permite ao decisor escolher entre um conjunto de projetos alternativos.

Considerando um problema de minimização de dois objetivos, funções f_1 e f_2, por exemplo, é possível melhorar (minimizando) ambos os objetivos, em que, dentro de todas as soluções possíveis, as soluções pertencentes à fronteira Pareto são as melhores, tal como ilustrado na Figura 1.

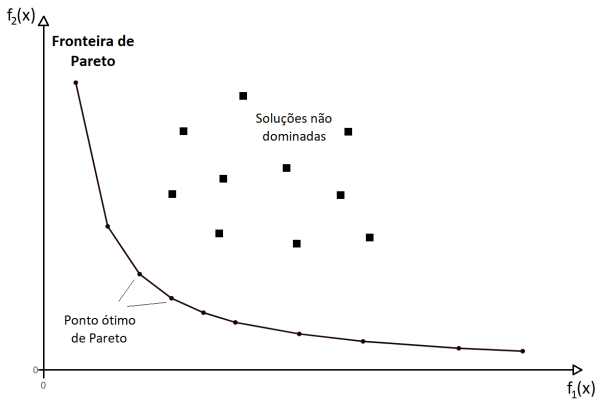


Figura 1. Exemplo de fronteira de Pareto. Adaptada de [2].

M. Rodrigues, A. Andrade-Campos, J. Dias-de-Oliveira
Otimização não-linear em Engenharia
Ano letivo 2021/22
E-mail: monica.joana@ua.pt; gilac@ua.pt; jalex@ua.pt

2 Caso de estudo: componente de teste

O caso de estudo em análise procura a compensação entre a resistência e o custo de uma viga, como tal, o problema consiste na minimização do custo e da deflexão final da viga soldada. O esboço apresentado na Figura 2 representa uma viga soldada a outro componente.

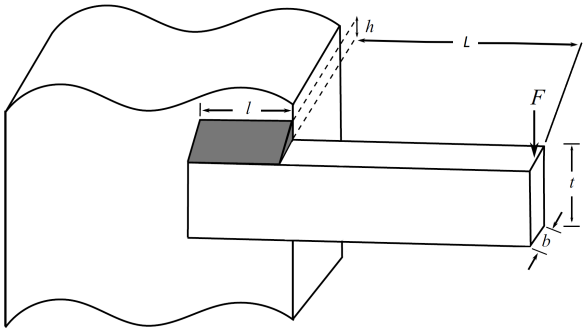


Figura 2. Geometria do problema da viga soldada. Adaptada de [3].

A viga deve suportar a carga F aplicada a uma distância L, com uma flecha máxima admissível de 0,00635 m no ponto de aplicação da carga. As quatro variáveis de projeto são: a espessura e o comprimento do cordão de solda, respetivamente h e l; a largura e a altura da secção retangular da viga, b e t, repetivamente. O material da viga é aço de construção, com um módulo de elasticidade (E) de 200 GPa e módulo de corte (G) de 75,8 GPa.

Como referido, os dois objetivos são a minimização do custo de fabrico e da deflexão da extremidade da viga sob a carga F aplicada. A carga F é de 2722 N e a distância $L = 0{,}3556$ m. Os custos unitários de viga e soldadura são padronizados, sendo que o custo do material, para a fabricação da viga, por unidade de volume, é 2935 €/m^3, e o custo do material para a soldadura, por unidade de volume, é de 67413 €/m^3 .

Neste problema, sabe-se ainda que o custo de fabricação da viga é proporcional à quantidade de material na viga, $(l + L)tb$, e à quantidade de material de solda, lh^2. A deflexão da viga é proporcional a F e inversamente proporcional a bt^3. A tensão de corte da solda não pode exceder 0,09 GPa, a tensão normal nas soldas não pode exceder 0,20 GPa e o limite máximo de deslocamento na extremidade da viga é de 0,0065 m. Existem, também, limites impostos às variáveis. No que concerne a espessura das soldas e a largura da viga, estas não podem ultrapassar os 0,127 m nem deverão ser inferiores a 0,0032 m, respetivamente. O comprimento das soldas e a altura da viga não podem exceder 0,254 m nem ser inferior a 0,0025 m, respetivamente.

A solução final proposta para este problema de *benchmark* deverá ir ao encontro da compensação entre a resistência e o custo de uma viga. De referir ainda que, com vista a promover o desenvolvimento de ferramentas robustas, será fornecido (em tempo oportuno) um novo caso de estudo para resolução e inclusão aquando da entrega final.

3 Formulação do problema de otimização

Tal como referido anteriormente, as variáveis de projeto são:

$x_1 = h$, a espessura das soldas
$x_2 = l$, o comprimento das soldas
$x_3 = t$, a altura da viga
$x_4 = b$, a largura da viga

Portanto, a informação anterior traduz-se no vetor $\boldsymbol{x} = (x_1,x_2,x_3,x_4) = (h,l,t,b)$. Posto isto, o problema consiste em resolver o seguinte:

Minimizar:

$$\begin{cases} f_1(\boldsymbol{x}) = 67413x_1^2x_2 + 2935x_3x_4(L + x_2) & \rightarrow \text{custo} \\ f_2(\boldsymbol{x}) = \dfrac{4FL^3}{Ex_4x_3^3} & \rightarrow \text{deflexão} \end{cases}$$

Sujeito a:

$$\begin{cases} g_1(\boldsymbol{x}) = \tau(\boldsymbol{x}) - \tau_{\max} \leq 0 \\ g_2(\boldsymbol{x}) = \sigma(\boldsymbol{x}) - \sigma_{\max} \leq 0 \\ g_3(\boldsymbol{x}) = F - F_{\text{c}} \leq 0 \\ g_4(\boldsymbol{x}) = \delta(\boldsymbol{x}) - \delta_{\max} \leq 0 \\ g_5(\boldsymbol{x}) = x_1 - x_4 \leq 0 \end{cases}$$

Os limites impostos às variáveis: $0{,}0032 \leq x_1,x_4 \leq 0{,}127$ e $0{,}0025 \leq x_2,x_3 \leq 0{,}254$

Onde $\tau(\boldsymbol{x})$ é a tensão de corte ao longo do apoio da viga; $\tau_{\max}$ é a máxima tensão admissível de corte do material; $\sigma(\boldsymbol{x})$ a tensão normal ao longo do apoio da viga; $\sigma_{\max}$ é a máxima tensão normal admissível do material; e $\delta_{\max}$ é o limite máximo de deslocamento na extremidade da viga.

Tal como referido, existem cinco restrições ao problema. As restrições g_1 e g_2 garantem que a tensão de corte e a tensão normal, desenvolvidas ao longo do apoio da viga, sejam, respetivamente, menores que as tensões admissíveis de corte e normal do material. A restrição g_3 garante que o esforço resistente da viga (ao longo da direção t) seja menor que a carga aplicada F. A restrição g_4 estabelece um limite máximo $\delta_{\max}$ para o deslocamento na extremidade da viga. E, por último, a restrição g_5 visa garantir que a largura da viga não é

menor do que a espessura da solda. De tal modo, as tensões e as restantes variáveis do problema são calculadas como se segue.

Tensão de corte:

$$\tau(\boldsymbol{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2} \tag{1}$$

Tensão normal:

$$\sigma(\boldsymbol{x}) = \frac{6FL}{x_4 x_3^2} \tag{2}$$

Esforço resistente ao longo da direção t:

$$F_c(\boldsymbol{x}) = \frac{4{,}013E\sqrt{\frac{x_3^2 x_4^6}{36}}}{L^2} \times \left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right) \tag{3}$$

Deslocamento:

$$\delta(\boldsymbol{x}) = \frac{4FL^3}{Ex_4 x_3^3} \tag{4}$$

Onde:

$$\tau' = \frac{F}{\sqrt{2}x_1 x_2} \tag{5}$$

$$\tau'' = \frac{MR}{J} \tag{6}$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2} \tag{7}$$

$$M = F\left(L + \frac{x_2}{2}\right) \tag{8}$$

$$J = 2\left\{\sqrt{2}x_1 x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\} \tag{9}$$

Limites e dados do problema:

$F = 2722$ N	$\tau_{\max} = 0{,}09$ GPa
$E = 200$ GPa	$\sigma_{\max} = 0{,}20$ GPa
$G = 75{,}8$ GPa	$\delta_{\max} = 0{,}0065$ m
$L = 0{,}3556$ m	

4 Resultados

Para a resolução deste benchmark de otimização multi-objetivo, em que existem dois objetivos, sendo eles o custo mínimo e flexão mínima, sem preferência entre os mesmos, recorreu-se à fronteira de Pareto para se encontrar um conjunto de soluções. Dentro de todas as soluções possíveis, as que se encontram na fronteira de Pareto são consideradas as melhores.

Para o cálculo da referida fronteira de Pareto, recorreu-se a um código genético que usa padrões de procura num conjunto de pontos de modo a procurar iterativamente pontos não dominados, no que resulta a fronteira de Pareto, sendo que a mesma é o conjunto de pontos dentro do conjunto de pontos de todas as soluções possíveis que não são dominados por qualquer outro ponto do conjunto das soluções. A classificação de um ponto tem uma definição iterativa, os pontos não dominados têm classificação 1 e para qualquer inteiro $k > 1$, um ponto tem classificação k quando os únicos pontos que o dominam têm classificação estritamente menor que k. A procura de padrões satisfaz todos os limites e restrições lineares em cada iteração. Teoricamente, o algoritmo converge para pontos próximos à verdadeira fronteira de Pareto.

Para um melhor conjunto de soluções, escolheu-se 160 pontos a considerar na fronteira de Pareto. O gráfico seguinte mostra a fronteira de Pareto para este problema.

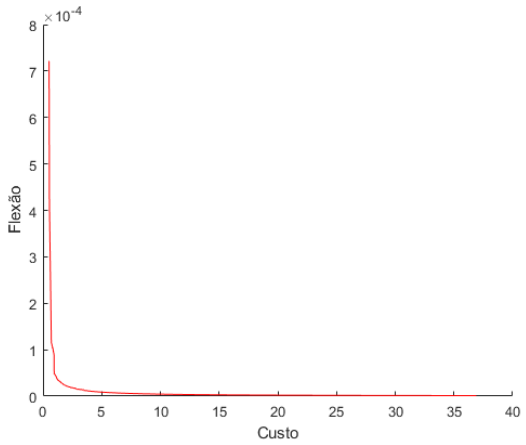


Figura 3. Fronteira de Pareto.

Para melhor interpretação e escolha dos pontos ótimos de Pareto, foram calculadas as soluções que minimizam cada uma das funções objetivo, separadamente. O resultado para o custo mínimo foi de 0,41 €, e de 0,0009 m para a flexão. Para uma flexão mínima, isto é, 0 m, o custo é de 47,97 €. – **J:** 0 m? –

Posteriormente, por forma a melhorar a solução anterior, recorreu-se a uma abordagem diferente da convencional, isto é, utilizou-se uma função híbrida. Trata-se de uma função de minimização que foi executada após o término do algoritmo genético multi-objetivo. Essa função contempla quatro passos, sendo eles o calculo do máximo e do mínimo de cada função objetivo nas soluções, para o objetivo j na solução k; o calculo do peso total em cada solução k; o calculo do peso para

cada função objetivo j em cada solução k; e, por fim, para cada solução k, volta-se a repetir o problema de multi-otimização com vetor objetivo $F_k(j)$ e vetor peso $p(j,k)$.

Como pontos iniciais da função híbrida utilizaram-se as soluções das funções objetivo calculadas separadamente, como referido anteriormente. Os resultados encontram-se representados no gráfico seguinte ampliado para melhor visualização.

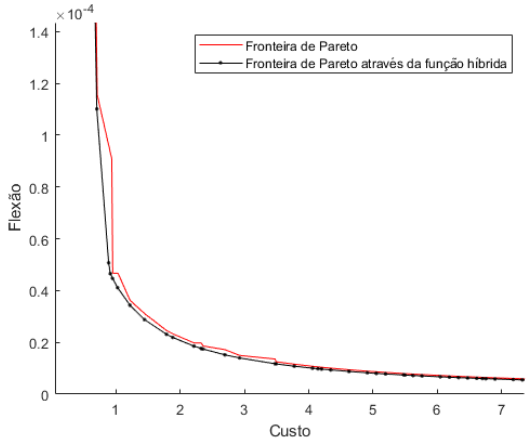


Figura 4. Comparação da fronteira de Pareto com a fronteira de Pareto recorrendo à função híbrida.

Os pontos no gráfico mostram os melhores valores no espaço funcional. Dentro desse conjunto de valores, ir-se-á escolher aqueles que forem mais oportunos para cada situação. Verificando o gráfico, observa-se uma curva mais acentuada em valores inferiores a um custo de 3 € e um flexão inferior a 0,00006 m. A tabela 1 representa um conjunto de soluções entre os valores que se referiu anteriormente, e os valores das respetivas variáveis do projeto.

Tabela 1. Algumas soluções possíveis.

h	l	t	b	Custo	Flexão
0,010683	0,0058222	0,254	0,010683	2,9232	1,3983e-05
0,0080451	0,0078388	0,254	0,0080451	2,214	1,8568e-05
0,006462	0,0098606	0,254	0,006462	1,7883	2,3117e-05
0,0085728	0,0073341	0,254	0,0085728	2,3558	1,7425e-05
0,0043499	0,014939	0,254	0,0043499	1,2206	3,4342e-05
0,0098353	0,0063502	0,254	0,0098353	2,6953	1,5188e-05
0,0068242	0,0093132	0,254	0,0068242	1,8857	2,189e-05
0,0051919	0,012403	0,254	0,0051919	1,4469	2,8773e-05
0,0033435	0,019733	0,254	0,0033435	0,9504	4,4679e-05
0,0036336	0,018066	0,254	0,0036336	1,0283	4,1111e-05
0,0084676	0,0074296	0,254	0,0084676	2,3275	1,7642e-05
0,0068242	0,0093132	0,254	0,0068242	1,8857	2,189e-05

Doze conjuntos de parâmetros atingem um custo inferior a 3 € e uma flexão inferior a 6×10^{-5} m, sendo que no geral em todos os conjuntos apresentados é possível referir que a espessura da solda (h) é ligeiramente superior a 0,003 m, o comprimento da solda (l) cerca de 0,01 m, altura da viga (t) aproxima-se do limite estabelecido, isto é, o valor é aproximadamente 0,254 m, já a largura da viga (b) está aproximadamente entre 0,003 m e 0,01 m. A espessura da solda e a largura da viga apresentam valores semelhantes.

5 Instruções de entrega

Os elementos de avaliação do *benchmark* devem seguir o formato indicado no respetivo template, com um limite de duas páginas.

Referências

1. Deb, K. (2011). Multi-objective optimisation using evolutionary algorithms: an introduction. In *Multi - objective evolutionary optimisation for product design and manufacturing* (pp. 3-34). Springer, London.
2. Liu, X., IJzerman, A. P., & van Westen, G. J. (2021). Computational approaches for de novo drug design: past, present, and future. *Artificial neural networks*, 139-165.
3. Kohli, M., & Arora, S. (2018). Chaotic grey wolf optimization algorithm for constrained optimization problems. *Journal of computational design and engineering*, 5(4), 458-472.
4. Ray, T., & Liew, K. M. (2002). A swarm metaphor for multiobjective design optimization. *Engineering optimization*, 34(2), 141-153.

Otimização Não-Linear em Engenharia | MEM.2122
(artigo/relatório de projecto)

Resolução de problemas de otimização com o algoritmo *Simplex de Nelder-Mead*

Minimização do tempo de corrida em linha reta

André Cabaço

Submetido: 12/7/2022

Resumo O presente documento descreve o estudo e implementação do algoritmo *Simplex de Nelder-Mead* na resolução de problemas no âmbito da U.C. O primeiro problema deriva das preferências do autor: minimizar o tempo de prova de uma corrida em linha reta. O último problema, em anexo, consiste num problema *benchmark* com o intuito de testar e validar o algoritmo selecionado pelos autores.

Palavras-Chave Otimização Não-Linear · *Simplex* · *Nelder-Mead* · *Drag Race*

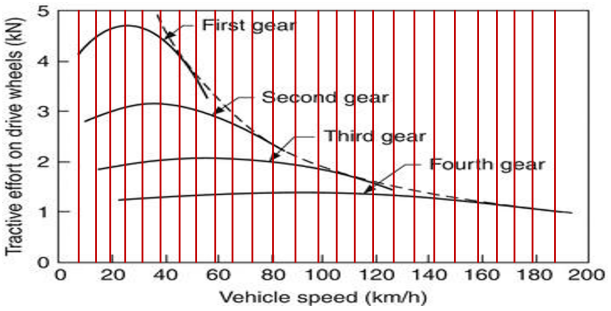


Figura 1. Curva da velocidade linear do veículo × força de tração aplicada as rodas motrizes (discretizada).

1 Introdução

O problema consiste em minimizar o tempo de corrida em linha reta de um veículo. Para o veículo, as relações de transmissão que existem entre o motor e as rodas podem ser preparadas de formas distintas e são um fator determinante na performance numa prova deste tipo. O objetivo que se propõe é determinar os *rapports* ótimos de uma caixa de seis velocidades de um veículo específico para obter o melhor desempenho na aceleração em 400 m. Na figura 1 está ilustrada a forma de como o problema irá ser abordado, discretizando o desenvolvimento ao longo da prova, uma vez que não poderá ser tratado como movimento uniformemente acelerado.

2 Definição de problemas de engenharia

Para definir uma função que devolva o tempo que o veículo demora a percorrer a distância pretendida L e como, ao longo do percurso, o movimento não tem um comportamento constante ao longo de cada *rapport*, divide-se o tempo total em intervalos com aceleração constante e finalmente soma-se, $\sum_{i=1}^{6}\sum_{j=e(i)}^{\mathrm{RPM}}\delta t(i,j)$, em que $\delta t(i,j)$ corresponde ao intervalo de tempo que existe entre sucessivas velocidades lineares do veículo, 6 é o número de *rapports* existentes e RPM é a rotação de troca de caixa, constante para todos os *rapports*. As variáveis de projeto serão o conjunto de *rapports* da caixa de velocidades, ou seja $\mathbf{x} = [x_1\ x_2\ ...\ x_6] \in \mathbf{R}^6$.

A transmissão num veículo deste tipo realiza-se através do *rapport* da caixa de velocidades selecionado e da razão do diferencial. A velocidade a um dado regime de rotações, e com um determinado *rapport* selecionado, depende do diâmetro do pneu da roda motriz, D_{P} e da razão do diferencial, G. Por sua vez, a força resultante no veículo num dado regime e *rapport* depende também de D_{P} e de G, assim como da força de arrasto correspondente à velocidade nesse ponto e do binário, que é uma função $\tau(\beta)$ que discretiza a curva de binário do motor associando valores desta a cada regime de rotação. Se a força aplicada pelo motor for superior à força de arrasto, a aceleração será dada pela força resultante dividida pela massa do veículo, caso contrário será nula.

A. Cabaço
n.º 109514
E-mail: andrecabaco@ua.pt

Como a aceleração em cada período de tempo depende do regime de rotações, definiu-se cada período de tempo como o intervalo entre rotações com um número de divisões considerável, neste caso de 50 em 50 rpm, $t\,(\alpha,\beta)$. De forma a calcular o tempo apenas até à distância pretendida, em cada intervalo, é realizada uma análise a uma função de distância ($d(\alpha,\beta)$), para perceber se, naquele instante, o veículo já ultrapassou a meta. Caso esta função retorne um valor superior a L, o intervalo de tempo correspondente toma um valor nulo.

3 Formulação dos problemas de otimização

Procurar $\mathbf{x} = [x_1\ x_2\ ...\ x_6]$ que minimiza:

$$T(\mathbf{x}) = \begin{cases} \sum_{i=1}^{6} \sum_{j=e(i)}^{\text{RPM}} \delta t(i,j) & \text{se } d(6,\text{RPM}) \geq L \\ \sum_{i=1}^{6} \sum_{j=e(i)}^{\text{RPM}} \delta t(i,j) + t' & \text{se } d(6,\text{RPM}) < L \end{cases}$$

$$\text{sujeito a } g(\mathbf{x}) = x_i - x_{i+1} > 0\ , \quad i = 1,2,...,5\ ,$$
$$0 < x_i \leq 6\ , \quad i = 1,2,...,6\ ,$$

onde:

$$t' = \frac{L - d(6,\text{ RPM})}{v(6,\text{ RPM})}\ ,$$

$$e(\lambda) = \begin{cases} 1050 & \text{se } \lambda = 1 \\ \frac{\text{RPM}\cdot x_\lambda}{x_{\lambda-1}} + 50 & \text{se } \lambda > 1 \end{cases}\ ,$$

$$\delta t(i,j) = \begin{cases} t(i,j) & \text{se } d(i,j) \leq L \\ 0 & \text{se } d(i,j) > L \end{cases}\ ,$$

$$t\,(\alpha,\beta) = \frac{v\,(\alpha,\beta) - v(\alpha,\beta-50)}{a(\alpha,\beta-50)}\ ,$$

$$a(\alpha,\beta) = \begin{cases} \frac{F_\text{R}(\alpha,\beta)}{m} & \text{se } F_R(\alpha,\beta) > 0 \\ 0{,}00001 & \text{se } F_R(\alpha,\beta) \leq 0 \end{cases}\ ,$$

$$F_\text{R}\,(\alpha,\beta) = \frac{2000\tau(\beta)x_\alpha G}{D_\text{P}} - \frac{1}{2}\rho A_\text{F} C_\text{D} v\,(\alpha,\beta)^2\ ,$$

$$v\,(\alpha,\beta) = \frac{D_\text{P}\pi\beta}{6000 G x_\alpha}\ ,$$

$$d(\alpha,\beta) = \begin{cases} \sum_{n=1050}^{\beta} \delta d(\alpha,n) & \text{se } \alpha = 1 \\ \sum_{m=1}^{\alpha-1} \sum_{n=e(m)}^{\text{RPM}} \delta d(m,n) + d' & \text{se } \alpha > 1 \end{cases}\ ,$$

$$d' = \sum_{n=\frac{\text{RPM}\cdot x_\alpha}{x_{\alpha-1}}}^{\beta} \delta d(\alpha,n)\ ,$$

$$\delta d(\alpha,\beta) = v(\alpha,\beta)t(\alpha,\beta) + \frac{1}{2}a(\alpha,\beta)t(\alpha,\beta)^2\ .$$

A função-objetivo é de natureza não-linear, o domínio das variáveis é contínuo $\in \mathbf{R}^6$, o modelo apresenta restrições de desigualdade lineares, de forma a garantir que a aceleração seja maior no início e de domínio também lineares, de modo a que as soluções sejam realistas em relação ao mercado.

4 Análise de sensibilidade

Explorou-se graficamente o modelo deste problema, de forma a verificar o comportamento esperado. Como caso de estudo, analisou-se o comportamento da função objetivo para um BMW E46 M3, ajustando os parâmetros e variáveis de projeto do problema, para os valores reais do caso de estudo [1]. Posteriormente, alteraram-se os *rapports* para dois casos de estudo: uma caixa mais curta e uma mais longa e geraram-se gráficos respetivos das curvas da aceleração em função da velocidade (ver figura 2).

Os resultados foram os esperados, uma vez que, para o primeiro caso, a velocidade desenvolve-se rapidamente, com maior amplitude de aceleração em cada *rapport* e atinge uma velocidade de ponta inferior, enquanto que, para o segundo caso, a velocidade desenvolve-se mais lentamente, sendo atingida uma maior velocidade de ponta.

É de notar que a partir de aproximadente 75 m/s, a curva tem valor de declive nulo, o que indica que, nesse ponto, a força motriz iguala a força de arrasto e a velocidade irá manter-se constante. Compararando os valores da função-objetivo, verifica-se que nenhum dos casos de estudo obteve melhor resultado que o caso real.

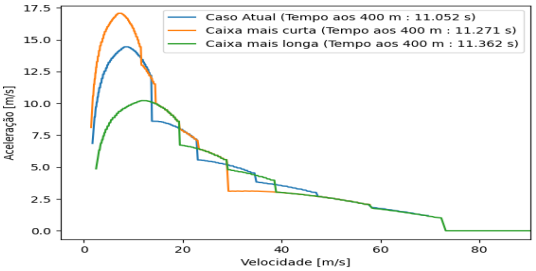


Figura 2. Valor da função-objetivo e curvas de aceleração em função da velocidade para casos de estudo.

5 Avaliação

Para modelar o problema, utilizou-se o *Excel* como primeira abordagem e aprofundou-se o problema em linguagem *Python*. Neste último, definiram-se as funções mencionadas na formulação, sendo que no caso da função objetivo $T(\mathbf{x})$, o ciclo é interrompido quando a distância analisada é superior a L retornando o instante anterior somado a um intervalo de tempo correspondente ao instante em que o veículo atinge L m, resolvendo para um movimento uniformemente acelerado. Discretizou-se a curva de binário do motor correspondente ao caso de estudo [2] e gerou-se um dicionário associando cada binário a uma velocidade angular, tornando o modelo mais exato. Optou-se por um método de penalidades exteriores para restringir a função-objetivo, sendo que as restrições de domínio foram normalizadas pelo *rapport* do caso real correspondente e as restrições de desigualdade foram divididas por uma ordem de grandeza muito inferior, de forma a ampliar o seu efeito. Todas estas, quando ativadas, foram elevadas ao quadrado de forma a conduzir a função progressivamente para a regiâo admissível e multiplicadas por um fator p, selecionado consoante o comportamento do sistema.

6 Algoritmos de otimização

O algoritmo selecionado é o *Método Simplex de Nelder-Mead*, que consiste em gerar uma figura geométrica com $n + 1$ vértices, sendo cada vértice uma solução para o problema e n a dimensão do vetor de variáveis do mesmo. É um método de procura-direta e tem como *inputs* as variáveis iniciais que levarão à geração dos retantes vértices, bem como a distância entre os mesmos (*step*) e coeficientes de reflexão, expansão, contração e redução. Com base na análise da função-objetivo em cada vértice, a solução com o pior valor executa uma sequência de operações, procurando melhorá-lo, o que faz com que a geometria varie de posição e dimensão ao longo das iterações, caminhando para o ótimo. O critério de paragem é o número máximo de iterações em que o melhor valor não varia num determinado intervalo definido como *input*. O autor dispunha de um código base do algoritmo [3] e, para verificar o funcionamento do mesmo, testou-se com funções matemáticas de exemplo, como a função de *Ackley* e uma função *benchmark* com restrições utilizando penalidades exteriores, obtendo-se resultados promissores. O primeiro problema é de natureza contínua, podendo ser resolvido diretamente, mas em termos práticos, sendo a solução um conjunto de *rapports*, resolveu-se adaptar sucessivamente os vértices do *simplex* a soluções com duas casas decimais, de forma a encontrar uma solução ótima mais exequível.

7 Resultados

Sendo um método de procura direta, o algoritmo tem tendência para convergir em mínimos locais. Deste modo, um parâmetro importante a ser testado é o *step*, uma vez que define o domínio de procura. Para tal realizou-se uma análise de sensibilidade, representada na tabela 1, com valores de *step* razoáveis às variáveis iniciais e que não originassem inicialmente violações de restrições.

Tabela 1. Análise de sensibiliadade ao *step*.

Step	Função-Objetivo [s]	Nº Avaliações
0,1	10,994	145
0,25	10,976	159
0,5	10,958	193

O vetor inicial $\mathbf{x}$ é o conjunto de *rapports* do caso real e definiu-se como critério de paragem um máximo de 15 iterações em estagnação com a tolerância de 10^{-3}. O conjunto de seis *rapports* ideal para percorrer 400 m, obtido com um *step* de 0,5 é dado por $\mathbf{x} = [4{,}64;\ 2{,}63;\ 1{,}86;\ 1{,}42;\ 1{,}08;\ 0{,}64]$ e o valor da função-objetivo associado é de 10,958 s, correspondente a uma melhoria de aproximadamente 0,81% em relação ao caso atual, o que dependente do tipo de prova, pode corresponder a uma melhoria significativa ou não. Na figura 3, observa-se a convergência do valor da função-objetivo de todos os vértices do simplex para o critério de paragem escolhido.

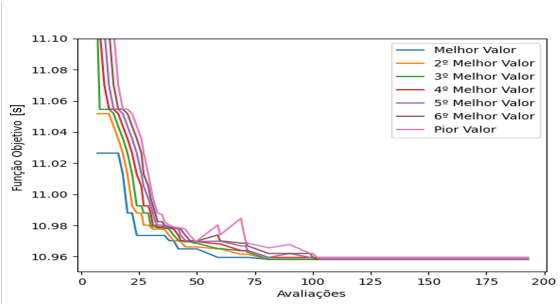


Figura 3. Convergência do *simplex* ao longo das avaliações da função-objetivo.

Seria de esperar que a solução se traduzisse numa forma de atingir a meta com a velocidade máxima, de forma a rentabilizar toda a aceleração, mas para esta solução, o veículo atravessará a meta trocando de caixa

para a 6ª velocidade, ou seja, velocidade máxima da 5ª velocidade, como é ilustrado nos gráficos das figuras 4 e 5. A vermelho estão representadas as condições na chegada. Este resultado, apesar de não esperado, é verosímil e significa que o algortimo encontrou uma forma de minimizar a função-objetivo utilizando apenas 5 relações de transmissão e que uma caixa de 6 não é a mais adequada para uma prova de 400 m.

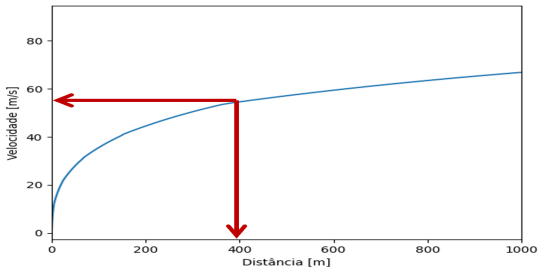


Figura 4. Curva de Distância × Velocidade para a solução encontrada.

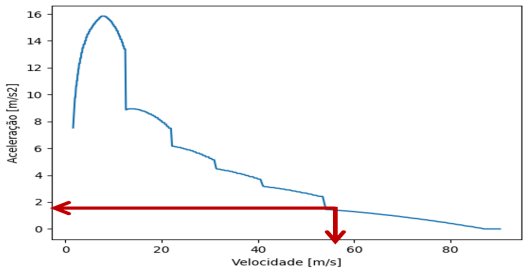


Figura 5. Curva de Velocidade × Aceleração para a solução encontrada.

Para perceber o comportamento do algoritmo ao longo das 193 avaliações, na figura 6 representa-se, em detalhe, a evolução de cada relação de transmissão ao longo do desenvolvimento do algoritmo, sendo x_1 referente à primeira velocidade, x_2 à segunda e assim sucessivamente.

De forma geral, foram obtidos bons resultados, mas deve ser tido em conta que se trata apenas de um modelo e que por isso, terá erros associados quando se traduzir para um caso real, pelo que a sua validação é indispensável. Uma das limitações deste modelo é o facto de o veículo não começar em repouso, partindo da velocidade correspondente às 1000 rpm do primeiro *rapport*. Um objetivo futuro é portanto, contabilizar esta situação e variar também outros parâmetros como a razão do diferencial, dimensão e tipo dos pneus, número

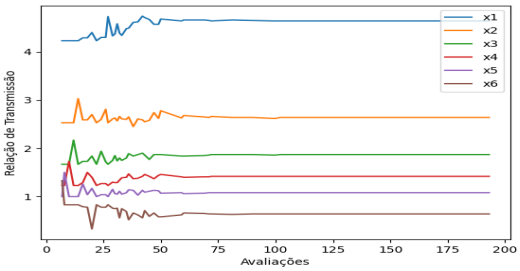


Figura 6. Evolução das variáveis ao longo das avaliações da função-objetivo.

de relações de transmissão, uma vez que foi verificado que para o presente estudo, existem *rapports* desnecessários.

8 Conclusões

O algoritmo *Simplex de Nelder-Mead* mostrou-se eficaz na otimização do problema, chegando a uma solução plausível em poucas avaliações da função objetivo, embora se tenha notado sensível ao valor de *step* utilizado, pelo que deve ser feita uma análise de sensibilidade a este parâmetro quando se trabalha com uma ferramenta deste tipo. Mostrou-se também robusto ao resolver não só este problema, como também um problema multi-objetivo, apresentado posteriormente, em anexo.

A Benchmark 2022

A.1 Enquadramento do benchmark

O problema proposto pelos docentes da UC como *benchmark* trata-se de um problema de um problema de minimização do custo e da defleção de uma viga soldada. Este é portanto um problema multiobjetivo, no qual se pretende minimizar ao mesmo tempo os custos de soldadura e a deflexão.
Sendo as variáveis as dimensões e localização da viga, estas são contínuas e devidamente restringidas num domínio admissível.
Essas dimensões são limitadas por: $0{,}0032 \leq x_1$, $x_4 \leq 0{,}127$ e $0{,}0025 \leq x_2$,$x_3 \leq 0{,}254$, sendo x_1 a espessura da viga, x_2 o comprimento das soldas, x_3 a altura da viga e x_4 a largura da mesma.

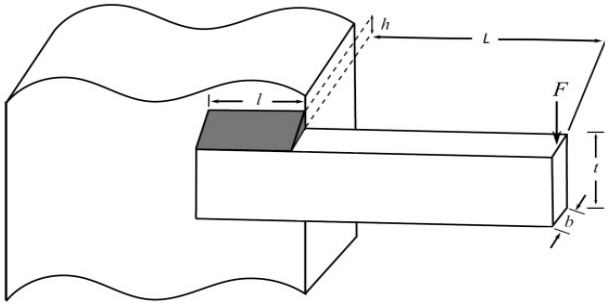


Figura 7. Geometria do problema da viga soldada [4].

A.2 Implementação

Tendo em conta não só o problema do Benchmark, mas também os problemas relativos aos autores deste relatório, optou-se por recorrer ao método de procura direta Simplex de Nelder-Mead, uma vez que nos problemas as variáveis são contínuas ou podem resolver-se como tal, recorrendo a um relaxamento, usando o *Python*.

Normalizaram-se as restrições e as funções objetivo, dividindo as mesmas por valores da mesma ordem de grandeza. Para o caso das funções objetivo, cada problema individualmente e para o caso multiobjetivo, as funções foram divididas pelos resultados obtidos na primeira iteração. De forma a penalizar as restrições, usou-se o método da penalidade exterior, elevando-as ao quadrado e multiplicando-as por um fator $p = 1000$, que se verificou suficiente para resolver o problema. O fluxograma na figura 8 ilustra o funcionamento do algoritmo. Para a gerar o *simplex* foi feita uma análise de sensibilidade à distância inicial (*step*) e ao ponto de partida do algoritmo. Neste caso, o *step* tem um valor de 0,05 e as primeiras variáveis são $h = 0{,}01$, $l = 0{,}005$, $t = 0{,}2$ e $b = 0{,}01$.

A.3 Resultados

Tal como mencionado anteriormente, primeiro resolveu-se o problema de cada função objetivo individualmente, ou seja, para $\alpha = 1$ e $\alpha = 0$.

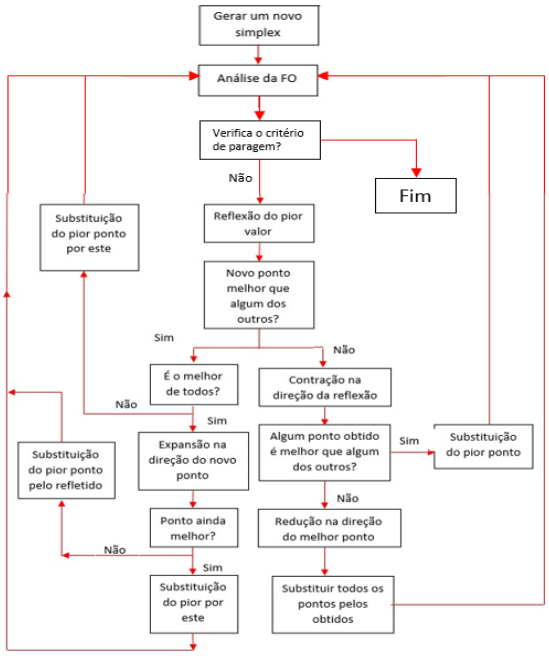


Figura 8. Fluxograma do método Simplex.

Sendo que o método se baseia num *simplex*, gerou-se o gráfico da figura 9, que ilustra a forma como a distância entre os pontos do mesmo diminui enquanto se converge num mínimo.

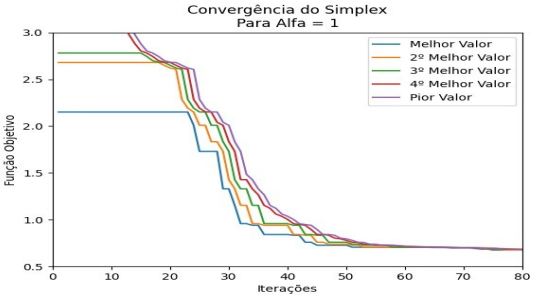


Figura 9. Convergência do *simplex*.

Posteriormente, fez-se variar o alfa em 100 valores igualmente espaçados num intervalo [0,1] e analisou-se a forma de como cada função reagia individualmente. O resultado foi o esperado, apesar de algumas irregularidades nas curvas, uma vez que a minimização da defleção foi desvalorizada progressivamente com o valor de alfa e vice-versa, como está representado na figura 10.

Após correr o algoritmo, foi possível chegar à curva de Pareto do problema (ver figura 11). Estes resultados foram de encontro ao esperado. É visível que a curva tem zonas a partir das quais estabiliza (aproximadamente). A partir de um custo de 12 euros a deflexão praticamente não se altera, assim como para deflexões superiores a 0,00002 m o custo não diminui praticamente.

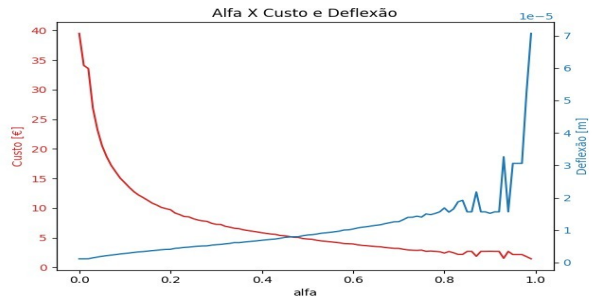


Figura 10. Variação das funções individualmente com o alfa.

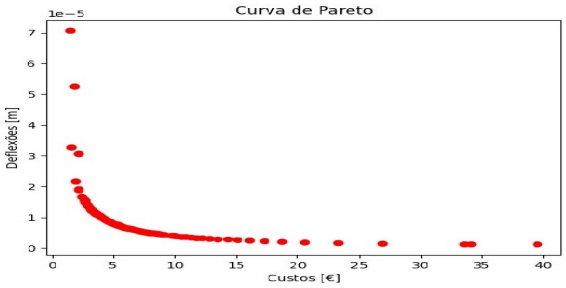


Figura 11. Curva de Pareto do sistema.

Na figura 12 está representada a convergência da função multiobjetivo normalizada para alguns valores de alfa. É possível verificar que nos casos extremos a função objetivo converge para valores perto de 1, como esperado e, nos restantes casos, o algoritmo leva muitas iterações para encontrar o valor ótimo, devido à tolerância de convergência considerada, que para estes casos poderia ter sido mais reduzida de forma a poupar recursos computacionais. No entanto, a curva de Pareto demorou aproximadamente 5 segundos a ser estabelecida, o que mostra a eficiência do algoritmo.

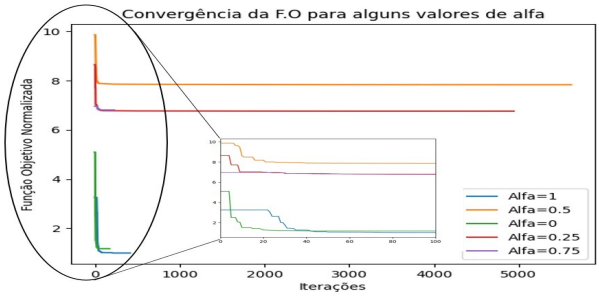


Figura 12. Convergência da função multiobjetivo normalizada.

Na tabela da figura 13 estão representadas algumas das soluções encontradas na curva de Pareto. Após a análise dos resultados, vê-se que para um alfa de aproximadamente 0,75 a deflexão é aceitável para um custo muito favorável, pelo que é um ponto bastante interessante do ponto de vista económico.

α	h [mm]	l [mm]	t [mm]	b [mm]	Custo [€]	Deflexão [mm]
0,00	21,6	45,9	254,0	127,0	39,46	0,001
0,25	5,2	7,1	254,0	30,0	8,20	0,005
0,50	4,9	7,5	254,0	17,6	4,78	0,009
0,75	10,7	5,0	254,0	10,7	2,90	0,014
1,00	9,8	13,0	54,6	9,8	0,66	1,542

Figura 13. Tabela com algumas soluções finais.

Referências

1. *MyE46*, https://mye46.com/gear-ratios/ (consultado em 10/5/2022).
2. *Automeris*, *WebPlotDigitizer*, https://automeris.io/WebPlotDigitizer/ (consultado em 17/6/2022).
3. fchollet, https://github.com/fchollet/nelder-mead (consultado em 30/5/2022).
4. Rodrigues M., Andrade-Campos A., Dias-de-Oliveira J.,Benchmark 2022: Problema de estrutura de viga soldada,Universidade de Aveiro, 2022
5. Kohli, M., Arora, S. (2018). Chaotic grey wolf optimization algorithm for constrained optimization problems. Journal of computational design and engineering, 5(4), 458-472

Otimização Não-Linear em Engenharia | MEM.2122
(artigo/relatório de projecto)

Resolução de problemas de otimização com recurso a um algoritmo de otimização GRASP híbrido

Minimização do tempo de evacuação de uma sala de espetáculos e maximização da dissipação de calor numa CPU

João Lopes · José Cação

Submetido: 24/06/2022

Resumo No presente relatório descreve-se o estudo e a implementação de um algoritmo metaheurístico de GRASP, conjugado com o Simulated Annealing, na resolução de dois problemas de otimização: minimização do tempo de eavcuação de uma sala de espetáculos e maximização da dissipação de calor numa CPU com recurso a uma superfície alhetada. Com o objetivo de testar a robustez e eficácia do algoritmo é também apresentada a resolução de um terceiro problema, o Benchmark, proposto pelos docentes da UC.

Palavras-Chave GRASP · *Simulated Annealing* · Otimização · Evacuação · Dissipação · Metaheurística

1 Introdução

O primeiro problema passa por encontrar a posição das saídas de emergência de uma sala de espetáculos que minimizam o tempo de evacuação. Este problema foi aplicado para um caso de estudo específico. No entanto, pode ser adaptado quer para outras salas de espetáculos, tal como para outros tipos de infraestruturas.

O segundo problema centra-se na otimização da distribuição e das dimensões de um conjunto de alhetas retangulares para promover a maximização da dissipação de calor numa CPU. É de interesse ao nível de enegenharia, visto que uma adequada capacidade de dissipação de calor pode garantir um maior desempenho e uma maior longevidade deste tipo de componentes.

João Lopes
n.º 95294
E-mail: jmlopes@ua.pt

José Cação
n.º 93414
E-mail: josemaria@ua.pt

O terceiro problema, o *Benchmark*, é um problema multi-objetivo de minimização do custo e da flecha máxima numa viga soldada. Este problema teve como objetivo avaliar as capacidades do algoritmo selecionado ao nível da eficiência, robustez e precisão.

2 Definição de problemas de engenharia

2.1 Minimizar o tempo de evacuação num auditório

A modelação do problema começou pela contrução das matrizes **Celulas** e **Lugares**, relativas à discretização apresentada na Figura 1, e que contêm, respetivamente, as coordenadas das células e dos lugares. Utilizou-se também uma matriz **Fronteiras** com todas as células presentes na fronteira da sala. Nesta matriz atua a variável de decisão $\mathbf{x} = \{x_1,x_2,\ldots,x_b\}$, $x_i \in \{0,1\}$, $i = 1,2,...,b$, um vetor booleano (onde b é o número de células na fronteira) que escolhe onde colocar saídas, resultando na matriz **Saidas**. A partir das matrizes **Lugares** e **Saidas**, obteve-se a matriz **Distancia**, dada pela distância euclidiana entre dois pontos. Desta matriz definiu-se a saída mais perto de cada lugar, constituindo o vetor **s**, com a saída associada a cada lugar. Por fim, obteve-se a matriz **Percursos**, que contém a distância efetivamente percorrida por cada pessoa até à saída. Utilizou-se também uma matriz **PO** que engloba para cada saída o conjunto de percursos ordenados de forma crescente.

Com a saída destinada a cada lugar e a distância percorrida por cada pessoa, calculou-se o tempo de evacuação, que é a soma do tempo para um número de pessoas passar a saída, e os tempos em que ninguém está a passá-la (furos causados pelo tempo que se demora a chegar às saídas). Como há várias saídas, o tempo

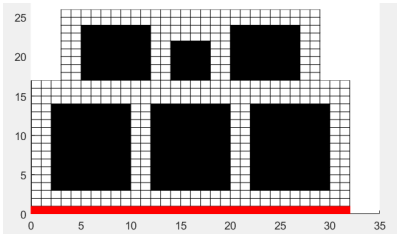


Figura 1. Discretização do caso de estudo (a preto os lugares, a vermelho o palco).

de evacuação corresponde ao máximo dos tempos de evacuação de todas as saídas. Assim os parâmetros utilizados no cálculo da função objetivo são o vetor **ns**, com o número de pessoas em cada saída, o vetor **a**, que calcula os furos derivados das diferenças entre distâncias a percorrer, e o vetor **t** com o tempo que a primeira pessoa de cada saída demora a chegar à saída.

2.2 Maximizar a dissipação de calor numa CPU

Neste problema otimizou-se a combinação número de alhetas e área de secção retangular que maximizam a dissipação de calor numa CPU. Assim, as variáveis do problema consideradas foram o número de filas de alhetas segundo x e y, $N_\mathrm{w} \in \mathbb{Z}^+$, $N_\mathrm{h} \in \mathbb{Z}^+$, e as dimensões da secção das alhetas, $w \in \mathbb{R}^+$, $h \in \mathbb{R}^+$ (ver Figura 2).

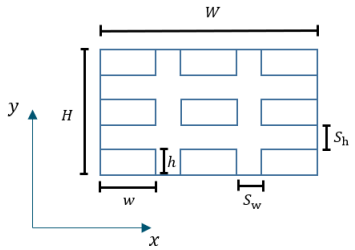


Figura 2. Representação esquemática de uma superfície alhetada, com 3 filas em ambas direções, que exemplifica o problema em causa.

Para determinar o calor dissipado pela superfície alhetada, foi preciso calcular o coeficiente de transferência de calor por convecção livre, dependente das variáveis do projeto. Segundo Harahap e McManus Jr. [1], o nº de Nusselt para superfícies alhetadas com alhetas retangulares, e sujeitas a convecção livre, é dado por

$$Nu = 0.00522 \left(\frac{Gr\ Pr\ S_\mathrm{h}\ n}{L}\right)^{0.57} \left(\frac{L}{w/2}\right)^{0.656} \left(\frac{S_\mathrm{h}}{w/2}\right)^{0.412} \tag{1}$$

$$\text{onde } Gr = g\beta \left(\frac{\rho}{\mu}\right)^2 (T_\mathrm{b} - T_\infty)\, r^3, r = \frac{2LS_\mathrm{h}}{2L + S_\mathrm{h}} \tag{2}$$

Daqui, é possível obter o coeficente de transferência de calor por convecção livre através de $h_\mathrm{conv} = \frac{Nu\, k_\mathrm{ar}}{r}$. Com o valor de h_conv, foi possível então calcular o calor dissipado pela superfície alhetada, que, de acordo com [2], corresponde à soma de dois termos: a dissipação de calor apenas pela área das alhetas, q_1, e a dissipação na base não alhetada, q_2, com

$$q_1 = N\left[M\frac{\sinh(mL) + \left(\frac{h_\mathrm{conv}}{mk}\right)\cosh(mL)}{\cosh(mL) + \left(\frac{h_\mathrm{conv}}{mk}\right)\sinh(mL)}\right], \text{ e} \tag{3}$$

$$q_2 = h_\mathrm{conv}(A_\mathrm{base} - NA_\mathrm{c})(T_\mathrm{b} - T_\infty). \tag{4}$$

3 Formulação dos problemas de otimização

3.1 Minimizar o tempo de evacuação num auditório

O problema pode ser enunciado pela procura de $\mathbf{x} = \{x_1,x_2,...,x_b\}$ que:

$$\text{minimiza } f(\mathbf{x}) = \max\left(\mathbf{ns} \times \frac{1}{vAw} + \mathbf{a} + \mathbf{t}\right)$$

$$\text{sujeito a: } g_1 : \sum_{j=1}^{b} x_j \leq m$$

$$g_2 : \mathbf{Saidas}(l,2) \neq y_\mathrm{palco}$$

$$x_j \in \{0,1\},\ j = 1,2,...,b;\ l,k = 1,2,...,m$$

$$\text{onde: } ns_k = \sum_{i=1}^{n}\left(\frac{s_{l_i}}{k} \wedge s_{l_i} = k\right) \wedge k = 1,2,...,m$$

$$a_k = \sum_{i=1}^{ns_k} \alpha_i$$

$$\alpha_i = \begin{cases} \alpha_i, \textit{se } \alpha_i \geq 0 \ \wedge\ i \neq 1 \\ 0, \textit{se } \alpha_i < 0 \ \vee\ i = 1 \end{cases}$$

$$\alpha_i = \frac{po_{k,i}}{v} - \left[\frac{i-1}{vAw} + \frac{po_{k,1}}{v} + \left(\sum_{r=1}^{i-1}\alpha_{i-r}\right)\right]$$

$$t_k = \frac{po_{k,1}}{v}, \text{(menor } p \text{ a cada saída)}$$

(A: [Pess/m^2]; v: Veloc.[m/s]; w: Larg.Saídas [m]; ns_k: [Pess/saída]; b: Número.Cel.Fronteira; m: Número de Saídas; n é o número de lugares; f é o máximo dos membros do vetor soma (caudal e furos))

A variável do problema, **x**, é booleana, e o problema é não linear. O número de pessoas em cada saída depende da distância dos lugares às saídas, parâmetro que varia de forma não linear. A restrição g_1 é de desigualdade, limitando o número de saídas. A restrição g_2 é de domínio, impondo que a zona do palco não pode ter saídas. A terceira restrição, de domínio, limita a variável.

3.2 Maximizar a dissipação de calor numa CPU

O problema pode ser formulado pela procura de N_w,N_h, w, h que:

$$\text{maximizam } q(N_\mathrm{w},N_\mathrm{h},w,h) = q_1 + q_2$$

$$
\begin{aligned}
\text{sujeito a: } & g_1: S_\text{w} = \frac{W - N_\text{w} \times w}{N_\text{w} - 1} > 0 \\
& g_2: S_\text{h} = \frac{H - N_\text{h} \times h}{N_\text{h} - 1} > 0 \\
& w \geq w_\text{min}\,, h \geq h_\text{min} \\
& w_\text{min} \times h_\text{min} \leq w \times h < 0.25 A_\text{base} \\
& w \geq h \\
& N_\text{w}, N_\text{h} \geq 2 \\
\text{onde: } & q_1 = N \left[M \frac{\sinh(mL) + \left(\frac{h_\text{conv}}{mk}\right)\cosh(mL)}{\cosh(mL) + \left(\frac{h_\text{conv}}{mk}\right)\sinh(mL)} \right] \\
& q_2 = h_\text{conv}(A_\text{base} - N A_\text{c})(T_\text{b} - T_\infty) \\
& A_\text{c} = w \times h,\, N = N_\text{w} \times N_\text{h} \\
& m = \sqrt{(h_\text{conv} P)/(k A_\text{c})} \\
& M = \sqrt{h_\text{conv} P k A_\text{c}}(T_\text{b} - T_\infty)
\end{aligned}
$$

(T_b: Temp. base [K]; T_∞: Temp. ambiente [K]; k: Cond. térmica [$\frac{\text{W}}{\text{m K}}$]; A_base: Área da base, $W \times H$ [m^2]; L: Comp. alhetas [m])

Este problema possui quatro variáveis, das quais duas são inteiras (N_w e N_h) e duas são contínuas (w e h). Trata-se de um problema não linear, onde a função objetivo está dependente das variáveis, por exemplo, em funções hiperbólicas não lineares, e que pode ser resolvido através do relaxamento das variáveis inteiras a variáveis contínuas. No que toca às restrições, são na sua maioria de desigualdade, com apenas uma de domínio, associadas a parâmetros dimensionais e estruturais do problema.

4 Análise de sensibilidade

4.1 Minimizar o tempo de evacuação num auditório

Através da ferramenta MATLAB, este problema foi modelado de forma a que escolhendo algumas posições de saídas, retornasse o valor do tempo de evacuação. Foram testadas algumas possíveis soluções, com a média destes testes a ser de 112,23 s e o desvio máximo em relação à média 17,73 s. Depois disso, testou-se colocar as saídas todas muito próximas, e o resultado da função objetivo foi de 201,50 s, ou seja, muito superior aos restantes, como seria de esperar. Os casos com menores tempos de evacuação foram aqueles que colocavam as saídas nas laterais da sala. Dividiam as saídas de forma igual para cada lado, e não colocavam no topo da sala, pois para isso os percursos efetuados pelas pessoas seriam maiores.

4.2 Maximizar a dissipação de calor numa CPU

Neste problema recorreu-se ao Excel para promover uma análise de sensibildade às variáveis de otimização.

Analisando a variação do número de filas de alhetas (para uma área das alhetas fixa), verificou-se que há comportamentos distintos (Figura 3): numa das direções, há uma maximização do calor dissipado para o maior número possível de alhetas nessa direção (N_w), priorizando a maximização da área disponível para transferência de calor; para a outra direção, o número de alhetas (N_h) é maximizado num valor mais baixo, e que está dependente das alhetas na outra direção, privilegiando o espaçamento entre alhetas, que promove um aumento do coeficiente de transferência de calor por convecção.

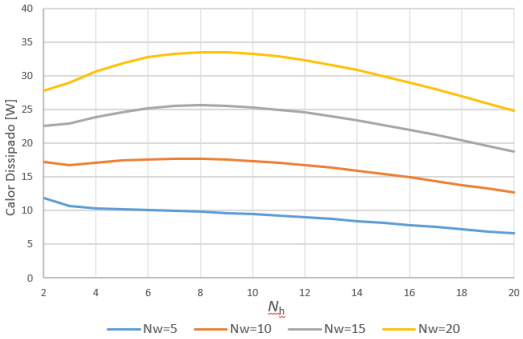


Figura 3. Análise de sensibilidade relativa ao número de filas.

Por outro lado, no que toca às dimensões das alhetas, verificou-se que a maximização do calor dissipado, para uma configuração com um espaçamento entre alhetas fixo, ocorria sempre para a área menor possível, ou seja, para os limites mínimos estabelecidos para as variáveis w e h, levando a uma maximização do número de alhetas para a troca de calor.

5 Avaliação

Ambos os problemas foram programados com recurso ao MATLAB. Em ambos os casos, a metodologia passou por atribuir valores à variável de decisão seguida de toda a modelação do problema, que incluiu a avaliação da função objetivo e das restrições. Ao nível das restrições implementaram-se funções de penalidade exterior, para penalizar a função objetivo quando as restrições não são cumpridas. Calculado o valor final da função objetivo, implementou-se a geração automática de resultados gráficos, que permitissem ver no final qual a solução estabelecida e o percurso até a obter.

6 Algoritmo de otimização

Para a escolha do algoritmo de otimização, a natureza distinta das variáveis de otimização (binária num problema, inteira/contínua no outro) levou a que se tivesse que optar por um método metaheurístico em detrimento de técnicas de procura direta ou por gradiente, adaptadas para problemas contínuos.

O algoritmo de otimização escolhido foi então um GRASP Híbrido reativo [3], conjugado com um *Simulated Annealing*, [4]. A implementação e explicação detalhadas são apresentadas em Anexo. A escolha centrou-se em três aspetos fundamentais:

- Robustez: facilmente se adapta o algoritmo a resolver problemas discretos ou contínuos;
- Precisão: engloba duas fases distintas, construção e procura local, que conjugam dois métodos de otimização distintos, e permitem uma maior capacidade de chegar ao ótimo global;
- Versatilidade: principal parâmetro do algoritmo permite criar um trade-off entre aleatoriedade e gulosidade de modo a manipular a diversidade e a intensidade da procura.

Tendo em conta a natureza distinta dos problemas propostos, foram feitas ligeiras adaptações do algoritmo para se adequar aos problemas em causa. Essas diferenças foram mais significativas na forma de gerar a solução inicial para a fase de construção, e também na geração de vizinhos da solução base, durante a fase de procura local.

7 Resultados

7.1 Minimizar o tempo de evacuação num auditório

Neste problema consideraram-se quatro saídas. Na Tabela 1 apresentam-se os resultados para a otimização da posição das quatro. É de notar que para todos os resultados apresentados o algoritmo foi executado com 120 iterações do loop principal, 50 iterações do SA, e 5 subiterações (5 vizinhos por cada iteração SA), o que dá mais de 30000 avaliações da função objetivo (considerando a Lista de Candidatos do GRASP).

Tabela 1. Resultados obtidos para seis otimizações com quatro saídas como objetivo final.

Run	Tempo de evacuação [s]	Localização das saídas (células da fronteira)
1	83.00	7, 24, 81, 104
2	82.33	9, 15, 101, 108
3	83.00	11, 23, 82, 100
4	84.67	11, 23, 84, 98
5	83.00	7, 24, 81, 102
6	82.17	10, 23, 81, 102

Na Figura 4 apresentam-se também os resultados gráficos para duas das execuções. Estes revelam que existem várias configurações possíveis que minimizam a função objetivo, pelo que não se tem apenas um mínimo global, mas um conjunto de mínimos locais. A configuração das saídas tende sempre a estabelecer alguma forma de simetria, associada à simetria da sala. Como era esperado, nenhuma solução apresentou saídas na zona do palco (pelo que a restrição do palco estava a funcionar), nem no topo da sala, pois saídas nessa zona implicariam um aumento das distâncias percorridas pelas pessoas para contornar os lugares.

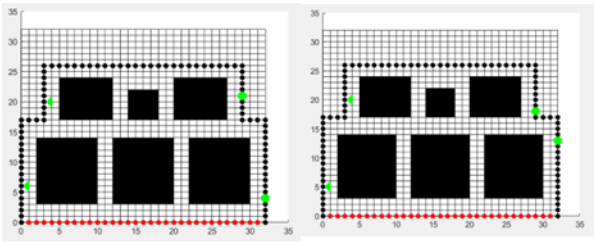


Figura 4. Resultados para a otimização da posição de 4 saídas para a sala cheia. As saídas estão representadas a verde, e a vermelho o palco.

Na Figura 5 é apresentada a evolução do melhor obtido ao longo das várias iterações, para as seis execuções referidas na Tabela 1. Como se pode verificar, tendo em conta as características estocásticas deste algoritmo, a cada execução o algoritmo atinge diferentes valores ótimos. No entanto, verifica-se que os valores para os quais converge são relativamente próximos, e que a convergência ocorre em média por volta das 15000 avaliações da função objetivo, havendo uma exceção onde a convergência ocorre às 25000.

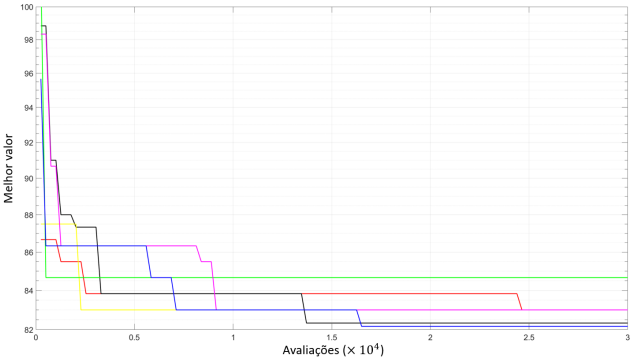


Figura 5. Melhor valor obtido ao longo das avaliações da função objetivo para seis execuções do algoritmo.

Fizeram-se ainda várias modificações nos parâmetros da sala, nomeadamente na disposição dos lugares, e do número de saídas, de modo a quebrar a simetria característica da sala e a verificar a adaptabilidade do algoritmo a estas alterações. Os resultados apresentados na Figura 6 comprovam que o algoritmo, de facto, foi

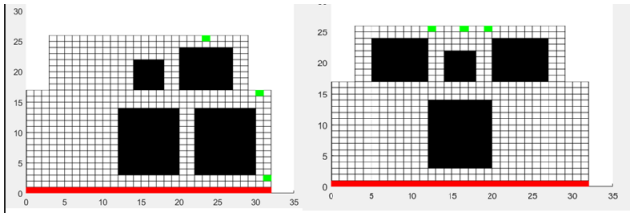


Figura 6. Resultados obtidos, considerando apenas 3 saídas, retirando todos os lugares do lado esquerdo da sala,(esquerda) e retirando dois blocos de lugares próximos do palco (direita).

capaz de ajustar os pontos ótimos aos casos impostos: retirando lugares num dos lados, as saídas localizam-se nas zonas opostas; retirando-se lugares junto ao palco, as saídas deslocam-se para o topo da sala.

7.2 Maximizar a dissipação de calor numa CPU

Para este problema, foi estabelecido inicialmente um limite mínimo de 1 mm para os valores das variáveis w e h, e tal como no primeiro problema, para aumentar a fiabilidade dos resultados, repetiu-se a otimização seis vezes. Relativamente aos parâmetros do algoritmo, para todas as execuções do algoritmo apresentadas, usaram-se 50 iterações do GRASP, e para o SA 100 iterações com 20 subiterações, o que aliado aos parâmetros de cada algoritmo se traduziu em cerca de 10^5 avaliações da função objetivo.

Tabela 2. Resultados obtidos para seis otimizações com um limite de 1 mm para as dimensões das alhetas.

Run	Calor dissipado [W]	N_w	N_h	w [mm]	h [mm]
1	52.8456	49	11	1.0018	1.0008
2	52.8516	49	11	1.0014	1.0012
3	52.8624	49	11	1.0029	1.0024
4	52.8562	49	11	1.0019	1.0017
5	52.8489	49	11	1.0014	1.0010
6	52.8537	49	11	1.0016	1.0015

Os resultados obtidos vão ao encontro da análise de sensiblidade: as alhetas tendem para as dimensões mínimas impostas, com uma das direções a possuir o número máximo de filas possível (sem que haja a quebra da restrição do espaçamento, $S_w > 0$) e a outra a estabilizar num valor muito mais baixo. Isto leva a que se tenha espaçamentos muito reduzidos numa das direções, com tendência a ter uma alheta única em toda essa direção, e um espaçamento bastante maior na outra. Do ponto de vista de dissipação de calor, verifica-se então que uma direção tende a maximizar a área de transferência de calor (com a tal alheta "única"), e a outra tende a maximizar o coeficiente de transferência de calor por convecção, com um maior espaçamento entre filas.

Relativamente ao comportamento do algoritmo, é apresentado na Figura 7 um gráfico que demonstra a convergência, em 6 execuções distintas. Como se pode ver, dado o número reduzido de variáveis, o algoritmo

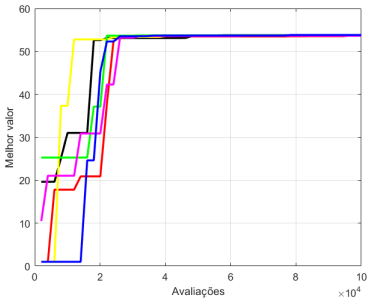


Figura 7. Convergência do algoritmo em função do número de avaliações.

apresenta uma excelente eficência, conseguindo valores próximos dos melhores obtidos por volta das 20000 avaliações, ou seja, por volta da décima iteração do GRASP. Para fazer uma validação mais profunda dos resultados, decidiu também testar-se outros limites para as variáveis w e h, com os resultados obtidos (Tabela 3) a demonstrarem a tendência dos anteriores: espaçamento mínimo numa das direções compensado pelo menor número de filas na outra.

Tabela 3. Resultados obtidos com a variação dos limites mínimos das dimensões.

Caso	Limites de w e h [mm]	Calor dissipado [W]	N_w	N_h	w [mm]	h [mm]
1	0.5	94.8614	99	24	0.5003	0.5001
2	1	52.8543	49	11	1.0016	1.0015
3	2	29.7156	24	5	2.0027	2.0002
4	5	15.6756	9	2	5.0105	5.0058

8 Conclusões

Conclui-se assim que o algoritmo escolhido mostrou-se robusto e preciso na resolução dos problemas de otimização propostos, tendo tido sucesso em ambos. Os dois problemas foram aplicados a casos de estudo específicos mas são facilmente adaptáveis a realidades diferentes. Possíveis melhorias passam por, no caso do problema da evacuação, procurar obter uma simulação mais real do processo de evacuação, talvez recorrendo a um software dedicado à matéria, e no segundo por comparar diferentes formas de alhetas, e incluir outras formas de transferência de calor na análise, como a radiação.

Referências

1. Yeh, L. and Chu, R., 2002. Thermal Management of Microeletronic Equipment. 1st ed. New York: ASME Press, p.226.
2. Bergman, T., Lavine, A., Incropera, F. and Dewitt, D., 2011. Fundamentals of Heat and Mass Transfer. 7th ed. New Jersey, pp.154-161.
3. Prata, B., 2020. Aula 8 - Meta-heurísticas (GRASP). [video] Disponível em: <https://www.youtube.com/watch?v=9R-_BL_5Gno> [Acedido em junho de 2022].
4. Celso Ribeiro, C. and Mauricio Resende, G., 2008. GRASP. ATT Labs Research Technical Report.

A Benchmark 2022

A.1 Enquadramento de benchmark

No âmbito da Unidade Curricular de Otimização Não Linear em Engenharia, foi proposta a resolução de um *benchmark* com vista a testar a robustez, a precisão e a eficência do algoritmo desenvolvido para os problemas autopropostos, tratando-se de um *benchmark* bastante conhecido em termos de otimização multiobjetivo.

O problema, cuja formulação pode ser encontrada e analisada em [1], consiste na minimização simultânea do custo e deflexão de uma viga soldada numa extremidade, estando sujeito a uma série de restrições estruturais e dimensionais. As variáveis são: $x_1 = h$ (espessura das soldas), $x_2 = l$ (comprimento das soldas), $x_3 = t$ (altura da viga), $x_4 = b$ (largura da viga). Na Figura 1 está representado um esquema do problema.

Para a resolução deste *benchmark*, e de ambos os problemas propostos, o algoritmo estudado e implementado foi um GRASP híbrido, dotado de uma fase de construção seguida por uma fase de pesquisa local com recurso a um algoritmo de *Simulated Annealing*, baseados em [2].

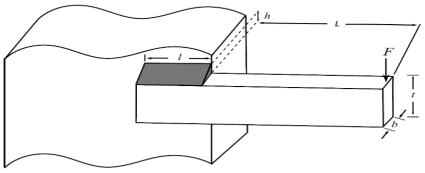


Figura 8. Esquema do problema da viga soldada, [1].

A.2 Implementação

A.2.1 Algoritmo Selecionado

O algoritmo de otimização foi programado em *Matlab*, com o *Simulated Annealing* da fase de procura local a ser baseado num código já existente [3]. Na Figura 9 apresenta-se um fluxograma relativo à implementação do mesmo.

A cada iteração é gerado, de forma reativa [4], um diferente parâmetro α_{GRASP}, o qual determina o tamanho da Lista Restrita de Candidatos (LRC) do GRASP. Após a geração desse parâmetro é iniciada fase de construção, onde a partir de uma lista de candidatos aleatória se irá aplicar um critério, baseado no α_{GRASP}, que irá restringir os candidatos à LRC, a partir da qual se escolhe um elemento aleatoriamente - solução inicial. Segue-se a fase de procura local, aplicando um SA clássico, que faz pesquisas na vizinhança da solução inicial e vai atualizando a melhor solução, sendo que no final fornece a solução ótima para cada iteração do GRASP.

O algoritmo termina quando se atinge o número máximo de iterações do GRASP, que corresponde ao critério de paragem.

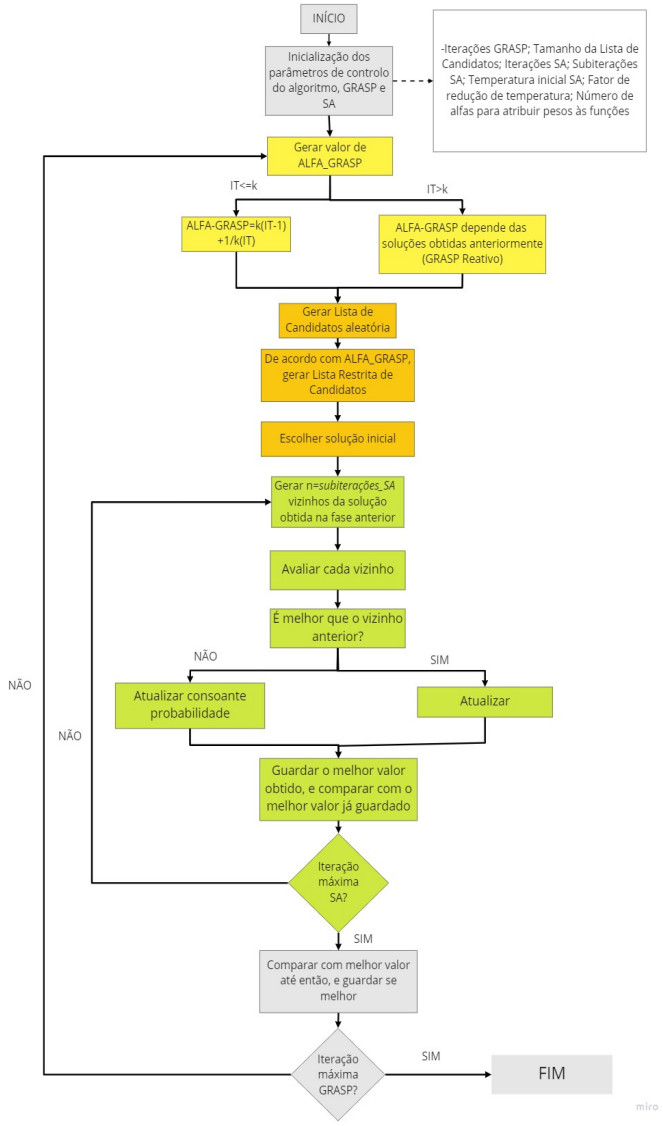


Figura 9. Fluxograma relativo ao algoritmo utilizado para resolver o benchmark.

A.2.2 Abordagem numérica ao benchmark multiobjetivo

Sendo um problema de otimização multiobjetivo, não é possível determinar apenas uma solução ótima, pois acontece que para diminuir uma função é preciso aumentar a outra. Nesse sentido, a abordagem tomada foi somar as duas funções objetivo, normalizá-las e atribuir a cada uma delas um determinado peso: α_{Peso} e $(1 - \alpha_{\text{Peso}})$. Como para cada valor de α_{Peso} o ponto ótimo da função multiobjetivo é diferente, então o objetivo passa por determinar um conjunto de pontos (f_1, f_2) que representam as melhores soluções de cada valor de peso atríbuido. A união de todos estes pontos permite construir a fronteira de Pareto.

A.3 Resultados

Para a obtenção da fronteira de Pareto, o algoritmo foi percorrido para 160 valores de $\alpha_{\text{Peso}} \in [0,1]$. Seguem-se os resultados.

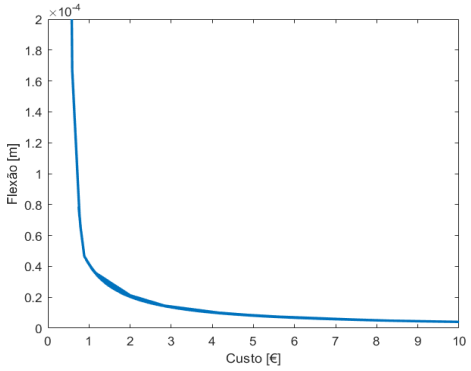


Figura 10. Parte da fronteira de Pareto obtida no benchmark.

Tabela 4. Alguns resultados obtidos na Fronteira de Pareto.

h [m]	l [m]	t [m]	b [m]	Custo [€]	Flexão [m]
0,009675	0,003895	0,254	0,009731	2,6297	1,5403e-5
0,006503	0,005840	0,254	0,006899	1,8745	2,1689e-5
0,004750	0,008104	0,254	0,005742	1,5691	2,6025e-5
0,007763	0,004748	0,254	0,010980	2,9672	1,3628e-5
0,009754	0,003918	0,254	0,010270	2,7766	1,4562e-5
0,004357	0,008695	0,254	0,004656	1,2756	3,2084e-5
0,008847	0,004412	0,254	0,010646	2,8805	1,4032e-5
0,023636*	0,016235*	0,254*	0,126998*	35,811*	1,1763e-6*
0,003202**	0,028054**	0,0955**	0,003206**	0,3641**	8,7688e-5**

* minimização da flexão

** minimização do custo

Os resultados apresentados na Figura 10 e na Tabela 4 mostram que há uma quantidade significativa de pontos em que se obtem um custo inferior a 3 € e uma flecha máxima inferior a 0.00004 m. Salienta-se também que para estes pontos as variáveis h, l , t, b não ultrapassam, respetivamente, os valores de 0.01 m, 0.009 m, 0.254 m, 0.011 m. No que toca à minimização do custo, obtem-se um valor de custo de 0,3641 €, associado a uma flexão de cerca de 8,7688e-5 m, enquanto que a minimzação da flexão leva a valores na ordem dos 1.176 μm para a flexão e 35.811 € de custo, o que está de acordo com os resultados espectáveis, disponibilizados em [1].

Para mostrar a evolução das variáveis de decisão com os valores de α_{Peso} reuniram-se, na Figura 11, as evoluções das variáveis com os vários valores de α_{Peso}. Para valores de α_{Peso} próximos de zero, x_3 e x_4 atingem os valores máximos, e à medida que o peso diminui estas também diminuem. Esta evolução faz sentido, visto que para α_{Peso} reduzidos o peso é mais atríbuido à função da deflexão, e o algoritmo tende então a aumentar estas duas variáveis de modo a diminuí-la. É possível verificar também que de acordo com uma das restrições ($x_1 < x_4$), a curva de x_1 está sempre abaixo ou igual à de x_4. No que toca à robustez do algoritmo, pode-se concluir que este é bastante robusto, tendo capacidade para resolver uma grande variedade de problemas, quer de natureza contínua quer discreta, e tal como se viu neste *benchmark*, para a resolução de problemas multiobjetivo.

Para se fazer uma avaliação do desempenho do algoritmo proposto, testou-se também um algoritmo de SA clássico para a resolução deste benchmark, com os resultados a serem apresentados na Figura 12. Relativamente à eficiência, conforme se pode ver na Figura 12, o algoritmo de GRASP implementado necessita de poucas iterações, mas muitas avaliações da

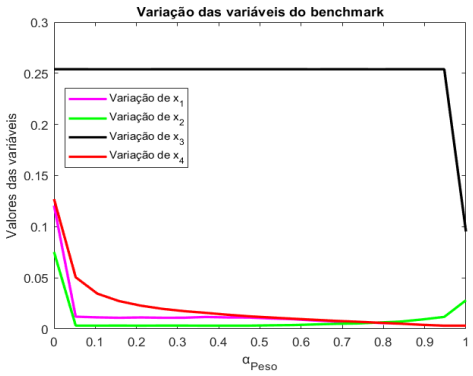


Figura 11. Evolução dos pontos ótimos das variáveis de decisão ao longo da evolução do α_{Peso}.

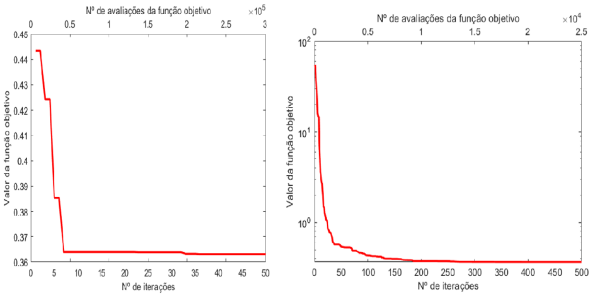


Figura 12. Evolução da função objetivo com $\alpha_{\text{Peso}} = 1$ (minimização do Custo), para o GRASP hibrido implementado à esquerda, e para o SA Clássico à direita.

função objetivo para realizar a otimização, mais que o SA clássico testado, por exemplo. Isso porque para cada iteração no GRASP, o algoritmo tem que percorrer uma fase de construção e outra de procura local, o que aumenta muito o número de avaliações da função objetivo.

Por fim, no que toca à precisão, comparando os resultados obtidos para a construção da fronteira de Pareto com os resultados analisados em [1], verificam-se bastantes semelhanças, quer da forma da curva de Pareto, quer dos valores das variáveis e funções objetivo. No entanto, não se pode deixar de salientar que sendo uma metaheurística, os resultados acabam sempre por ter uma natureza estocástica, não havendo garantias de que se chegou ao verdadeiro ótimo global.

Referências

1. Rodrigues, M., Andrade-Campos, A. and Dias-de-Oliveira, J., 2022. *Benchmark 2022: Problema da estrutura da viga soldada.*
2. Celso Ribeiro, C. and Mauricio Resende, G., 2008. GRASP. ATT Labs Research Technical Report.
3. Yarpiz. 2022. *Metaheuristics Archives - Yarpiz.* [online] Disponível em: <https://yarpiz.com/category/metaheuristics> [Acedido a 25 de maio de 2022].
4. Prata, B., 2020. Aula 8 - Metaheurísticas (GRASP). [video] Disponível em: <https://www.youtube.com/watch?v=9R-_BL_5Gno> [Acedido a junho de 2022].

Esta página foi intencionalmente deixada em branco.

Otimização Não-Linear em Engenharia | MEM.2122
(artigo/relatório de projecto)

Otimização aplicada a problemas de Last-mile

Otimização da logística de um drone de entregas e minimização do tempo de uma rota de trânsito de um veículo

Vasco Lourenço · André Figueiredo

Submetido: 24/06/2022

Resumo Neste relatório é apresentado o processo de otimização de dois problemas associados a *last-mile*. Realizou-se a sua formulação matemática e uma análise de sensibilidade a cada um dos problemas. Aplicou-se um algoritmo de *Ant-Colony* ao primeiro e o algoritmo de *Dijkstra* ao segundo. Após a análise dos resultados conclui-se um bom funcionamento em ambos os problemas. Estes resultados validam uma possível cooperação conjunta entre os dois algorítmos para resolução de um problema real de *last-mile delivery*.

Palavras-Chave Otimização · *Last-mile* · *Ant-Colony Optimization* · Algoritmo *Dijkstra*

1 Introdução

Atualmente é necessária uma constante evolução e adaptação dos processos para os manter competitivos, independentemente da área de aplicação. A otimização tem um papel fulcral neste âmbito. Esta consiste na análise e estudo de problemas em busca dos valores requeridos para obter o resultado ótimo, quer seja ele um mínimo ou máximo.

Neste relatório, no âmbito da unidade curricular de Otimização não-linear em engenharia, procurou-se focar estas técnicas em problemas de *last-mile*. O objetivo fundamental nesta área consiste em encontrar a melhor maneira de realizar a logística de distribuição da forma mais eficiente. Este problema apresenta dois problemas complementares. Primeiramente, a otimização da sequência de entregas e recolhas de uma frota e, por fim, o melhor percurso a percorrer entre cada uma das etapas, tendo em consideração os parâmetros de circulação.

O objetivo final da resolução destes problemas é combinar ambas as soluções e oferecer uma solução completa eficiente que minimize o tempo de todo o processo logístico.

Este tipo de soluções oferece uma mais-valia para qualquer empresa de distribuição visto que uma maior eficiência resultará numa maior competitividade. Embora este trabalho se foque no processo de *last-mile* pretende-se obter uma robustez e flexibilidade que permite a aplicação a outros tipos de distribuição.

V. Lourenço
n.º 93317
E-mail: v.lourenco@ua.pt

A. Figueriedo
n.º 93369
E-mail: andrefigueiredo@ua.pt

2 Definição de problemas de engenharia

2.1 Otimização da logística de um *drone* de entregas

O primeiro problema consiste em determinar o conjunto das operações realizadas por um *drone* de entregas que traduza o menor tempo de operação. O conjunto de operações $\mathbf{p}$ será composto pela sequência de pontos por onde o *drone* passará ($\mathbf{p_i}$), incluindo os regressos ao armazém para reabastecimento da sua carga.

O *drone* movimenta-se numa grelha quadrada. Nesta estão contidos os pontos de interesse, clientes e armazéns. Cada ponto de interesse é definido pelas suas coordenadas e o número de parcelas do ponto: $\mathbf{L_i} = \{x_i, y_i, N_{Ai}, \cdots,$ Os valores característicos de parcelas em cada ponto correspondem ao número de encomendas efetuadas por um cliente ou ao stock do armazém.

O tempo de operação do *drone* engloba o tempo de viagem (tv) e os tempos de manipulação de par-

celas (tm) em cada local (carga/descarga). O *drone* caracteriza-se pela sua velocidade (v_{drone}) e pela sua carga máxima, $Cmax$, a qual restringirá a movimentação do *drone*, conforme a carga de cada parcela C_j.

2.2 Minimização do tempo de uma rota de trânsito de um veículo

No segundo problema, a função objetivo consiste no tempo de uma rota entre dois pontos, **I** e **F**, o qual se tenciona minimizar. Esta minimização obter-se-á através da escolha de pontos de passagem entre o início e fim da rota, representados pelo vetor **p**.

Utilizar-se à uma representação por grafos em que cada ponto tem um conjunto de vizinhos $N(\mathbf{p_i})$. Um determinado ponto apenas tem estradas com os seus vizinhos e esta conexão é caracterizada pela velocidade limite do troço ($vmax$), a distância (d), e o fator de transito (F_t). Este conjunto de fatores afetará o tempo total que o veículo demora em cada estrada.

Foi gerado um modelo que simule o ambiente rodoviário de uma cidade. A grelha gerada foi depois transformada num grafo. Foi utilizado um fator de escala, d_{res}, que transforma as distâncias unitárias do grafo em distâncias reais. O valor escolhido para este parâmetro foi de 100 m. O modelo conta também com uma função de geração de tráfico e de tipologia de estrada.

3 Formulação dos problemas de otimização

3.1 Otimização da logística de um *drone* de entregas

A formulação do problema 1 pode ser descrita como:

Procurar **p** de modo a :

$$\begin{aligned}
&\text{minimizar} \quad f(\mathbf{p}) = \sum_{i=1}^{n} tv(\mathbf{p_i}) + tm(\mathbf{p_i}), \text{ com} \\
&tv(\mathbf{p_i}) = \frac{\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}}{v_{\text{drone}}}, \\
&tm(\mathbf{p_i}) = t_{\text{Base}} \times \sum_{j=A}^{e} N_{ji}, \\
&\text{sujeito a} \ : Cd_i \geq N_{j(i+1)} \times C_j,\ i = 1,\cdots,n \wedge j = 1,\cdots,e \\
&\qquad 0 \leq Cd_i \leq Cmax,\ i = 1,\cdots,n \\
&\qquad \mathbf{p_i} \ \subset \mathbf{L},\ \forall i \in \{1,2,\cdots,n\}
\end{aligned} \tag{1}$$

onde **p** é uma variável independente discreta de dimensão $n \times (e+2)$. A função objetivo é combinatória. Quanto aos parâmetros, estes são discretos, dos quais **L** é a matriz dos locais da grelha, armazéns e clientes, de dimensão $k \times (e+2)$. As restrições na equação 1 são refentes ao funcionamento do *drone* para cargas desfuncionais e ao domínio da variável de decisão.

3.2 Otimização da rota de trânsito de um veículo

A formulação do problema 2 é dada por:

Procurar **p** de modo a : (2)

$$\begin{aligned}
&\text{minimizar } f(\mathbf{p}) = \sum_{i=0}^{n} t(\mathbf{p_i}), \text{ com} \\
&t(\mathbf{p_i}) = \frac{d_{\text{res}} \cdot d_{i(i-1)}}{v(\mathbf{p_{(i-1)}} \rightarrow \mathbf{p_i})}, \text{ onde} \\
&v(\mathbf{p_{(i-1)}} \rightarrow \mathbf{p_i}) = vmax_{i(i-1)} \times Ft_{i(i-1)}, \\
&\text{sujeito a: } \mathbf{p_{(i-1)}} \in N(\mathbf{p_i}),\ i = 1,n \\
&d_{jj} = \infty \\
&v_{jj} = 0 \\
&Ft_{jk} \in \ ]0,1[,\ j,k = 1,a \\
&\exists!\ v \in \{50,90,120\} = vmax_{jk},\ j,k = 1,a \\
&\mathbf{p_0} = \mathbf{I} \\
&\mathbf{p_n} = \mathbf{F}
\end{aligned} \tag{3}$$

onde **p** é uma variável independente, discreta, de dimensão $n \times 2$ e corresponde ao conjunto de pares de coordenadas dos pontos de passagem do veículo. A função objetivo caracteriza-se como combinatória. Os parâmetros **d**, **vmax** e **Ft** são matrizes simétricas de dimensão a^2, sendo a o número de pontos do grafo. As diagonais das matrizes expressam a ligação entre um mesmo nó, assim o valor deste será zero e infinito para a distância e velocidade, respetivamente. Os valores de **Ft** e **d** são contínuos mas os valores de **Ft** são discretos. **I** e **F** são vetores linha de dimensão 2. Quanto às restrições, as primeiras três garantem a teoria de grafos, as duas seguintes são respetivas à velocidade do veículo e as últimas duas às condições iniciais da rota.

4 Análise de sensibilidade

4.1 Otimização da logística de um drone de entregas

Uma vez definida a formulação do trabalho, foi então realizada uma análise de sensibilidade no que toca ao

valor da carga máxima a ser transportada pelo drone. Assim, percebeu-se que, se a carga máxima fosse muito pequena, o drone acabaria por ter de regressar sempre ao armazém após cada entrega, tornando o problema imediato. No caso de *Cmax* ser muito elevado, a importância do armazém acabaria por se retirar da equação, tornando-se o problema num problema de *travelling salesman*. Tendo em conta tudo isto, foi então definido *Cmax* que tivesse isto em consideração, sendo este valor variável com o número de clientes definido.

4.2 Minimização do tempo de uma rota de trânsito de um veículo

No segundo problema foi realizada uma análise sensibilidade relativamente à tipologia das estradas. Tentou-se identificar quais os valores de dimensão adequados para um troço ser considerado autoestrada, estrada nacional e estrada urbana. Após vários testes foi possível perceber que quanto mais autoestradas existissem menor seria o valor da função objetivo e a escolha da solução daria prioridade sempre a este tipo de troços. Foi ainda realizada uma análise se sensibilidade para a representação do trânsito, onde se chegou à conclusão que estradas que não tivessem estradas alternativas teriam um trânsito superior, validando assim a função trânsito.

5 Avaliação

Toda a programação destes dois problemas foi realizada em python com recurso ao *Google Colaboratory*, facilitando o desenvolvimento do colaborativo do código. Este código encontra-se acessível na bibliografia [4, 5].

5.1 Otimização da logística de um drone de entregas

Numa fase inicial foi criada a função objetivo do problema, em seguida criou-se uma função auxiliar avaliadora da carga do drone. Esta garante que o drone teria a carga necessária para realizar a próxima entrega ao longo de todo o processo. Depois de tudo isto gerou-se o conjunto de pontos relativos aos locais dos clientes e do armazém. Por fim, definiu-se também uma função de geração de encomendas, a qual atribuía aleatoriamente um valor contido em [1,5] para o número de parcela encomendadas.

5.2 Minimização do tempo de uma rota de trânsito de um veículo

No segundo problema o primeiro passo foi gerar um modelo que simulasse o ambiente rodoviário de uma cidade. Este foi gerado com recurso a um algoritmo de *Recursive subdivision* [3]. A grelha gerada foi depois transformada num grafo, através da biblioteca Networkx presente no python. O modelo conta também com uma função de geração de tráfego. Esta gera 1000 percursos entre dois pontos aleatórios do grafo através do algoritmo de *Dijkstra*. Após uma normalização, a função retorna um valor para F_t, contido entre]0,1[.

Por fim é necessário definir o tipo de estrada para cada troço. Este será definido consoante a distância total da estrada. Primeiramente é necessário agrupar os *links* do grafo em estradas e depois avaliar a sua distância. Tendo a tipologia de cada estrada atribuída a cada troço, defini-se o limite de velocidade de 50, 90 e 120 km/h dependendo se a estrada é urbana, nacional ou autoestrada, respetivamente.

6 Algoritmos de otimização

6.1 Otimização da logística de um drone de entregas

O algoritmo escolhido para este problema foi o algoritmo *Ant-colony*. Escolheu-se este algoritmo uma vez que se adapta perfeitamente a este tipo de problemas combinatórios e é altamente eficiente [1]. Tal como o próprio nome indica, este algoritmo baseia-se no comportamento de uma colonia de formigas. Estas partilham informação entre si através de feromonas que libertam por onde passam. A quantidade de feromonas libertadas é tanto maior quanto melhor forem os recursos encontrados no trajeto.

O completo funcionamento do algoritmo encontra-se descrito na Figura 1.

6.2 Minimização do tempo de uma rota de trânsito de um veículo

Para este segundo problema a implementação do algoritmo *Ant-colony Optimization* requeria alterações significativas. Enquanto que o algoritmo *Ant-colony* explora a conectividade de todos os pontos, neste problema pretende-se avaliar apenas a rota entre dois pontos. Assim recaiu-se sobre um algoritmo de *Dijkstra*.

Este algoritmo vai avaliando o custo de cada troço de estrada, construindo caminhos e acrescentando-os a uma lista. A lista é ordenada pelo custo de cada caminho e os caminhos com menor custo são selecionados

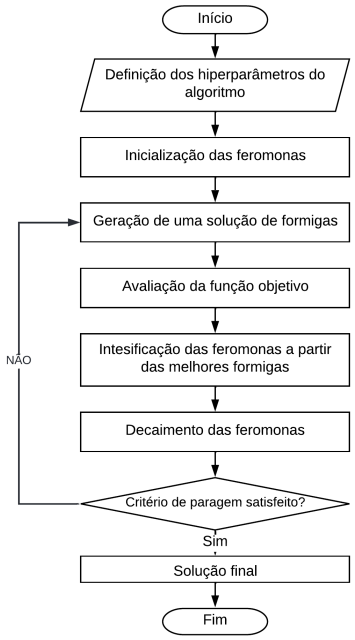


Figura 1. Fluxograma do algoritmo *Ant-colony*.

para uma nova avaliação e consequentemente expansão. Este algoritmo é altamente eficiente e converge sempre para a melhor solução [2], embora existam variações mais eficientes, como o algoritmo *a-star*.

7 Resultados

7.1 Otimização da logística de um drone de entregas

De maneira a validar o *ACO* como possível solução do problema, tentou-se resolver o problema comparando-se os resultados obtivos através de *brute-force* e de um algoritmo genético.

Através da consulta da Tabela 1, é percetível que o algoritmo *ACO*, apresenta o mesmo resultado que o algoritmo genético, o que era de esperar, uma vez que o número de locais de entrega é muito reduzido. No entanto é notória a diferença entre *runtime*. É expectável que à medida que se aumente a complexidade do problema o algoritmo *ACO* supere os valores obtidos pelo genético. Como o algoritmo de *brute-force* tem um *runtime* que aumenta rapidamente com o número de locais, limitou-se o número de iterações a apenas 10% das possíveis permutações. Esta limitação permite também obter *runtimes* na mesma ordem de grandeza. Como era de esperar, este algorítmo obteve o pior resultado dos três.

Tendo validado o correto funcionamento do algoritmo *ACO* procedeu-se a uma otimização do mesmo

Tabela 1. Resultados iniciais obtidos para 20 pontos e 2 tipos de encomendas.

	score (s)	Runtime (s)
Brute-force	1141,8	140
ACO	1046,7	40
Genético	1046,7	87

através de uma *grid search*. Fez-se variar quatro hiperparâmetros até 192 possíveis permutações, e com 40 clientes. Os resultados obtidos estão presentes no gráfico de dispersão 2, dos quais os melhores encontram-se na Tabela 2.

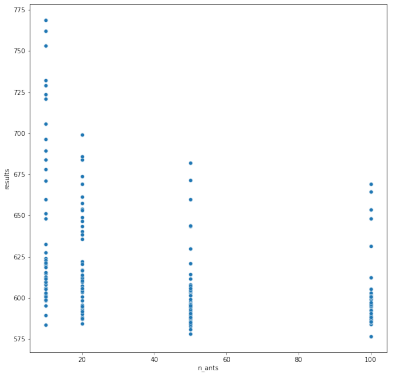


Figura 2. Dispersão dos resultados obtidos, fixando o número de formigas.

Tabela 2. Resultados das melhores combinações de hiper parâmetros obtidos no grid search.

beta	n_ants	n_best	decay	results (s)
3	100	10	0,9	576,554
3	50	5	0,99	578,3162
5	50	10	0,95	580,8865
1	50	10	0,95	582,7804
5	10	5	0,99	583,7394
5	50	5	0,8	583,9904
3	100	75	0,95	584,0367
5	20	2	0,99	584,5169

Com o auxílio da Tabela 2 foram então escolhidos 2 casos de estudo. O caso de estudo A corresponde aos parâmetros que obtiveram o melhor desempenho no *grid-search*. O caso B, embora tenha apenas uma população de 10 formigas, tem um desempenho muito próximo ao melhor resultado. Como a população é muito reduzida o custo computacional vai ser muito reduzido, pelo que foi interessante comparar o seu desempenho com as outras alternativas.

Estes casos foram também comparados com um algoritmo genético desenvolvido também no âmbito da

UC. Os resultados obtidos da comparação destes 3 casos para um numero de clientes de 20 e 40 encontra-se na Tabela 3. Os resultados para o algoritmo de *Ant-colony* são respetivos à média de 10 *runs*. Um exemplo de resultado encontra-se presente na Figura 3.

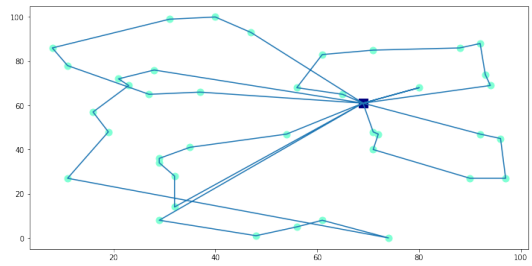


Figura 3. Solução obtida pelo ACO A para 40 clientes.

Com recurso à Tabela 3 e à Figura 4 pode-se perceber que o caso A do *ACO* é aquele que apresenta melhor resultado em ambas as situações, convergindo muito rapidamente (por volta das 20 iterações). O caso B, apesar de apresentar desempenho ligeiramente inferior é uma opção bastante eficiente, uma vez que apresenta valores de *runtime* muito inferiores às outras duas opções. Com o aumento do grau de complexidade do problema é notório que o *Ant-colony* se encontra mais bem preparado que o genético, mostrando ser uma opção bem adequada.

Tabela 3. Resultados obtidos para para os casos de 20 e 40 clientes pelas duas variantes de Ant-colony e pelo algoritmo genético.

	20 clientes (Cmax=1000)		40 clientes (Cmax=2000)	
	score (s)	runtime (s)	score (s)	runtime (s)
ACO A	762.9	18.6	1362.3	40
ACO B	791.1	1.4	1407.5	3.8
Genético	785	64	1421.7	121

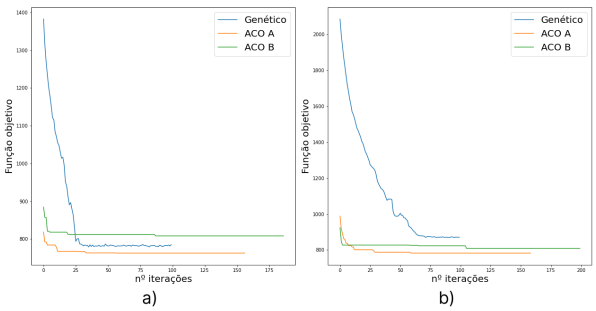


Figura 4. Evolução dos 3 algoritmos nos dois casos a e b, para 20 e 40 clientes, respetivamente.

7.2 Minimização do tempo de uma rota de trânsito de um veículo

Após o desenvolvimento de um modelo aceitavelmente próximo da realidade (enunciado na Secção 5.2) procedeu-se à implementação do algoritmo. Para a cidade gerada a rota gerada encontra-se ilustrada na Figura 5. Esta rota tem um tempo de 467 s e foi calculada 82 ms.

A rota obtida é a melhor solução do problema e é encontrada muito rapidamente pelo algoritmo. Esta comporta-se exatamente como o esperado. Procura utilizar as autoestradas para realizar a maior parte do trajeto e faz vários desvios de forma a evitar troços com um trânsito mais elevado.

A descoberta da melhor rota poderia ter sido ainda melhor se o algoritmo considerasse na sua heurística a distância relativa ao destino. Um algoritmo *a-star*.

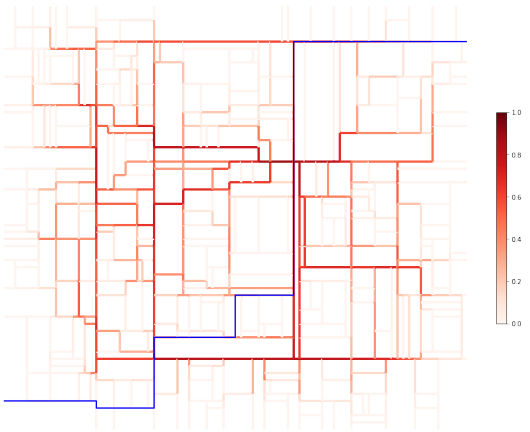


Figura 5. Rota de trânsito calculada pelo algoritmo *Dijkstra*.

8 Conclusões

Conclui-se que ambos os algoritmos se enquadraram com os problemas propostos e apresentaram uma boa eficiência. Embora o *Ant-colony* não tenha sido aplicado ao segundo problema ,demonstrou uma grande robustez, podendo, através de algumas variações, ser adaptado a problemas quer discretos, quer contínuos, como no caso do *benchmark* (Secção A). As melhorias passariam pela implementação de um vetor de feromonas que traduzisse a probabilidade de reabastecimento do drone em cada ponto. Assim esta escolha passaria a ser dada pelo algoritmo e não por uma restrição cega. Quanto ao segundo problema, as melhorias passariam pelo o desenvolvimento de um modelo mais realista, com estradas de sentidos únicos, com geometrias mais complexas e o problema tornar-se dinâmico.

A Benchmark 2022

A.1 Enquadramento de benchmark

A edição de 2022 da Unidade Curricular (UC) de Otimização Não-Linear em Engenharia propõem como *benchmark* um problema de minimização do custo e da deflexão final de uma viga soldada. Este problema foi apresentado a todos os grupos da UC. Desta forma promoveu-se a discussão e análise da utilização de vários métodos e técnicas de otimização para o mesmo problema.

Neste problema são abordadas simultaneamente a deflexão e o custo da viga, pelo que a otimização deste problema apresenta um carácter multi-objetivo. Nenhum dos objetivos terá qualquer preferencia relativamente ao outro.

O objetivo da otimização passa pela obtenção do menor custo e deflexão de uma viga soldada, deste modo, é perceptível que se trata de um problema multiobjetivo. Na resolução deste problema vai ser obtido um conjunto de pontos ótimos que são solução do problema. A junção destes pontos vai então formar a chamada fronteira de pareto. Para a obtenção destes pontos vai-se analisar a variação da solução do problema para diferentes pesos associados a cada um dos objetivos.

O problema caracteriza-se por ser problema multiobjetivo, contínuo e não-linear. As variáveis do problema são quatro e relacionam-se com o dimensionamento da viga e da ligação soldada [7]. Estas variáveis são: h, a espessura da ligação soldada, l, o comprimento da ligação soldada, t, a altura da viga e b, a largura da viga. O problema conta restrições do domínio das variáveis e restrições associadas ao comportamento mecânico da estrutura. A definição deste problema, incluindo uma formulação do problema de optimização, análise de sensibilidade e avaliação numérica, encontra-se em [7].

A utilização do mesmo algoritmo para os dois problemas e para o *benchmark* é um dos objetivos da UC. O algoritmo escolhido para a otimização dos problemas foi o algoritmo *Ant colony*, próprio para problemas de na natureza discreta e combinatória. Como o *Benchmark* é um problema com uma natureza completamente distinta, foi necessário procurar uma maneira de viabilizar a implementação deste algoritmo para um problema desta natureza.

A.2 Implementação

O algoritmo escolhido para a otimização deste problema é uma variação do *Ant colony Otimization* aplicado a domínios reais, $ACO_{\mathbb{R}}$ [8,10]. Neste algoritmo é gerado um arquivo composto por vários conjuntos de soluções que iterativamente será refinado. Em cada iteração um destes conjuntos de soluções é escolhido tendo em conta a avaliação da função objetivo e um fator aleatório. A partir deste conjunto de soluções é gerada uma distribuição normal $\mathcal{N}(\mu, \sigma^2)$ com a qual se gerará um novo conjunto de soluções. Seguidamente atualizar-se-á o arquivo caso a solução obtida seja melhor do que alguma das soluções do arquivo. O pseudo-código do algoritmo $ACO_{\mathbb{R}}$ encontra-se presente na Figura 6 [10].

O algoritmo foi implementado em *Python* [1] e testado recorrendo às funções de *Rosenbrok* e *Ackley*. .

Após testar o algorítmo $ACO_{\mathbb{R}}$ partiu-se para a otimização do problema de *Benchmark*. A estratégia de otimização partiu por utilizar uma função objetivo composta pelas funções de deflexão e custo, bem como uma componente de penalização. A componente de penalização é dada por uma função que retorna um valor elevado por cada restrição infringida, uma penalidade cega.

1: **procedure** ACO($k, n, \Delta x, f(.), m, \xi, \epsilon$) ▷ $k \rightarrow$ Archive size, $n \rightarrow$ dimension, $f(.) \rightarrow$ objective function, $m \rightarrow$ # of new solution , $\xi \rightarrow$ evaporation rate and $\epsilon \rightarrow$ stopping criteria.
2: **for** $i = 1$ to k **do**
3: **for** $j = 1$ to n **do**
4: $S_{ij} := rand(min, max)$
5: **end for**
6: $f_i = function(S_i)$ ▷ Compute function value.
7: **end for**
8: $S := Sorting(S)$; ▷ Sorting in ascending order of f.
9: **repeat**
10: **for** $l = 1$ to m **do**
11: **for** $i = 1$ to n **do**
12: Choose a solution S_{ji} according to probability of selection where $j \in [1, k]$.
13: $\mu_i = S_{ji}$ and σ_i as per (5)
14: $S'_{li} = \mathcal{N}(\mu_i, \sigma_i)$
15: **end for**
16: $f_l = function(S'_l)$
17: **end for** ▷ m new solution constructed
18: $S'' = S + S'$; ▷ $|S| = k$, $|S'| = m$ and $|S''| = k + m$, appending m new solution to k solutions.
19: $S'' = Sorting(S'')$
20: $S := S'' - S_m$; ▷ Truncate m poor solutions.
21: **until** Stopping criterion is satisfied
return f_0 ▷ Optimum function value.
22: **end procedure**

Figura 6. Pseudo-código do algoritmo $ACO_{\mathbb{R}}$.

Com o objetivo da obtenção da curva de pareto foi necessário atribuir um peso λ e $(1 - \lambda)$ às funções de custo e deflexão, respetivamente. Além disto foi também necessário normalizar ambas as funções. Para isto foi necessário calcular os valores mínimos das mesmas, recorrendo ao algoritmo $ACO_{\mathbb{R}}$. Fez-se variar λ 197 vezes para valores contidos no intervalo [0,1].

A.3 Resultados

De maneira a apresentar os resultados de uma maneira sucinta e clara, realizou-se um gráfico que representa os valores relativos à fronteira de Pareto, Figura 7. Este gráfico realizou-se tendo em conta os 197 valores de λ. Compararando os resultados obtidos com os resultados apresentados no *Benchmark* [7], conclui-se que estes foram de encontro ao esperado e são satisfatórios.

A Tabela 4, contém algumas das soluções de pareto calculadas, contendo os valores para os casos de custo mínimo e deflexão mínima, $\lambda = 0.99$ e $\lambda = 0.01$ respetivamente. Foi obtido um valor mínimo para o custo de €0.64 e 1.35e-05 m para a flexão da viga.

Através da análise da fronteira de pareto obtida percebe-se que a partir de um custo de aproximadamente €1.5 a curva de pareto tende a estabilizar, o que leva a concluir que o aumento do custo, que ainda é significativo, não irá compensar a diminuição da deflexão. No entanto, não é isto que se verifica

[1] O código desenvolvido é público e pode ser consultado em: `https://colab.research.google.com/drive/1icSQrSLXF_ckacyVAEpZ7W8EEuHHQkkS?usp=sharing`

no início da curva, onde para uma variação muito pequena do custo influencia em muito a flexão. A melhor solução acaba por se situar entre um custo de €1 e €2, uma vez que acaba por aproveitar o rápido decréscimo da flexão para custos relativamente baixos e não chega a entrar na zona de estagnação da curva.

Do conjunto de soluções presente na Tabela 4, a solução com um custo de €1.24 e uma flexão de 1.30e-04 m, seria uma ótima escolha, uma vez que oferece um compromisso estável entre as duas variáveis.

Apresenta-se ainda um gráfico da evolução da função objetivo onde é possível observar o comportamento da função objetivo ao longo das iterações, para diferentes valores de λ. Este gráfico evidencia uma convergência na função objetivo por volta da interação número 15 para um de cerca de 260.

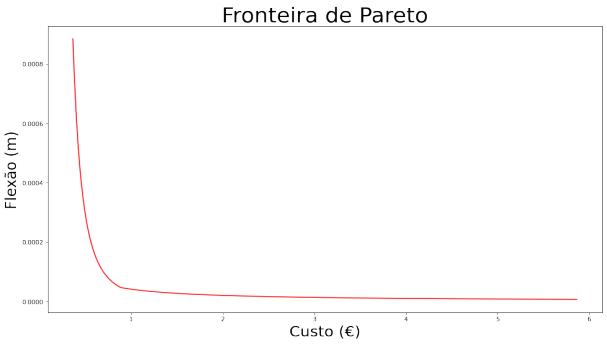


Figura 7. Fronteira de Pareto obtida

Tabela 4. Algumas das soluções de pareto obtidas.

λ	h [m]	l [m]	t [m]	b [m]	Custo [€]	Flexão [m]
0.01	0.018	0.0025	0.254	0.0307	11.57	1.35e-05
0.25	0.00533	0.00889	0.254	0.00533	2.04	7.78e-05
0.5	0.0032	0.0151	0.254	0.0032	1.24	1.30e-04
0.75	0.0032	0.0153	0.250	0.0032	1.22	1.36e-04
0.99	0.00376	0.0294	0.104	0.00375	0.64	1.58e-04

Para este problema o algoritmo $ACO_{\mathbb{R}}$ foi bastante eficiente, tendo apenas demorado 2 minutos e 17 segundos para calcular as 197 soluções. No entanto esta versão de *Ant colony Optimization* é das mais básicas, havendo versões mais eficientes que acrescentam, por exemplo, buscas locais e uma heurística de geração de soluções mais refinada. Um exemplo de algoritmo $ACO_{\mathbb{R}}$ mais avançado é o $IACO_{\mathbb{R}} - LS$ [9]. Além disto, o algoritmo ainda seria passível de uma otimização dos seus hiper-parâmetros, o que beneficiaria a eficiência do mesmo.

A robustez do algoritmo implementado é altamente satisfatória, uma vez que partindo de um algoritmo altamente especializado em problemas discretos esta variação permite resolver problemas contínuos de diferentes naturezas. Assim conclui-se que, não só o *Ant colony Optimization* é altamente eficiente em problemas discretos, mas também é eficiente em problemas de natureza contínua [8].

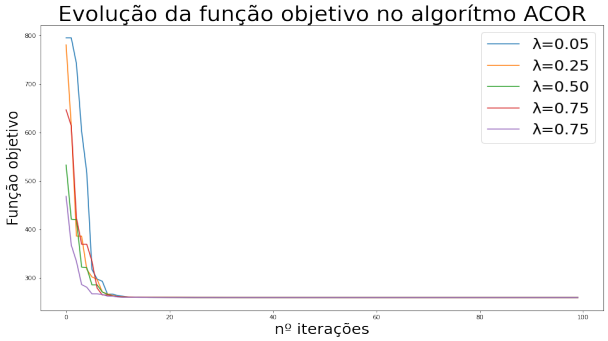


Figura 8. Evolução da função objetivo no algoritmo $ACO_{\mathbb{R}}$ para diferentes valores de λ

B Anexo

B.1 Otimização da logística de um drone de entregas

Este problema passou por diversas fases distintas de abordagem. No entanto manteve-se constante a grelha na qual o drone se movimentou. Um exemplo deste modelo está presente na Figura 9. O drone obrigatoriamente começava e acabava a sua trajetória no armazém.

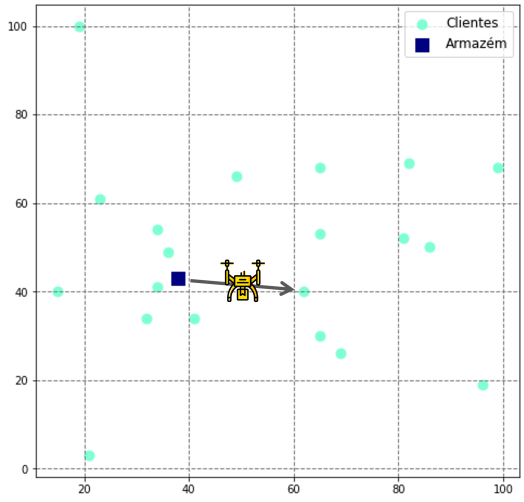


Figura 9. Exemplo de grelha com 20 pontos na qual o drone se movimenta.

Uma das fases do projeto passou pela análise do problema para vários tipos de encomendas, A e B. Na fase o algoritmo ACO ainda não estava pronto para implementação. Assim a análise foi realizada com recurso ao algoritmo genético. Neste caso as várias encomendas A e B tinham diferentes pesos, 100 e 200 gramas respetivamente. Salienta-se também que se variou os pesos e a probabilidade de cada uma das parcelas para determinar a sensibilidade das mesmas na rota do drone. Tendo em conta estas considerações apresenta-se na Figura ref um exemplo de grelha gerada e a respetiva solução gerada pelo algoritmo genético.

A partir dos resultados obtidos evidenciaram-se vários problemas. Com o aumento da complexidade do problema, os algoritmos necessitam de abordagens mais detalhadas para

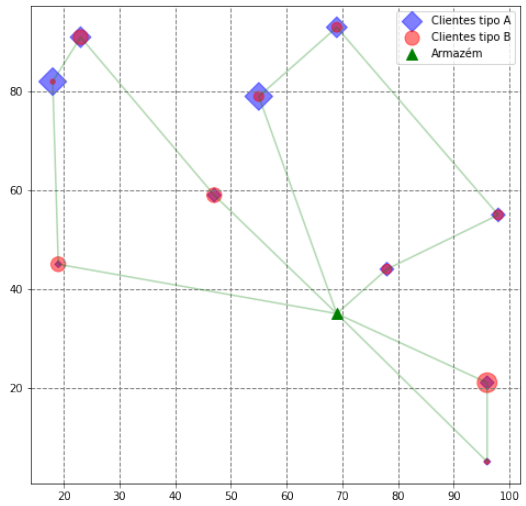


Figura 10. Solução exemplo de um caso com dois tipos de parcelas e 20 localizações.

que consigam convergir numa solução aceitável. Uma das abordagens necessárias seria o processamento independente de cada tipo de parcela, uma gestão autónoma do rácio de encomendas da carga e as restantes melhorias enunciadas anteriormente (Secção 8).

B.2 Minimização do tempo de uma rota de trânsito de um veículo

Apresenta-se de forma complementar as matrizes **p** e **d** enunciadas na formulação deste problema (Seccção 3.2). Note-se que as matrizes correspondentes à velocidade máxima e fator transito de cada troço de estrada apresentam uma configuração semelhante à matriz. **d**

$$\mathbf{p} = \begin{bmatrix} x_0 & y_0 \\ x_1 & y_1 \\ \cdots & \cdots \\ x_n & y_n \end{bmatrix}$$

$$\mathbf{d} = \begin{bmatrix} \infty & d_{01} & d_{02} & d_{03} & \cdots & d_{0a} \\ \vdots & \infty & d_{12} & d_{13} & \cdots & d_{1a} \\ \vdots & & \infty & d_{23} & \cdots & d_{2a} \\ \vdots & & & \ddots & & \vdots \\ d_{a0} & & \cdots & & & \infty \end{bmatrix}$$

Na matriz **d** os valores da distância são infinto na diagonal e para os pontos que não são vizinhos. Por exemplo, para a configuração presente na Figura 11, a matriz **d** corresponde a:

$$\mathbf{d} = \begin{bmatrix} \infty & 5 & \infty & \infty & 3 \\ & \infty & 6 & 4 & \infty \\ & & \infty & 10 & 12 \\ & & & \infty & 7 \\ & & & & \infty \end{bmatrix}$$

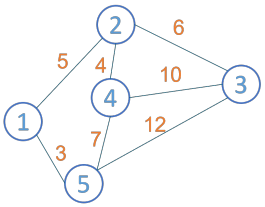


Figura 11. Exemplo de grafo, das suas conetividades e custos.

B.2.1 Geração do modelo de simulação

Um dos maiores destes desafios foi desenvolver um modelo de cidade suficientemente realista para testar o algoritmo. Este modelo passou por várias etapas com várias fases cada uma. A primeira fase consistiu no desenvolvimento da malha, a segunda a geração do trafego e por fim a geração das estradas.

A primeira etapa teve três principais fases. A primeira, ainda muito simples utilizava uma grelha discreta, com quadrados correspondentes a estrada e a obstáculos (Figura 12). Posteriormente optou-se pela abordagem de grafos e obteve-se uma malha composta por linhas (Figura 13). Finalmente, o resultado final com refinamento dos parâmetros de geração (Figura 14).

Figura 12. Exemplo de grafo, das suas conetividades e custos.

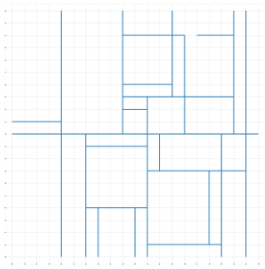


Figura 13. Exemplo de grafo, das suas conetividades e custos.

A segunda etapa foi a geração de trânsito, tal como foi descrito anteriormente (Secção 2.2). A primeira abordagem, ainda com uma modelo simples, foi utilizar uma função de *perlin noise* parametrizada para simular o trafego. Este tipo de funções é muito utilizada para a geração topográfica, logo o fator de trânsito aproximar-se ia a uma grelha com diferentes declives. Os resultados obtidos estão presentes na Figura 15. A segunda abordagem passou pela simulação de uma cidade

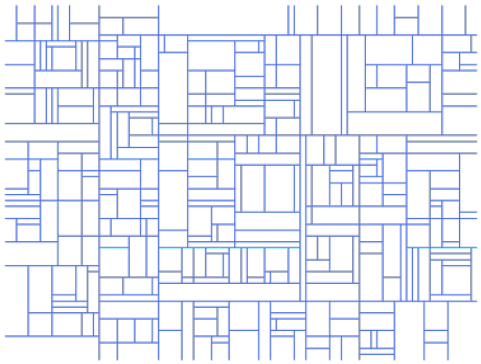


Figura 14. Exemplo de grafo, das suas conetividades e custos.

em funcionamento, gerando múltiplos trajetos de vários veículos e determinado o número de carros em cada estrada. Normalizou-se estes valores entre o valor máximo e mínimo de carros nos troços e através de uma função exponencial obteve-se um fator de trânsito entre os limites pretendidos. A função utilizada é descrita pela seguinte equação:

$$F_t = 1 - \exp\left(\frac{-n/flow}{(0.01 + traf)}\right) \tag{4}$$

Os valores de n, *traf* e *flow*, correspondem ao número de veículos simulados, ao valor de trafego normalizado e à fluidez da via. Este último valor foi fixado para 40 e corresponde à facilidade com que uma via consegue escoar veículos. Este valor deveria ser dependente do tipo de via uma vez que, por exemplo, uma autoestrada consegue escoar uma grande quantidade de carros que numa estrada local poderia levar a um engarrafamento. A malha obtida com os valores de trânsito finais encontra-se presente na Figura 16.

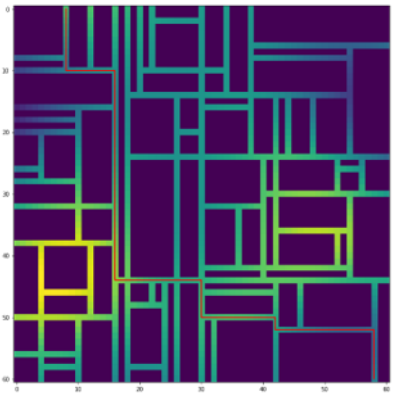


Figura 15. Fatores de trânsito obtidos com a função de *perlin noise*.

Por fim, a última etapa consistiu em agrupar os vários troços do grafo de forma a formar estradas. Dependendo do comprimento total de cada estrada designou-se a sua tipologia e consequentemente a sua velocidade máxima. Os resultados obtidos estão presentes na Figura 17. As autoestradas correspondem às linhas mais escuras, as estradas nacionais às linhas intermédias e as estradas urbanas às linhas mais claras.

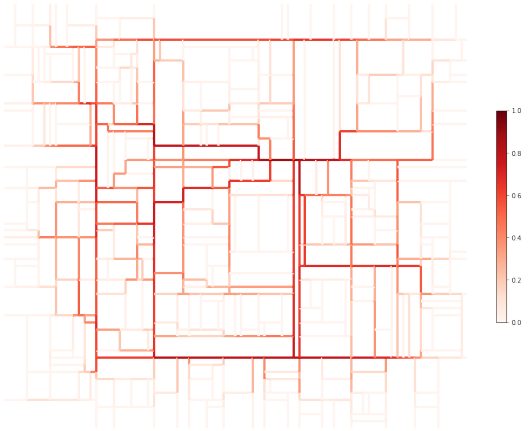


Figura 16. Fatores de trânsito finais obtidos através da simulação da circulação de veículos.

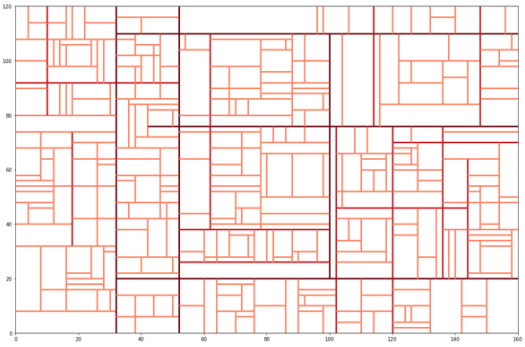


Figura 17. Grafo com a designação das diferentes tipologias de estrada.

Referências

1. Dorigo M. and Birattari M. and Stutzle T., Ant colony optimization, *IEEE Computational Intelligence Magazine*, Vol(**1.4**)::pp.28–39, 2006.
2. Javaid, A., Understanding Dijkstra's algorithm, disponível em SSRN 2340905, 2013.
3. *Vários tipos de algoritmos de Recursive Division*, `https://www.boristhebrave.com/2021/08/14/recursive-subdivision-variants/`, Consultado em 21/4/2022
4. *Código do Problema 1*, Figueiredo A. e Lourenço V., `https://colab.research.google.com/drive/1b1soMhPdANqLXtNs7WarWScaNZuNk3hX?usp=sharing`
5. *Código do Problema 2*, Figueiredo A. e Lourenço V., `https://colab.research.google.com/drive/1DSbEtVdOUUgP5ppmqDi6rBOmwi0Dvfbk?usp=sharing`
6. *Código de otimização grid-search do algoritmo Ant-colony*, Figueiredo A. e Lourenço V., `https://colab.research.google.com/drive/1JWkNa-ILnwFa2x8CoSBo_A266KPbYchA?usp=sharing`
7. Rodrigues M., Andrade-Campos A., Dias-de-Oliveira J., Benchmark 2022: Problema de estrutura de viga soldada, Otimização Não-Linear em Engenharia, Universidade de Aveiro, 2022.
8. Omran M. and Al-Sharhan S., Improved continuous Ant Colony Optimization algorithms for real-world engineering optimization problems, *Engineering Applications of Artificial Intelligence*, Vol(**85**)::pp. 818-829, 2019.

9. Tianjun Liao et al., *An Incremental $ACO_{\mathbb{R}}$ with Local Search for Continuous Optimization Problems*, IRIDIA, Bruxelles, 2011.
10. Ojha V., Abraham A. and Snášel V., *ACO for Continuous Function Optimization: A Performance Analysis*, em 2014 14th International Conference on Intelligent Systems Design and Applications, Czech Republic, 2014.

Otimização Não-Linear em Engenharia | MEM.2122
(artigo/relatório de projecto)

Otimização de horários escolares com recurso a algoritmo genético híbrido

Minimizar distâncias percorridas por estudantes entre salas de aula de blocos horários consecutivos

José Pedro Pinto

Submetido: 12/jul/2022

Resumo O presente documento descreve o estudo e a implementação de um algoritmo genético a um problema do quotidiano do autor, que se prende com a minimização das distâncias percorridas entre salas de aula. Mais ainda, foram também desenvolvidas duas versões híbridas do algoritmo original de forma a aumentar a eficiência, precisão e robustez do algoritmo. A primeira é aplicada ao caso de estudo original de otimização de horários e a segunda é aplicada num problema modelo, o *benchmark*, proposto pelos docentes da Unidade Curricular, com o intuito de validar e testar o algoritmo proposto. Ambas as implementações híbridas revelaram um desempenho superior ao algoritmo convencional, tanto a nível de eficiência, como de precisão.

Palavras-Chave Otimização Não-Linear · *Genetic Algorithm* · *Scheduling* · *Hybrid Genetic Algorithm*

1 Introdução

O agendamento de aulas é um dos problemas mais difíceis com que se deparam as universidades e escolas em todo o mundo. É um problema recorrente que ao longo dos anos tem vindo a ser discutido e analisado de forma a utilizar as instalações e os recursos existentes da maneira mais eficaz e económica possível.

Num típico problema de horários para uma instituição educacional, como uma universidade, o objetivo principal é a atribuição e distribuição de estudantes, professores e unidades curriculares a um conjunto limitado de recursos como as salas de aula e blocos de tempo. Este tipo de problema, com as suas diversas variáveis, especificidades e restrições, tem sido alvo de vários estudos de investigação nas últimas décadas, no sentido de otimizar diversas temáticas associadas ao mesmo [1,2].

J. P. Pinto
n.º 85118
E-mail: josemcp@ua.pt

2 Definição de problemas de engenharia

Com o rápido crescimento do número de estudantes e o aumento significativo do número de cursos lecionados em universidades e faculdades em todo o mundo, a tarefa de agendar aulas e conseguir inseri-las nos horários e nas instalações existentes tem vindo a tornar-se cada vez mais complicada. Na Universidade de Aveiro, em muitos casos, o processo de alocação desses recursos continua a ser efetuado manualmente, o que o torna suscetível de ser otimizado. O cariz real e atual do desafio faz acrescer a sua pertinência e urgência de resolução, e também faz com que a sua definição deva ser o mais aproximada da realidade possível. Neste sentido, esta formulação irá focar-se numa problemática que muitas vezes não é tida em consideração aquando do processo de elaboração de horários, e que pode ter um impacto significativo no quotidiano dos estudantes: a distância percorrida na deslocação entre salas de aula durante o período letivo.

O tempo de pausa disponível entre duas aulas consecutivas é muito limitado (às vezes inexistente) para que estudantes e professores possam trocar de salas, que muitas vezes estão muito distanciadas entre si. Isto resulta em atrasos sucessivos, desgaste físico e psicológico acumulado, que, consequentemente, prejudica o desempenho e aproveitamento das atividades letivas.

Servem de base ao problema os parâmetros de entrada definidos de acordo com a base de dados apresen-

tada no Anexo B. As unidades curriculares definem-se por **UC** que formam um vetor com $uc_i \in \{1, \cdots, u\}$, onde cada uc_i é caracterizada pelo seu nome, pela sua tipologia, Tuc, que define requisitos específicos para lecionar a uc_i, bem como um professor p que a pode lecionar, que forma também o vetor P com $p_i \in \{1, \cdots, m\}$, onde cada p_i é caracterizado pelo seu nome, pelas unidades curriculares que está habilitado a lecionar e pelo seu bloco horário de atendimento. Os blocos horários onde as aulas podem ser alocadas, B, com $b_i \in \{1, \cdots, r\}$ onde cada bloco define período de tempo num dia da semana. Os grupos de alunos formam o vetor G com $g_i \in \{1, \cdots, z\}$, cada g_i é caracterizado pelo número de estudantes que o constituem, NAG, e pela lista de unidades curriculares que devem ser lecionadas a esse grupo durante uma semana de aulas. Por fim, são caracterizadas as salas, pelo vetor S, com $s_i \in \{1, \cdots, l\}$, onde cada s_i é caracterizado pela sua tipologia, Tuc, pela sua lotação, LS e pela sua distância Δ ao referencial. De salientar também o parâmetro RL, que define o rácio de lotação de cada sala e é calculado através do quociente entre a lotação da sala, LS, e o número de estudantes do grupo, NAG. Assim sendo, a cada unidade curricular uc será alocado um grupo g, lecionada pelo professor p na sala s durante o bloco b, recorrendo à variável de decisão **X**. Esta variável de decisão será explanada ao longo desta secção, e será utilizada para validar as combinações através da sua contribuição binária do seu tensor na função objetivo.

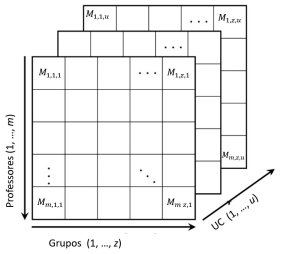


Figura 1.: Esquema representativo da matriz de combinações **M**.

A Figura 1 representa a hipermatriz **M** com dimensões $u \times z \times m$, onde se encontram todas as possíveis combinações entre os elementos dos vetores de professores, grupos e unidades curriculares. Cada elemento da matriz corresponde a uma possível combinação dos três vetores, $M_{p,g,uc}$, desta forma é criado o vetor **C** onde $c_i \in \{1, \cdots, k\}$, sendo os elementos tridimensionais da matriz **M** transportados para os elementos unidimensionais do vetor **C**, tal que $M_{1,1,1} = C_1$. Desta forma cada elemento do vetor **C** representa uma possível combinação.

A partir do vetor **C** é criada a matriz **X** com dimensões $r \times l \times k$, a variável de decisão. De salientar que a dimensão k resulta das dimensões de **M**. Esta matriz é composta também pelo vetor de salas **S** e pelo vetor de blocos horários **B**. A hipermatriz **X** resulta do desdobramento da hipermatriz **M**, formando assim um tensor de 5ªordem, $X_{b,s,p,g,uc}$, que representa a validação da alocação de um professor p, de um grupo g, de um bloco b e de uma sala s à unidade curricular uc. Por fim as distâncias percorridas pelos grupos de estudantes Δ serão calculadas através da subtração das distâncias do bloco horário $\Delta_{b+1} - \Delta_b$ ao longo de todos os blocos horários da semana letiva.

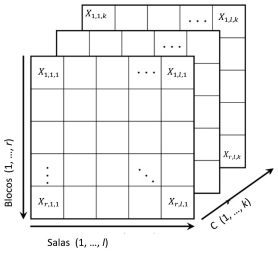


Figura 2.: Esquema representativo da matriz variável de decisão **X**.

3 Formulação dos problemas de otimização

Foi definida a variável de decisão $\mathbf{X} \in \{0,1\}$ com as dimensões $u \times m \times r \times z \times l$ como uma hipermatriz (Figura 2) que valida a distribuição de professores, p, grupos, g, salas s e blocos horários b, nas unidades curriculares, uc.

Assim sendo, o problema apresenta a seguinte formulação (Equação 1):

encontrar **X** de modo a : (1)

minimizar

$$F(\mathbf{X}) = \sum_{uc=1}^{u}\sum_{p=1}^{m}\sum_{b=1}^{r}\sum_{g=1}^{z}\sum_{s=1}^{l}\Big(\Delta_{b+1} - \Delta_b\Big) \cdot X_{b,s,p,g,uc}$$

sujeito a:

$$\sum_{p=1}^{m}\sum_{b=1}^{r} X_{b,s,p,g,uc} = 1, \quad \forall uc, \forall s, \forall g$$

$$\sum_{uc=1}^{u}\sum_{b=1}^{r} X_{b,s,p,g,uc} \leq 1, \quad \forall p, \forall s, \forall g$$

$$\sum_{uc=1}^{u} Tuc \ni Tuc(uc), \quad \forall s = 1,...,l$$

$$RL_{g,s} \geq 1, \quad \forall X$$

$$\sum_{uc=1}^{u} uc \ni P(uc), \quad \forall p = 1,...,m$$

$$\sum_{b=1}^{r} b \notin P(b), \quad \forall p = 1,...,m$$

$$\sum_{uc=1}^{u} \sum_{b=1}^{r} \sum_{p=1}^{m} X_{b,s,p,g,uc} \leq 1, \quad \forall s, \forall g$$

$$\sum_{uc=1}^{u} \sum_{b=1}^{r} \sum_{p=1}^{m} \sum_{s=1}^{l} X_{b,s,p,g,uc} \leq 1, \quad \forall g$$

A função objetivo caracteriza-se por ser combinatória e de natureza discreta.

4 Avaliação

De modo a simular da forma mais aproximada o problema de otimização de horários, foi criada uma base de dados com uma amostra significativa, que serviu de caso de estudo para o problema (*vd.* Apêndice B). A integração da base de dados foi implementada tendo em vista a maior generalização possível, de modo a que, ao longo do projeto, o caso de estudo fosse ajustado sempre que necessário. E além disso, que fossem gradualmente adicionados níveis de complexidade tornando o problema cada vez mais próximo do que seria uma situação real. O programa foi implementado em *Python* com o algoritmo descrito na secção 5, a inicialização do programa gera os primeiros horários de forma aleatória que serão iterativamente avaliados através da função descrita na Secção 3. Todos os horários são compilados, extraídos e guardados em formato de texto e extensão *.txt* no final de cada *run* do algoritmo.

5 Algoritmos de otimização

O algoritmo escolhido foi um algoritmo genético, um *population based*, implementado em linguagem *Python*, com base em código pré-existente em [3], mas com duas versões distintas. A primeira versão, um algoritmo genético convencional (AGC) com hiperparâmetros fixados *a priori*, serviu de base à segunda versão da implementação, um algoritmo genético híbrido com recomeço adaptativo e hiperparâmetros dinâmicos (AGH-RA-PD *vd. Figura 3*), inspirado em [4]. De notar que ambas as versões partilham a mesma codificação *Real-Coded* dos "Cromossomas".

É possível sub-dividir o fluxograma da Figura 3 em dois ciclos, o primeiro ciclo traduz a implementação

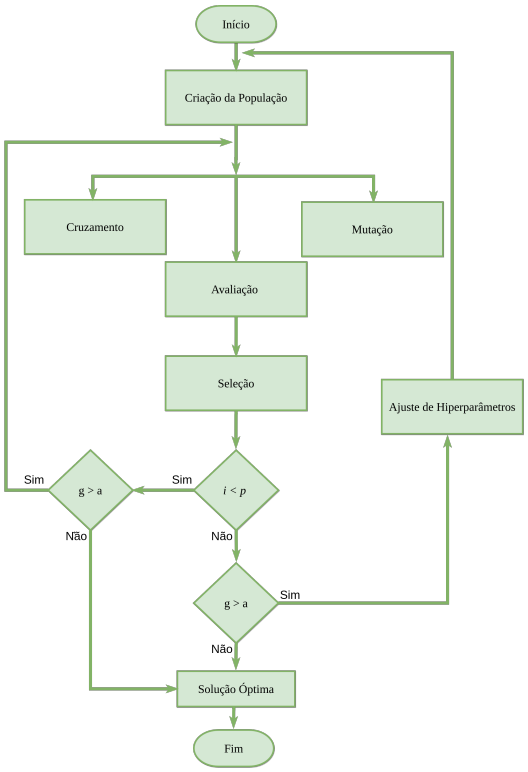


Figura 3.: Fluxograma do algoritmo AGH-RA-PD

convencional do algoritmo genético que, após a inicialização, submete a população a cruzamentos e mutações para posterior avaliação e seleção dos indivíduos reprodutores para a população seguinte. Isto acontece de forma iterativa e ininterrupta até que alguma condição de paragem seja satisfeita. O segundo ciclo traz uma nova abordagem que permite o ajuste e alteração dos hiperparâmetros do algoritmo de forma dinâmica ao longo do seu funcionamento, de forma a potenciar a sua execução ao nível da eficiência e precisão. A condição de recomeço adaptativo define-se por $i < p$. Sendo i um contador que avalia o número de avaliações da função objetivo consecutivas em que o valor ótimo se manteve inalterado, este contador é reiniciado sempre que o algoritmo encontra um novo ótimo. O fator p é uma constante de recomeço definida empiricamente pelo autor e fixada no inicio de cada execução. Sempre que o ciclo maior é ativado, o número da população será duplicado, sendo que metade será originária da última população criada (mais apta até ao momento), e a outra metade será criada através dos métodos de inicialização aleatória. De notar também que a partir de n recomeços, o índice de mutação fixado como constante inicialmente será também aumentado ciclicamente juntamente com o número população a cada recomeço. O fator n foi definido empiricamente pelo autor. O seu efeito foi propositadamente projetado para uma fase mais avançada da

convergência de modo a aumentar a exploração nessa instância, e evitando um potencial anulamento dos efeitos do cruzamento nas primeiras instâncias. Ambos os ciclos são interrompidos quando o número de avaliações g da função objetivo for superior ao número máximo de avaliações estabelecido, a.

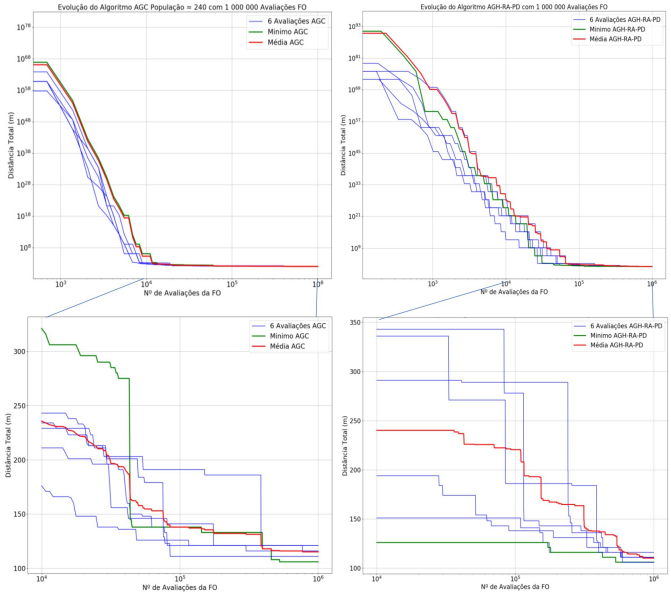
O algoritmo genético não contempla a utilização de restrições na função objetivo. Desta forma, torna-se necessária a adaptação da função objetivo para uma função objetivo aumentada, recorrendo ao método das penalidades exteriores. Este, é frequentemente utilizado em problemas similares de forma a acautelar o efeito das restrições. Apesar de uma das restrições ser continua, foi opção do autor que todas fossem avaliadas como restrições discretas de cariz binário. Não sendo mensurável o afastamento de cada restrição do domínio admissível do problema, o grau de penalização foi calculado através do número de incompatibilidades, N_{inc}, que representa o número de restrições que estão a ser infringidas em cada avaliação da função objetivo. A sua implementação na função objetivo pode ser observada na Equação 2. Importa enfatizar que, quando o número de incompatibilidades é zero, o valor da função objetivo aumentada é igual ao valor da função objetivo, e que o aumento exponencial da função objetivo é congruente com o maior número de restrições quebradas.

$$G(\mathbf{X}) = F(\mathbf{X}) + \left|1 - e^{7 \cdot N_{\text{inc}}}\right| \tag{2}$$

6 Resultados

De modo a efetuar uma análise comparativa detalhada, numa primeira instância foram definidos os moldes em que a comparação entre os dois algoritmos se ia desenrolar. Desta forma, foi fixado o critério de paragem de número máximo de avaliações da função objetivo como sendo 1 000 000 de avaliações. Para base de comparação foi executado o AGH-RA-PD com uma população inicial de 15 indivíduos e com o ajuste de hiperparâmetros, terminou a sua execução com uma população de 480 indivíduos. Assim sendo, a metodologia de testes define-se por uma análise a toda a gama de populações com a execução do AGC com o número de população a assumir os valores de 15, 30, 60, 120, 240, 480 e comparados com a execução de AGH-RA-PD. Mais ainda, cada uma das execuções foi repetida 6 vezes, devido ao cariz de aleatoriedade presente em metaheuristicas.

Os gráficos da Figura 4 apresentam a comparação da curva de convergência entre AGH-RA-PD, e a execução de AGC com o valor de população que obteve melhores resultados. Os gráficos de cada uma das outras execuções podem ser consultados no Anexo C. A nível gráfico é identificável uma grande diferença entre a Figura 4(a) e a Figura 4(b), desde logo, a curva de convergência do algoritmo híbrido com um maior escalonamento gráfico e menos secções constantes, o que demonstra claramente o efeito do recomeço adaptativo que contraria o arrastamento do mesmo ótimo durante um longo período de tempo. Com o aumento da população e do índice de mutação propicia-se gradualmente uma exploração cada vez maior e mais alargada, aumentando a probabilidade de alcance de ótimos globais. Cada ótimo local encontrado é validado sucessivamente com o mecanismo de ajuste de hiperparâmetros.



((a)) AGC População=240 ((b)) AGH-RA-PD

Figura 4.: Comparação de AGC com AGH-RA-PD com 6 execuções e 1 000 000 Avaliações da FO.

O gráfico da Figura 5 mostra a comparação entre o mínimo de cada um dos testes, bem como a média das 6 execuções. Analisando com mais detalhe o gráfico, é possível observar claramente através da curva de tendência (representada a vermelho e obtida através de uma regressão polinomial de quinta ordem) uma crescente obtenção de melhores mínimos da função, com o aumentar do número da população, o que se explica pelo aumento da diversidade da população que permite a obtenção de melhores soluções com uma maior área de exploração. Importa também salientar que, para além do aumento da população acarretar um aumento de custos computacionais de forma exponencial, o aumento a partir de certo ponto deixa de ser benéfico e começa a resultar no efeito contrário com fenómenos de convergência prematura. Isto é evidenciado pelos testes da

população de 480 que são de qualidade inferior aos da população de 15. Apenas dois dos sete testes foram capazes de atingir o mínimo de 106 m. O algoritmo híbrido alcançou esse valor com muito menos avaliações da função objetivo, e os resultados revelam que o algoritmo híbrido alcançou uma eficiência 2,1 vezes superior ao do algoritmo genético convencional (534 983 avaliações e 254 039 avaliações). Mais ainda, também a média das 6 execuções de cada teste é consideravelmente mais baixa no caso do AGH-RA-PD o que revela não só a sua precisão de alcance de bons resultados, mas também a sua consistência na obtenção dos mesmos.

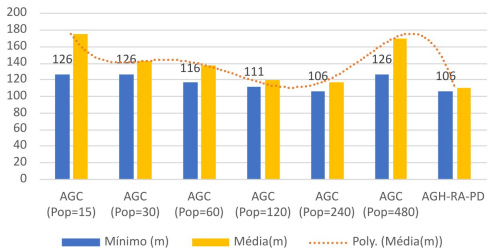


Figura 5.: Gráfico de comparação de valores mínimos e médios dos testes para 1 000 000 avaliações da FO

Para verificar e validar o valor 106 m como ótimo global do caso de estudo, foi efetuada uma nova fase de testes, aumentando o número máximo de avaliações da função objetivo para 2 500 000, e utilizando como comparação os dois melhores resultados do teste anterior, AGH-RA-PD e AGC com População de 240. Os resultados podem ser visualizados no gráfico da Figura 6. A análise gráfica mostra uma tendência de convergência mais rápida do AGC para resultados de baixa ordem de grandeza. A análise de dados mostra que, apesar desta convergência, necessita de um número superior de avaliações da função objetivo para obter um resultado da mesma precisão do algoritmo híbrido. Este, faz uma evolução contida e controlada em fases iniciais, e gradualmente vai aumentando o seu espaço de procura, reduzindo a probabilidade de convergência prematura e estagnação do algoritmo em ótimos locais. A Tabela 1 revela que a convergência para o melhor resultado do algoritmo híbrido acorreu cerca de 4,6 vezes mais rápido do que o AGC o que se traduz num aumento de eficiência de 460 %, sendo esta eficiência avaliada pelo número de avaliações até obtenção do ótimo local. Também o valor médio geral e médio do número de avaliações é consideravelmente inferior no AGH-RA-PD. Embora apenas com certeza probabilista, é possível afirmar que após uma sucessão exaustiva de testes a ambos os algoritmos, existe um elevado grau de confiança que valida o mínimo encontrado como ótimo global do caso de estudo em análise.

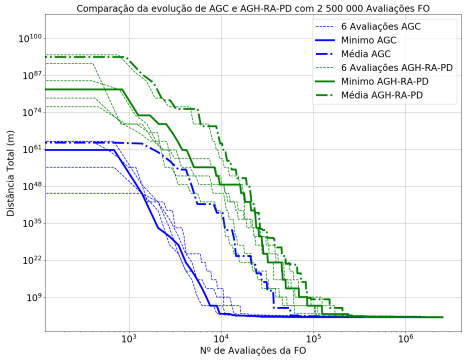


Figura 6.: Comparação AGC(Pop=240) e AGH-RA-PD para 2 500 000 Avaliações da FO.

Tabela 1.: Comparação AGC(Pop=240) e AGH-RA-PD para 2 500 000 Avaliações da FO

	AGC (Pop = 240)	AGH-RA-PD
Mínimo	106 m	106 m
Nº Avaliações	929 057	201 641
Média	111 m	110, 1 m
Nº Avaliações Médio	827 505	206 053

7 Conclusões

A profundidade da análise do algoritmo genético permitiu ao autor perceber com muito mais detalhe todo o funcionamento de um algoritmo de otimização. Isto leva necessariamente à identificação de possíveis melhorias e alterações que se traduziram nas adaptações híbridas do algoritmo base. Estas adaptações mostraram grande potencial de otimização com aumentos significativos, tanto de precisão, como de eficiência, aliando a redução do custo computacional ao aumento de probabilidade de convergência em ótimos globais, o que forma a junção perfeita para qualquer algoritmo de otimização.

Com o termino do projeto, considera-se o mesmo um sucesso. É notória a evolução de competências com unidade curricular tanto a nível de programação como de formulação e transposição de problemas reais em matemáticos, mas sobretudo o pensamento e o *mindset* de otimização que é fundamental na cabeça de qualquer engenheiro.

A Benchmark 2022

A.1 Enquadramento do benchmark

A edição de 2022 da Unidade Curricular (UC) de Otimização Não-Linear em Engenharia propõem como *benchmark* um problema real, multiobjetivo, com resultados conhecidos e, com o intuito de avaliar a robustez e a precisão do algoritmo selecionado para a resolução dos problemas autopropostos dos estudantes, propiciando assim a discussão e comparação entre as diferentes técnicas de otimização não-linear. O *benchmark* do presente ano letivo baseia-se na minimização do custo e da deflexão final de uma viga que necessita ser soldada a uma estrutura [7].

O algoritmo selecionado foi um algoritmo genético, que pertence à família das metaheurísticas, inspirado na teoria da evolução natural de Charles Darwin. Reflete o processo de seleção natural onde os indivíduos mais aptos são selecionados para reprodução de modo a produzir descendentes da próxima geração. Tipicamente este algoritmo é mais indicado e eficaz para a resolução de problemas com variáveis discretas. Sendo o *benchmark* um problema de caráter continuo, será implementada uma nova abordagem para um algoritmo genético híbrido baseada em [5, 6] que vai procurar aumentar a eficiência e precisão do algoritmo genético convencional. O seu fluxograma pode ser analisado na Figura 7 e o seu funcionamento será explanado na secção seguinte.

A.2 Implementação

O algorithmo foi implementado em linguagem *Python*, com base em código pré-existente em [3], a primeira versão, algoritmo genético convencional (AGC), foi programada com os operadores convencionais de "Cruzamento", "Mutação", "Avaliação" e "Seleção" e serviu de base para a segunda versão da implementação, algoritmo genético híbrido com procura local e recomeço adaptativo (AGH-PL-RA) (*vd.* Figura 7). Ambas as versões partilham os mesmos parâmetros intrínsecos, mas também a codificação real dos "cromossomas" com as variáveis de projeto (todas foram consideradas continuas e restritas aos limites do domínio impostos): $\mathbf{x} = (x_1, x_2, x_3, x_4) = (h, l, t, b)$. A função multi-objetivo base foi adaptada para uma função aumentada de forma a acautelar as 5 restrições impostas pelo problema $g_1(x)$, $g_2(x)$, $g_3(x)$, $g_4(x)$ e $g_5(x)$. Uma vez que o algoritmo não contempla a utilização de restrições, foram utilizadas penalidades exteriores que variam a sua intensidade consoante o afastamento do domínio admissível da restrição. Mais ainda, a função foi normalizada com o mínimo de cada umas funções uni-objetivo. A função fica completa com a inserção do peso α que servirá de fator ponderação da importância de cada parcela na solução final.

$$f(x) = \left(\alpha \cdot \frac{C(x)}{C_{\min}} + (1-\alpha) \cdot \frac{D(x)}{D_{\min}}\right) + \tag{3}$$
$$+10 \cdot \left(\frac{g_1(x)}{g_{1\max}} + \frac{g_2(x)}{g_{2\max}} + \frac{g_3(x)}{g_{3\max}} + \frac{g_4(x)}{g_{4\max}} + \frac{g_5(x)}{g_{5\max}}\right)$$

O coeficiente $C_{\min}$ e $D_{\min}$ foram obtidos com $\alpha = 1$ e com $\alpha = 0$ respetivamente. Já $g_{1\max}$, $g_{2\max}$, $g_{3\max}$, $g_{4\max}$, $g_{5\max}$ são os limites dos domínios admissíveis de cada uma das restrições. Por fim, o fator 10 é a constante de penalidade definida empiricamente pelo autor e ajustada ao caso de estudo.

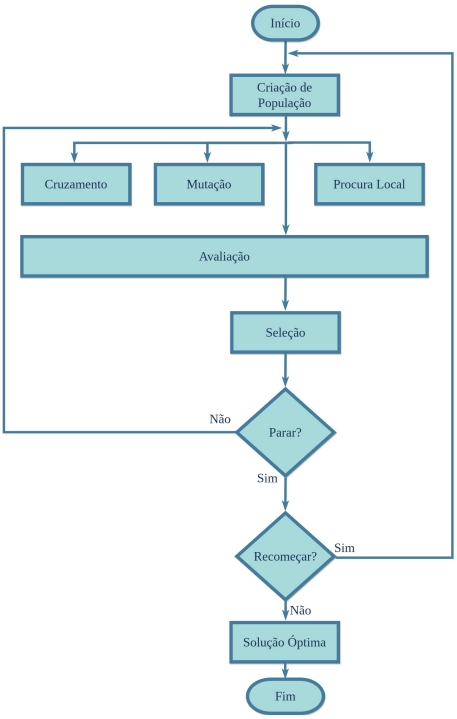


Figura 7.: Fluxograma do algoritmo AGH-PL-RA

É possível visualizar dois ciclos dentro do fluxograma da Figura 7, onde o ciclo mais pequeno é a essência do AGC, no ciclo maior a cada i iterações do ciclo mais pequeno a população será criada novamente mantendo os melhores cromossomas da iteração anterior, aumentando o *step* da procura local e de novo colocada no ciclo mais pequeno. Desta forma é possível aumentar não só a eficiência do algoritmo, mas também probabilidade de "saltar" e não convergir com ótimos locais.

Mais ainda, é incorporada dentro do ciclo mais pequeno a operação de procura local. Em cada iteração são alocados percentagens de indivíduos de forma aleatória, não só para cruzamento e para mutação, mas também para procura local onde cada individuo irá procurar nas suas proximidades melhores soluções, o resultado destas 3 operações é concatenado num grupo de indivíduos maior do que o inicial, que será avaliado e depois truncado através de seleção probabilística, mantendo assim a dimensão da população para a iteração seguinte. Os ciclos são controlados por dois critérios de paragem, o ciclo mais pequeno através da constante de recomeço adaptativo, a, e o ciclo maior através do número máximo de iterações, g.

A.3 Resultados

De modo a efetuar uma análise comparativa detalhada foi, em primeiro lugar, realizada uma primeira comparação entre os dois algoritmos com uma população de 50 indivíduos e com

uma peso α constante igual a 0,5. Devido ao seu cariz aleatório e probabilistico, os algoritmos foram avaliados 12 vezes (6 cada) e, para além das curvas de convergência, foram também retiradas as curvas de média. Os resultados podem ser observados na Figura 23. É de salientar não só a convergência muito mais rápida do AGH-PL-RA, mas também a coerência e pouca discrepância entre execuções.

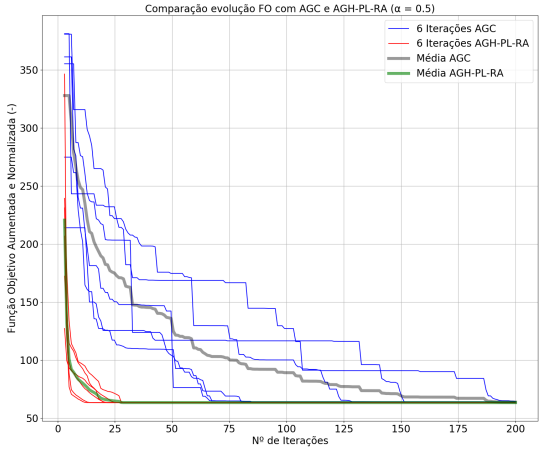


Figura 8.: Comparação Evolução FO com AGC e AGH-PL-RA

De modo a avaliar o comportamento da função com a variação dos pesos (α) foi construida a curva limite de pareto (Figura 9), para isto α foi variado de 0,1 a 1 com intervalos de 0,01, perfazendo um total de 100 avaliações. Para este teste foram efetuadas 10 avaliações para cada um dos algoritmos. Na Figura 9 encontra-se o resultado desta avaliação para o AGH-PL-RA, bem como a curva a verde com o mínimo de todas as avaliações, formando assim a curva limite de pareto.

A Tabela 2 congrega e resume as avaliações efetuadas aos algoritmos com a variação de α. Nesta pode ser facilmente identificada a superioridade do algoritmo híbrido, não só no que diz respeito à sua precisão com valores de Custo e Deflexão mais reduzidos, mas também em eficiência pelo que atingiu este mínimo global mais rapidamente, em alguns casos, com cerca de 9 a 10 vezes menos avaliações da função objetivo, o que demonstra claramente o seu potencial. Concluindo, a vantajosa maior exploração de procura global do algoritmo genético aliada à rapidez e precisão do método de procura local e ao recomeço adaptativo, formam um algoritmo bastante mais robusto, eficiente e com maior precisão.

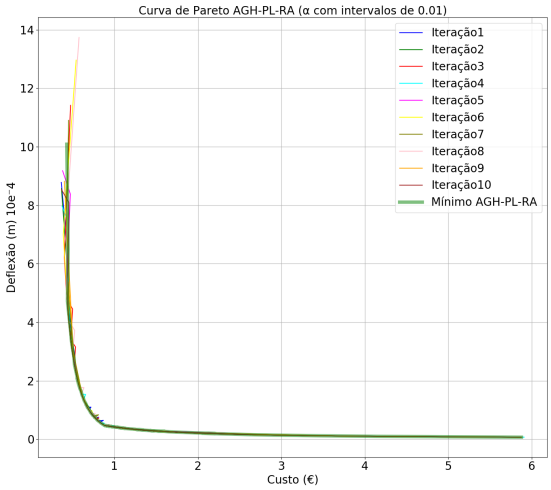


Figura 9.: Curva de Pareto AGH-PL-RA

Tabela 2.: Comparação entre AGC e AGH-PL-RA curva de pareto

	α= **0.1**		α = **0.5**		α = **0.9**	
	AGC	AGH-PL-RA	AGC	AGH-PL-RA	AGC	AGH-PL-RA
x_1=h	0.0032	0.00632505	0.0032	0.0032	0.0032	0.0032
x_2=l	0.02042814	0.00581876	0.01491646	0.01177782	0.02113709	0.01714873
x_3=t	0.254	0.254	0.254	0.254	0.15811285	0.15826985
x_4=b	0.00673786	0.00633986	0.0032	0.0032	0.0032	0.0032
Custo (€)	1.9028925	1,7824839	0.894189	0,884535	0.574044	0,565917
Deflexão(m)	2.30847e-05	2.28086e-05	4.66822e-05	4.66823e-05	19.35316e-05	19.29562e-05
Nº Avaliações	10 0000	1 500	3 750	1250	5000	1100

Referências

1. Li, T., Xie, Q., Zhang, H., Design of College Scheduling Algorithm Based on Improved Genetic Ant Colony Hybrid Optimization. *Security and Communication Networks*), 2022.
2. Nasien, D., Andi, A., Optimization of Genetic Algorithm in Courses Scheduling. *IT Journal Research and Development*),Vol(**6**(2)) 2022.
3. Wirsansky, E., *Hands-on genetic algorithms with Python: applying genetic algorithms to solve real-world deep learning and artificial intelligence problems*. Packt Publishing Ltd.2020
4. Pereira, J. C., Lobo, F. G. , Parameter-less Evolutionary Portfolio: First Experiments, *In Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pp.(**773-774**):2015.
5. Dao, S. D., Abhary, K., Marian, R., An adaptive restarting genetic algorithm for global optimization. *In Proceedings of the World Congress on Engineering and Computer Science*, Vol(**1**): Out2015.
6. GHARSALLI, L., Guérin, Y., A hybrid genetic algorithm with local search approach for composite structures optimization, *In Proceedings of the European Conference For Aeronautics And Space Sciences*, Vol(**1**)::2019.
7. M.Rodrigues, A. Andrade-Campos and J.Dias-de-Oliveira,, *Benchmark 2022: Minimização do custo e da deflexão final de uma viga soldada*, Universidade de Aveiro, Aveiro, Portugal, 2022

B Base de Dados do Caso de Estudo

```
#Sala(id, tuc, lota  o , distancia):


Salas= [['S1', 0, 40, 15],
        ['S2', 0, 40, 20],
        ['S3', 0, 30, 10],
        ['S4', 2, 35, 32],
        ['S5', 1, 35, 30],
        ['S6', 1, 50, 35],
        ['S7', 2, 60, 50],
        ['S8', 0, 25, 10],
        ['S9', 0, 50, 100]]

#Bloco(id, nome):

Blocos = [['B000', 'Seg 09:00 - 11:00'],
          ['B001', 'Seg 11:00 - 13:00'],
          ['B002', 'Seg 14:00 - 16:00'],
          ['B003', 'Seg 16:00 - 18:00'],
          ['B004', 'Ter 09:00 - 11:00'],
          ['B005', 'Ter 11:00 - 13:00'],
          ['B006', 'Ter 14:00 - 16:00'],
          ['B007', 'Ter 16:00 - 18:00'],
          ['B008', 'Qua 09:00 - 11:00'],
          ['B009', 'Qua 11:00 - 13:00'],
          ['B010', 'Qua 14:00 - 16:00'],
          ['B011', 'Qua 16:00 - 18:00'],
          ['B012', 'Qui 09:00 - 11:00'],
          ['B013', 'Qui 11:00 - 13:00'],
          ['B014', 'Qui 14:00 - 16:00'],
          ['B015', 'Qui 16:00 - 18:00'],
          ['B016', 'Sex 09:00 - 11:00'],
          ['B017', 'Sex 11:00 - 13:00'],
          ['B018', 'Sex 14:00 - 16:00'],
          ['B019', 'Sex 16:00 - 18:00']]

#Professor(nome, atendimento):

Professores = [['Antonio_Bastos', 'B002'],
               ['Rui_Moreira', 'B005'],
               ['Alfredo_Balaco ', 'B012'],
               ['Robertt_Valente', 'B007'],
               ['Ricardo_Sousa', 'B013'],
               ['Gil_Campos', 'B010'],
               ['Joao_Oliveira', 'B009'],
               ['Antonio_Ramos', 'B019'],
               ['Rosario_Correia', 'B014'],
               ['Antonio_Festas', 'B003'],
               ['Vitor_Neto', 'B006'],
               ['Miguel_Riem', 'B009'],
               ['Jose_Santos', 'B008'],
               ['Alexandre_Cruz', 'B020'],
               ['Raquel_Pinto', 'B017'],
               ['Luis_Descalco', 'B004']]

#UC(id, nome, Professores):

uc_1 = UC("uc1", "tpl",
[Professores[0]], 2)

uc_2 = UC("uc2", "dem",
[self._professores[0],
self._professores[1]], 0)

uc_3 = UC("uc3", "estruturas",
[self._professores[2]], 0)

uc_4 = UC("uc4", "desenho",
[self._professores[1],
self._professores[6]], 0)

uc_5 = UC("uc5", "ipm",
[self._professores[2],
self._professores[7]], 0)

uc_6 = UC("uc6", "solidos",
[self._professores[3],
self._professores[4]], 0)

uc_7 = UC("uc7", "iem",
[self._professores[3],
self._professores[5]], 0)

uc_8 = UC("uc8", "spt",
[self._professores[4]], 1)

uc_9 = UC("uc9", "onle",
[self._professores[5],
self._professores[6]], 0)

uc_10 = UC("uc10", "cfac",
[self._professores[7]], 1)

uc_11 = UC("uc11", "emag",
[self._professores[8]], 1)

uc_12 = UC("uc12", "sistemas",
[self._professores[2]], 0)

uc_13 = UC("uc13", "tecmec",
[self._professores[9]], 2)

uc_14 = UC("uc14", "seminarios",
[self._professores[10],
self._professores[6]], 0)

uc_15 = UC("uc15", "visao",
[self._professores[11]], 0)

uc_16 = UC("uc16", "automacao1",
[self._professores[12]], 0)

uc_17 = UC("uc17", "mecaplicada",
[self._professores[13],
self._professores[4]], 0)

uc_18 = UC("uc18", "psr",
[self._professores[11]], 0)

uc_19 = UC("uc19", "algebra",
[self._professores[14]], 0)

uc_20 = UC("uc20", "calculo",
[self._professores[15]], 0)

uc_21 = UC("uc21", "automacao2",
[self._professores[12]], 1)

uc_22 = UC("uc22", "edp",
[self._professores[6]], 0)
```

```
131
132 uc_23 = UC("uc23", "adpe",
133 [self._professores[1],
134 self._professores[5]], 0)
135
136 uc_24 = UC("uc24", "maquinastermicas",
137 [self._professores[6]], 0)
138
139 #Grupo(id, lotacao, UCs):
140
141 grupo1 = Grupo("g1", 25,
142 [unidade_curricular1,
143 unidade_curricular2,
144 unidade_curricular3,
145 unidade_curricular5,
146 unidade_curricular6,
147 unidade_curricular21,
148 unidade_curricular22,
149 unidade_curricular10])
150
151 grupo2 = Grupo("g2", 35,
152 [unidade_curricular13,
153 unidade_curricular12,
154 unidade_curricular19,
155 unidade_curricular9,
156 unidade_curricular4,
157 unidade_curricular16,
158 unidade_curricular23,
159 unidade_curricular2])
160
161 grupo3 = Grupo("g3", 30,
162 [unidade_curricular18,
163 unidade_curricular7,
164 unidade_curricular1,
165 unidade_curricular4,
166 unidade_curricular12,
167 unidade_curricular20,
168 unidade_curricular24,
169 unidade_curricular17])
170
171 grupo4 = Grupo("g4", 50,
172 [unidade_curricular8,
173 unidade_curricular10,
174 unidade_curricular11,
175 unidade_curricular15,
176 unidade_curricular17,
177 unidade_curricular5,
178 unidade_curricular14,
179 unidade_curricular19])
```

C Suporte Gráfico Comparação de Resultados

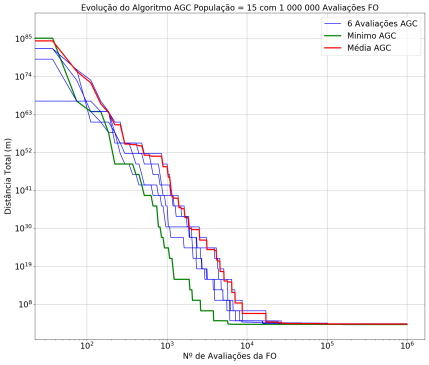


Figura 10.: Evolução AGC com População= 15 e com 1 000 000 Avaliações da FO

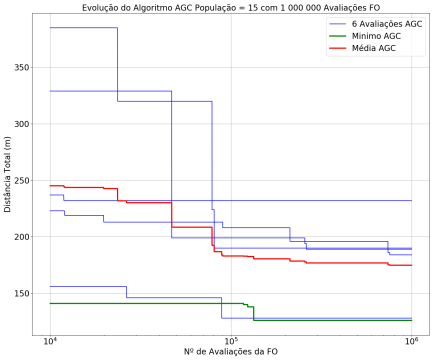


Figura 11.: Evolução AGC com População= 15 e com 1 000 000 Avaliações da FO (*Zoom*)

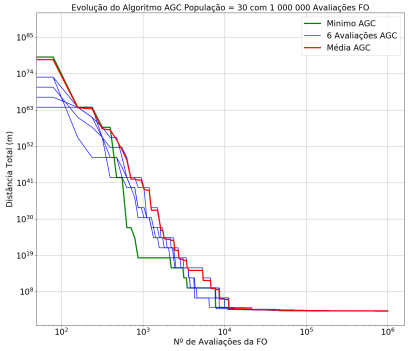


Figura 12.: Evolução AGC com População= 30 e com 1 000 000 Avaliações da FO

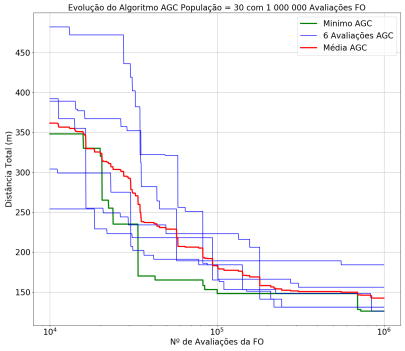


Figura 13.: Evolução AGC com População= 30 e com 1 000 000 Avaliações da FO (*Zoom*)

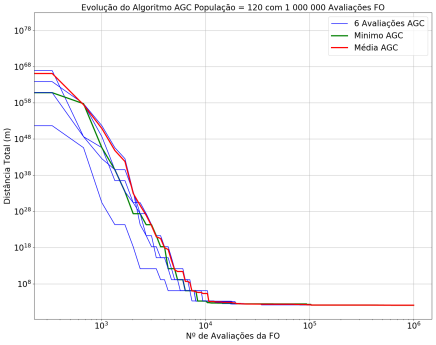


Figura 16.: Evolução AGC com População= 120 e com 1 000 000 Avaliações da FO

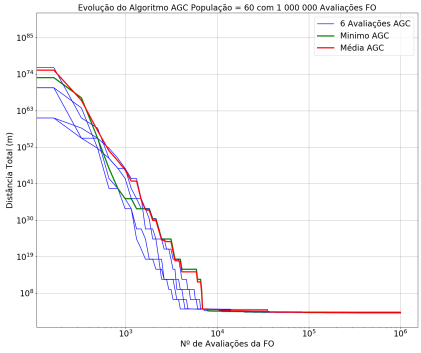


Figura 14.: Evolução AGC com População= 60 e com 1 000 000 Avaliações da FO

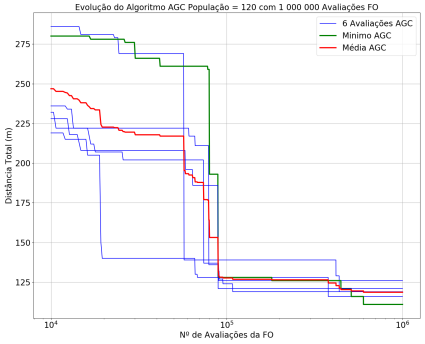


Figura 17.: Evolução AGC com População= 120 e com 1 000 000 Avaliações da FO (*Zoom*)

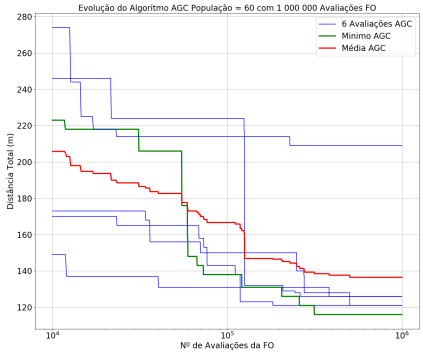


Figura 15.: Evolução AGC com População= 60 e com 1 000 000 Avaliações da FO (*Zoom*)

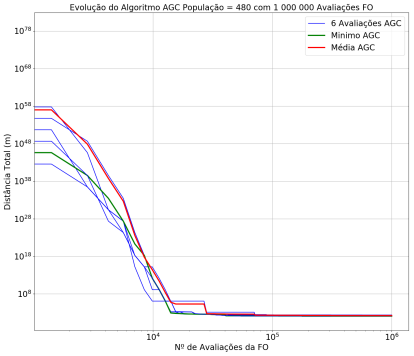


Figura 18.: Evolução AGC com População= 480 e com 1 000 000 Avaliações da FO

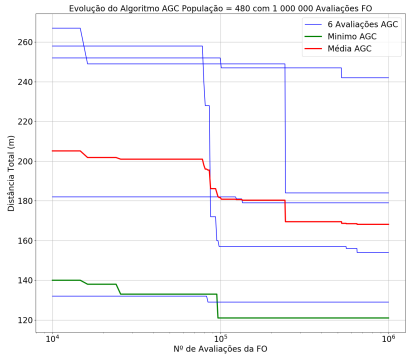


Figura 19.: Evolução AGC com População= 480 e com 1 000 000 Avaliações da FO (*Zoom*)

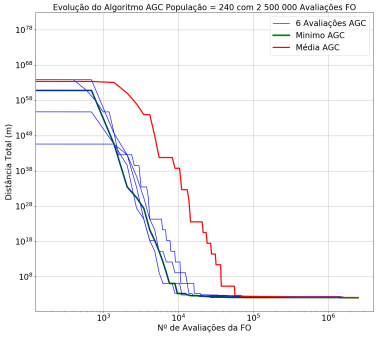


Figura 20.: Evolução AGC com População= 240 e com 2 500 000 Avaliações da FO

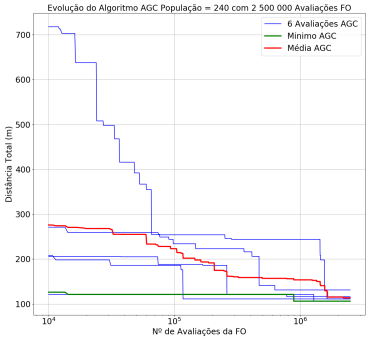


Figura 21.: Evolução AGC com População= 240 e com 2 500 000 Avaliações da FO (*Zoom*)

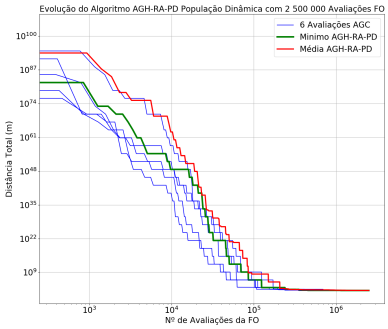


Figura 22.: Evolução AGH-RA-PD com 2 500 000 Avaliações da FO

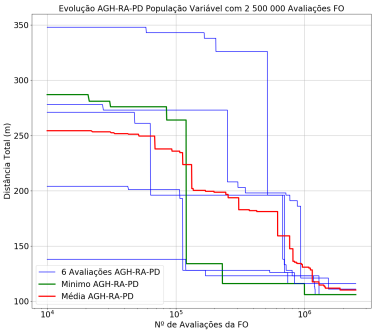


Figura 23.: Evolução AGH-RA-PD com 2 500 000 Avaliações da FO (*Zoom*)

Esta página foi intencionalmente deixada em branco.

Capítulo 3

Comentários Finais

Os trabalhos compilados neste documento são fruto do esforço das/dos estudantes da unidade curricular de ONLE, do MEM 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 unidade curricular.

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 quarta edição desta compilação de trabalhos, os docentes agradecem aos estudantes do ano letivo de 2021/2022. 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 2021/2022
Coordenadores:	A. Gil Andrade Campos, João Dias de Oliveira
Editora:	UA Editora – Universidade de Aveiro
	1ª edição – Setembro 2022
ISBN:	**978-972-789-801-5**
DOI:	https://doi.org/10.48528/aret-4p88

Os conteúdos apresentados são da exclusiva responsabilidade dos respetivos autores.
© Autores. Esta obra encontra-se sob a Licença Internacional Creative Commons Atribuição 4.0.