ISCTE-IUL
Licenciatura em Ciência de Dados

Trabalho Individual II

Exercício realizado no âmbito da Unidade Curricular de Optimização Heurística do 2º ano da Licenciatura em Ciência de Dados

André Plancha, 105289
Andre_Plancha@iscte-iul.pt

27 Maio 2023
Versão 1.0.0

Índice

a)

Uma solução S diz-se admissível se satisfaz todas as condições do problema. Para este problema, uma solução admissível é uma solução que admite cada cientista C_i um projeto P_j, cada projeto tem apenas um cientista, e todos os projetos têm um cientista, $i, j \in \{1 \,..\, 10\}$. Ou seja,

$$\forall S \in U : S \in A \Leftarrow S = \{C_1 \mapsto P_a, C_2 \mapsto P_b, ..., C_{10} \mapsto P_j\}$$

, sendo A o conjunto de soluções admissíveis, U o conjunto de soluções, e $C_\alpha \mapsto P_\beta$ o associamento do projeto α ao cientista β; $a \,..\, j, \alpha, \beta \in \{1 \,..\, 10\} \wedge a \neq b \neq ... \neq j$, $A \subset U$.

Como há apenas 10 cientistas e apenas 10 projetos, todos os cientistas vão ter projetos e todos os projetos vão ter cientistas.

b)

Uma possível heurística construtiva seria admitir como lider do projeto P_i o cientista C_i para todos os projetos, $i \in \{1 \,..\, 10\}$. Ou seja,

$$S = \{C_1 \mapsto P_1, C_2 \mapsto P_2, ..., C_{10} \mapsto P_{10}\}$$

Esta heurística, embora simples e produtora de uma solução admissível, não é interessante para **Lusa_med**, sendo que é equivalente a uma heurística que escolhe os lídderes de forma aleatória (sem reposição). Desta forma, uma heurística construtiva alternativa seria alocar para o cientista C_i o projeto P_j que tenha a melhor aptidão, entre as ainda não alocadas, $j \in \{1 \,..\, 10\}$. Ou seja, sendo $a(C_i, P_j)$ a aptidão do cientista C_i, e $\boldsymbol{P}$ os projetos para o projeto P_j,:

$$\begin{aligned}
S = \Big\{ & \\
& C_1 \mapsto P_{\text{argmax}\{a(C_i,P):P\in\boldsymbol{P}\}}, \\
& C_2 \mapsto P_{\text{argmax}\{a(C_2,P):P\in\boldsymbol{P}\setminus\{P\mapsfrom C_1\}\}}, \\
& C_3 \mapsto P_{\text{argmax}\{a(C_3,P):P\in\boldsymbol{P}\setminus\{P\mapsfrom C_1,P\mapsfrom C_2\}\}}, \\
& ..., \\
& C_{10} \mapsto P_{\text{argmax}\{a(C_{10},P):P\in\boldsymbol{P}\setminus\{P\mapsfrom C_1,P\mapsfrom C_2,...,P\mapsfrom C_9\}\}} \\
\Big\} &
\end{aligned}$$

c)

Para construir a solução, o Algorítmo 1 pode ser usado.

Recebe: Ciencistas $\boldsymbol{C}$, Projetos $\boldsymbol{P}$, função aptidão $a(C, P)$
Devolve: Solução S

1: **para todos** C_i em $\boldsymbol{C}$ **fazer:**
2: $j \leftarrow \texttt{argmax}\{a(C_i, P) : P \in \boldsymbol{P}\}$
3: adicionar $(C_i \mapsto P_j)$ a S
4: remover P_j de $\boldsymbol{P}$

Algorítmo 1: Heurística construtiva

A solução resultante do algorítmo neste problema é a seguinte:

$$
\begin{aligned}
S_0 = \{ & \\
& C_1 \mapsto P_9, \\
& C_2 \mapsto P_5, \\
& C_3 \mapsto P_2, \\
& C_4 \mapsto P_8, \\
& C_5 \mapsto P_1, \\
& C_6 \mapsto P_3, \\
& C_7 \mapsto P_7, \\
& C_8 \mapsto P_{10}, \\
& C_9 \mapsto P_6, \\
& C_{10} \mapsto P_4 \\
\}
\end{aligned}
$$

, para uma aptidão total de 832.

d)

Uma estrutura de vizinhança para este problema pode ser definida como a troca de projetos entre dois cientistas, para todos os cientistas. Ou seja,

$$
\begin{aligned}
V(S) = \{ S' \in A : S' = \left(S \setminus \{C_\alpha \mapsto P_i, C_\beta \mapsto P_j\}\right) \cup \{C_\alpha \mapsto P_j, C_\beta \mapsto P_i\} \\
\wedge \{C_\alpha \mapsto P_i, C_\beta \mapsto P_j\} \subset S \\
\}
\end{aligned}
$$

Isto pode ser implementado com o Algorítmo 2.

Recebe: Solução S
Devolve: Vizinhança da solução $\boldsymbol{V}$

1: **para todos** $C_\alpha \mapsto P_i$ em S **fazer:**
2: **para todos** $C_\beta \mapsto P_j$ em $(S \setminus C_\alpha \mapsto P_i)$ **fazer:**
3: $S' \leftarrow$ S
4: tirar $C_\alpha \mapsto P_i$ e $C_\beta \mapsto P_j$ de S'
5: adicionar $C_\alpha \mapsto P_j$ e $C_\beta \mapsto P_i$ a S'
6: adicionar S' a $\boldsymbol{V}$

Algorítmo 2: Estrutura de vizinhança

e)

Uma solução vizinha pode ser a aolução que troca os projetos de $C_1 \leftrightarrows C_2$, sendo essa:

$$
\begin{aligned}
S_1 = \{ \\
&C_1 \mapsto P_5, \\
&C_2 \mapsto P_9, \\
&C_3 \mapsto P_2, \\
&C_4 \mapsto P_8, \\
&C_5 \mapsto P_1, \\
&C_6 \mapsto P_3, \\
&C_7 \mapsto P_7, \\
&C_8 \mapsto P_{10}, \\
&C_9 \mapsto P_6, \\
&C_{10} \mapsto P_4 \\
\}
\end{aligned}
$$

, para uma aptidão total de 804.

f)

A lista tabu é uma lista de movimentos que o algoritmo tabu tem que evitar em cada iteração, de forma a o algoritmo não ficar preso num máximo local não global.Esta lista começa vazia no início do algoritmo, aumentando o seu tamanho a cada iteração, até um máximo t, a adimensão da lista tabu, ou tempo de permanência ($t \in \mathbb{N}_0$), definido *à priori* (como parâmetro do algorítmo). A cada iteração, o algoritmo adiciona na lista tabu uma representação do movimento da solução anterior para o vizinho efetuado, e retira o movimento mais velho se esceder t.

Desta forma, o algorítmo será forçado a explorar vizinhos potencialmente piores, em vês de vizinhos previamente visitados, de forma a sair do máximo local e explorar outro máximo.

Uma outra forma de interpretar o parâmetro é ele como *tempo de permanência*, ou seja, t representa o número de iterações que o movimento fica restrito de ser válido. Este conceito

é equivalente ao conceito anterior, sendo que uma dimensão t significa que o movimento M_i irá saír da lista $i + t$ iterações do movimento, sendo que este será o mais velho.

Um t demasiado alto pode fazer com que o algoritmo não visite soluções que poderiam ser melhores (nos mesmos vizinhos mas por caminhos alternativos), e um demasiado baixo pode fazer que o algoritmo não saia do máximo local. Devido a esse facto, se o algoritmo não conseguir chegar ao mínimo de aptidão desejado, serão tentados tempos diferentes. Para a primeira tentativa, $t = 4$.

g)

À lista tabu vai ser adicionado uma representação da diferença entre a solução anterior e o vizinho escolhido, independentemente do valor de aptidão da solução anterior. Aqui, vai ser escolhido representar a diferença com a troca do projeto que os cientistas fizeram, independentemente do projeto que trocaram. Ou seja, se o vizinho de f) fosse S_1 do algorítmo tabu, sendo i a iteração do algorítmo,

$$\forall t > 0 : T_1 = [C_1 \leftrightarrows C_2]^T$$

Neste caso, a lista tabu é uma matriz de dimensão $t \times 1$, sendo que cada movimento tem apenas 1 troca de projetos.

h)

Os movimentos Tabu são as tais representação da diferença entre a solução anterior e o vizinho escolhido, aqui representado como $C_i \leftrightarrows C_j$. Ou seja, na escolhida do vizinho durante o algorítmo tabu, um movimento não é válido se o vizinho tiver os projetos de C_i e C_j trocados. Por exemplo, se a lista tabu for a apresentada em g), e a solução for a apresentada em e), o movimento criado pelo vizinho S_0 e movimentos equivalentes (qualquer movimento $C_1 \leftrightarrows C_2$), seria um movimento inválido, independente dos projetos trocados, mesmo se esses tivessem uma aptidão maior, e o algorítmo teria que escolher outro vizinho na iteração em causa.

i)

O algorítmo tabu com os critérios de paragem específicados é implementado com o Algorítmo 3, $m = 850$, $M = 100$, $t = 4$.

Recebe:
Solução inicial S_0,
Mínimo de aptidão m,
Máximo de iterações M,
Tempo de permanência t,
Estrutura de vizinhança $V(S)$
Definição de movimentos $\Delta(S_a, S_b)$
Função de aptidão $a(S)$
Devolve: Melhor solução encontrada S_O

1: $T \leftarrow$ uma matriz $t \times 1$
2: $h = 0$
2: **para** i **de** 1 a M **fazer**
3: $V_i \leftarrow V(S_i)$
4: Remover de V_i todos S onde $\Delta(S, S_i)$ é uma das linhas de T
5: $S_i \leftarrow$ o S de V_i com maior aptidão
6: **se** $a(S_i) > a(S_O)$ **então**
7: $S_O \leftarrow S_i$
8: **se** $a(S_O) > m$ **então** acabar algorítmo
9: $h \leftarrow h + 1$
10: $T_{h,1} \leftarrow \Delta(S_i, S_{i-1})$
11: **se** $h > t$ **então** $h \leftarrow 0$

Algorítmo 3: Algorítmo Tabu

Cada iteração do algorítmo pode ser dividido em cinco partes:

- A construção da vizinhança [3]
- Escolha do melhor vizinho que não esteja na lista tabu [4,5]
- Critério de Aspiração [6,7]
- Critério de paragem [2,8]
- Atualização da lista tabu [9 - 11]

Ao implementar o algorítmo para o nosso S_0, o algorítmo revelou a seguinte solução:

$$
\begin{aligned}
S_O = \{ \\
& C_1 \mapsto P_9 \\
& C_2 \mapsto P_5 \\
& C_3 \mapsto P_2 \\
& C_4 \mapsto P_8 \\
& C_5 \mapsto P_1 \\
& C_6 \mapsto P_{10} \\
& C_7 \mapsto P_7 \\
& C_8 \mapsto P_3 \\
& C_9 \mapsto P_6 \\
& C_{10} \mapsto P_4 \\
\}
\end{aligned}
$$

, com uma aptidão de 865, encontrada na primeira iteração ($S_O = S_1$).

Para efeitos elustrativos, foi feito o mesmo algorítmo, mas com $m = 1001$, uma aptidão impossível de chegar, para deixar o algoritmo correr durante as 100 iterações. O algorítmo encontrou a seguinte solução:

$$
\begin{aligned}
S_O = \{ & \\
& C_1 \mapsto P_9 \\
& C_2 \mapsto P_2 \\
& C_3 \mapsto P_6 \\
& C_4 \mapsto P_8 \\
& C_5 \mapsto P_7 \\
& C_6 \mapsto P_4 \\
& C_7 \mapsto P_{10} \\
& C_8 \mapsto P_3 \\
& C_9 \mapsto P_1 \\
& C_{10} \mapsto P_5
\end{aligned}
$$

, com uma aptidão de 902.

O gráfico da Figura 1 mostra a evolução da aptidão ao longo das iterações.

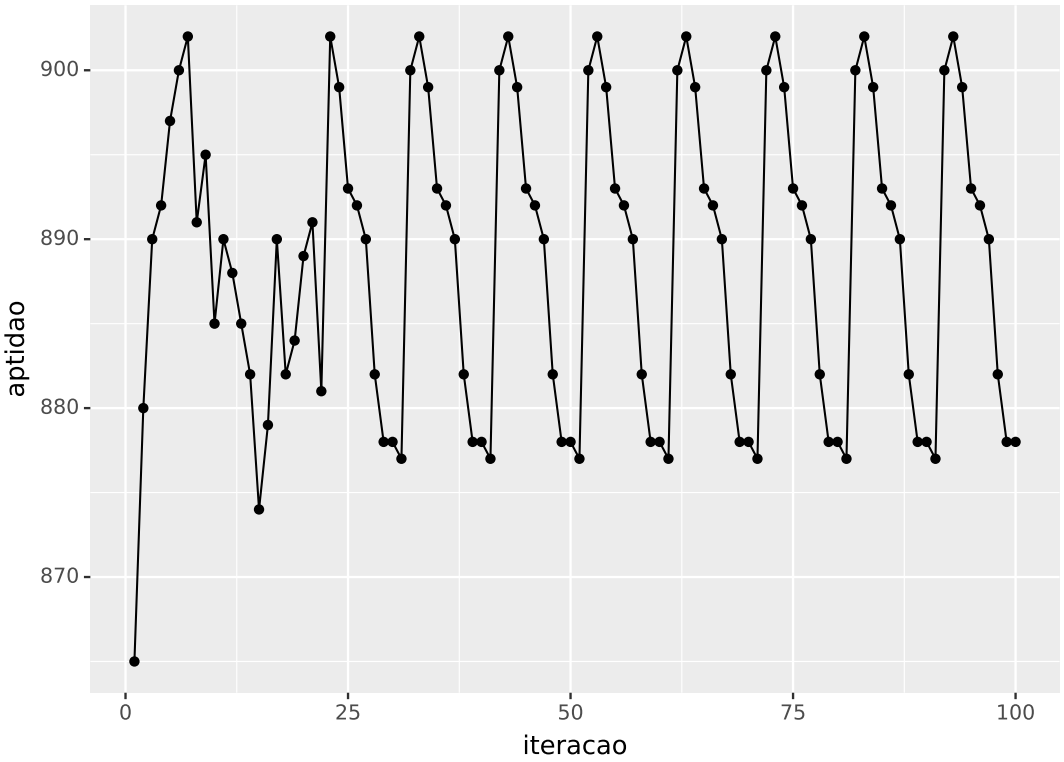

Figura 1: Evolução da aptidão, $t = 4$

Como se pode obervar, o algorítmo parece ficar num ciclo de vizinhos na iteração 22, o que significa que o algoritmo não está a explorar outras soluções que podem ser melhores, devido ao facto da lista tabu ser demasiado pequena.

O gráfico da Figura 2 mostra o mesmo algoritmo, mas com $t = 5$.

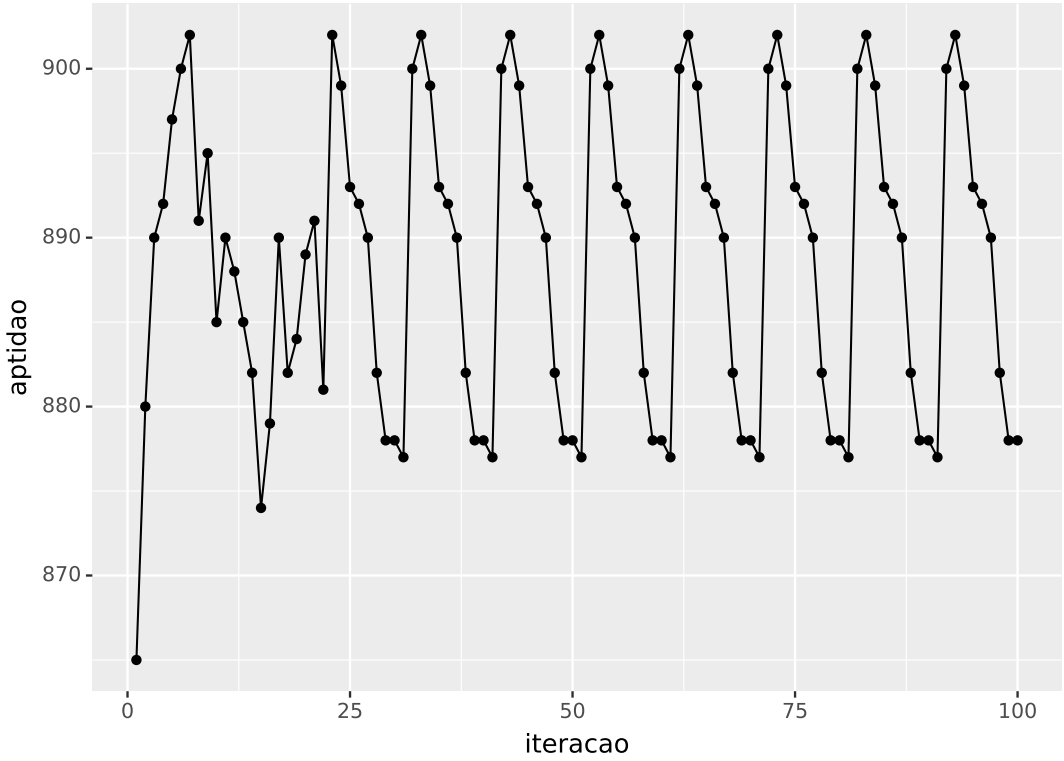

Figura 2: Evolução da aptidão, $t = 5$

Perante este resultado, o algorítmo parece procurar mais soluções, sem ficar preso num ciclo de vizinhos, embora este se comece a formar no ponto 25 para 80. No entanto, uma aptidão de 902 é completamente apropriada, e baseado nestas duas execuções, como o problema não é muito complexo, será razoável dizer qualquer solução que tenha uma aptidão de 904 é uma solução ótima.