ISCTE - Instituto Universitário de Lisboa

IBS - ISCTE Business School

Departamento de Métodos Quantitativos para Gestão e Economia

2º Ano Licenciatura em Ciência de Dados

Optimização Heurística

Trabalho Individual 2

Trabalho realizado por : João Francisco Botas ; nº104782 ; CDB1

Docentes da U.C. :

Anabela Ribeiro Dias da Costa

Maria João Sacadura Fonseca Calado de Carvalho e Cortinhal

Mafalda Coutinho de Ponte

$f(n) = g(n) + h(n)$

$X_i \geq G + d_i^+ - d_i^-, d_i^\pm \geq 0$

$\text{Min Max } Z = \{P_{x1}^\pm \frac{d_{x1}^\pm}{t_{x1}}, P_{x2}^\pm \frac{d_{x2}^\pm}{t_{x2}}, P_{x3}^\pm \frac{d_{x3}^\pm}{t_{x3}}\}$

Maio de 2023

a) Solução admissível ao problema

Neste problema, uma solução admissível consiste em atribuir a cada um dos 10 cientistas um e apenas um dos 10 projetos de investigação disponíveis. Conforme a aptidão de cada cientista num determinado projeto, queremos tentar maximizar a aptidão total, na tentativa de desenvolver os medicamentos da **Lusa_Med** de forma eficiente.
Como a matriz das aptidões é do tipo ($n \times n = 10 \times 10$), existem 10 cientistas para 10 projetos e, por isso, todos os cientistas vão ter um projeto associado.

b) Heurística para determinar uma solução admissível ao problema

— **Passo 1** : Calcular a média da aptidão de cada um dos cientistas

— **Passo 2** : Ordenar a nova coluna da média da aptidão de cada cientista por ordem crescente

— **Passo 3** : Tendo as médias ordenadas, começamos pelos cientistas com pior aptidão média, até chegarmos ao com melhor

— **Passo 4** : Ir associando um cientista a um e só um projeto de investigação começando pelos cientistas com pior aptidão média, mas pegando o melhor valor de aptidão nos projetos, em que cada iteração vai retirando o projeto de investigação já atribuído

— **Passo 5** : Calcular o nível de aptidão total para a solução admissível

Após este ciclo iremos ter uma solução admissível para o problema que iremos observar abaixo.

c) Solução admissível para o problema

Primeiramente, começamos por calcular as médias de aptidão de cada cientista e colocar em uma tabela e, em seguida, ordenar essa coluna para estabelecer uma ordem para a resolução da solução admissível. Após isso, serão seguidos os passos descritos em *b*), associando cada cientista a um projeto na heurística construtiva estabelecida.

TABLE 1 – Tabela de cientistas com média de aptidão

Cientista	**Média de aptidão**
C_1	72.4
C_2	69.9
C_3	70.1
C_4	66.2
C_5	72.1
C_6	69.2
C_7	78.4
C_8	71.8
C_9	66.8
C_{10}	69.9

Com a tabela disponível, observamos que a ordem crescente dos cientistas a serem atribuídos os projetos é a seguinte :

$$C_4, C_9, C_6, C_2, C_{10}, C_3, C_8, C_5, C_1, C_7$$

Em primeiro lugar, atribuíremos ao cientista 4 o seu melhor projeto de investigação que, neste caso é o projeto 8, com 100 de aptidão.

Lista com os projetos já atribuídos : $\{P_8\}$

Para o cienstista 9 aplicamos o mesmo processo, em que o maior nível de aptidão é no projeto 1, com 90 de aptidão.

Lista com os projetos já atribuídos : $\{P_8, P_1\}$

O cientista 6 vai ser atribuído ao projeto 5, sendo este o projeto com maior nível de aptidão(94) e que também não foi escolhido.

Lista com os projetos já atribuídos : $\{P_8, P_1, P_5\}$

O próximo cientista é o 2 que irá receber o projeto 2, visto que o projeto 5 já está na lista de selecionados, então passamos para o segundo melhor que será o 2.

Lista com os projetos já atribuídos : $\{P_8, P_1, P_5, P_2\}$

Para o cientista 10, será selecionado o projeto 7, sendo o projeto com melhor nível de aptidão dos que não estão já atribuídos na lista (projeto 5 é melhor mas já foi atribuído).

$$\text{Lista com os projetos já atribuídos : } \{P_8, P_1, P_5, P_2, P_7\}$$

No cientista 3 teremos o projeto 9, dado que tem um nível de aptidão de 100 (máximo) e que ainda não foi selecionado.

$$\text{Lista com os projetos já atribuídos : } \{P_8, P_1, P_5, P_2, P_7, P_9\}$$

Tal como no cientista 3, o cientista 8 ainda tem o seu projeto com maior capacidade em termos de aptidão disponível e, por isso, atribuímos o projeto 3.

$$\text{Lista com os projetos já atribuídos : } \{P_8, P_1, P_5, P_2, P_7, P_9, P_3\}$$

Agora, no cienstista 5, restam apenas três projetos em que o com melhor aptidão é de 77, no projeto 4.

$$\text{Lista com os projetos já atribuídos : } \{P_8, P_1, P_5, P_2, P_7, P_9, P_3, P_4\}$$

Para o cienstista só podem ser alocados os projetos 6 ou 10, mas será alocado o projeto 10, com 85 de aptidão.

$$\text{Lista com os projetos já atribuídos : } \{P_8, P_1, P_5, P_2, P_7, P_9, P_3, P_4, P_{10}\}$$

Por último, e sem opção de escolha, será atribuído ao cientista 7 (que tem melhor aptidão média) o projeto 6, com 77 de aptidão. Assim, os projetos disponíveis já foram todos adicionados à lista de projetos, como podemos ver :

$$\text{Lista com os projetos já atribuídos : } \{P_8, P_1, P_5, P_2, P_7, P_9, P_3, P_4, P_{10}, P_6\}$$

Assim, ficaremos com a seguinte alocação de projetos a cientistas. No *python* iremos ter a solução dada como uma lista de projetos, onde o index é o cientista por ordem de 1 a 10.

TABLE 2 – Solução admissível

Cientistas	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}
Projetos	P_{10}	P_2	P_9	P_8	P_4	P_5	P_6	P_3	P_1	P_7

A aptidão total é dada pelo somatório de todas as aptidões dos cientistas nos projetos a que foram alocados, sendo que, para a solução apresentada, foi retornado o valor de 886.

d) Definir estrutura de vizinhança

O conceito de vizinhança é explicado quando, dada uma solução $s \in S$, há um conjunto de soluções vizinhas, que define uma **estrutura de Vizinhança** $\aleph(s)$, definida pela função $\aleph : S \Rightarrow 2^S$.

Neste problema, a troca de cientistas alocados a projetos só pode ser realizada após uma solução admissível. Para esse efeito, utilizaremos a solução admissível obtida em c) como ponto de partida para a troca de projetos entre cientistas. Ou seja, por outras palavras, seria trocar o projeto associado ao cientista 1 com o projeto associado ao cientista 2, resultando num novo valor de aptidão e de solução admissível.

Assim, da forma como foi definida a estrutura de vizinhança, existem $\binom{10}{2} = 45$ soluções vizinhas possíveis para a solução admissível atual (a troca do projeto do cientista 1 com todos, a do cientista 2 com todos exceto o 1, a do cientista 3 com todos exceto o 1 e o 2, etc...). Ou seja, semelhante a dizermos o seguinte :

$$\sum_{x=1}^{10-1} x = 45$$

e) Solução vizinha à solução em *c)*

Uma solução vizinha, tal como demonstrado na estrutura de vizinhança, será a de trocar um projeto alocado a um cientista x, a outro projeto distinto alocado a um cientista y.

Se quiséssemos trocar os projetos do cientista 7 com o do cientista 10 teríamos a nova solução demonstrada na tabela abaixo :

TABLE 3 – Solução vizinha com projetos de C_7 e C_{10} trocados

Cientistas	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}
Projetos	P_{10}	P_2	P_9	P_8	P_4	P_5	P_7	P_3	P_1	P_6
Aptidão	85	83	100	100	77	94	90	100	90	70

Como podemos observar face à tabela 3, foi acrescentada uma linha com o valor de aptidão dado a cada associação cientista-projeto. Ainda, representado por células cor-de-laranja, temos as trocas efetuadas, tal como os valores das novas aptidões das ligações que foram retiradas da tabela. O valor de aptidão total desta solução vizinha $\hat{s}$ foi de 889, 3 pontos superior ao da solução s, apresentada em c).

f) Definição da lista Tabu

A lista tabu guarda pares de pontos/vértices que correspondem às últimas trocas no algoritmo. Por isso, esta lista guarda apenas os últimos movimentos, com base no tamanho da lista, podendo designá-la de um algoritmo de memória curta. O tamanho desta lista é variável, assumindo que dará soluções diferentes para diferentes tamanhos de listas, pois o algoritmo tem o risco de ficar num máximo local ou de criar um ciclo como veremos de seguida (tamanho da lista não deve ser demasiado extenso porque assim o modelo perde a sua utilidade, não procurando o máximo global porque os movimentos ficam muito tempo armazenados na lista).

Para o problema em questão, iremos adicionar à lista tabu a última troca efetuada de projetos de cientistas que, no contexto da aplicação, será o índice da nossa lista de projetos (cientistas). O processo inicia-se na solução inicial e, por isso, no 1º ciclo/iteração a solução admissível é igual à inicial, mas que será trocada para a melhor solução caso o valor de aptidão total seja superior.

g) Atualização da lista Tabu, considerando as alíneas anteriores

Para a troca considerada na alínea *e)*, na solução vizinha, e considerando que essa passaria a ser a nova solução atual, teríamos a seguinte atualização na lista tabu :

$$\textbf{Lista Tabu}=\{C_7, C_{10}\}$$

Ou seja, teríamos adicionado à lista tabu as trocas dos projetos entre o cientista 7 e o cientista 10, ficando a nova solução atual/vizinha da seguinte forma :

$$[P_{10}, P_2, P_9, P_8, P_4, P_5, P_7, P_3, P_1, P_6]$$

Importante salientar que como só foi feita apenas uma solução vizinha, que se tornou na solução atual, a lista tabu antes desta "iteração" estava vazia e, por isso, só tem um par com os cientistas cujos projetos foram trocados. Para o próximo ciclo (2ª iteração) começaremos com :

— **Solução atual** :

Alocação de cientista-projeto $\Rightarrow$ $SA = [(C_1 : P_{10}),\ (C_2 : P_2),\ (C_3 : P_9),\ (C_4 : P_8),\ (C_5 : P_4),\ (C_6 : P_5),\ (C_7 : P_7),\ (C_8 : P_3),\ (C_9 : P_1),\ (C_{10} : P_6)]$

Nível de aptidão total : 889

— **Melhor solução** :

Como o nível de aptidão total da solução atual > nível de aptidão total da solução inicial, há uma atualização na melhor solução (889 > 886).

$S^* = [(C_1 : P_{10}),\ (C_2 : P_2),\ (C_3 : P_9),\ (C_4 : P_8),\ (C_5 : P_4),\ (C_6 : P_5),\ (C_7 : P_7),\ (C_8 : P_3),\ (C_9 : P_1),\ (C_{10} : P_6)]$

Nível de aptidão total : 889

— **Atualização da lista tabu** : $\{\} \Rightarrow \{C_7, C_{10}\}$ (porque a solução vizinha foi obtida trocando os projetos do cientista 7 e 10)

h) Movimentos Tabu, considerando as alíneas anteriores

Os movimentos tabu correspondem àqueles que estão presentes na lista e que não poderão ser trocados até serem retirados da lista. Um movimento seria tabu se implicasse trocar os projetos entre um par de cientistas que já se encontra na lista tabu. Estes movimentos deixam de ser tabu quando a lista atinge o tamanho máximo definido e são retirados para que tentem atingir novos máximos globais.

i) Implementação do algoritmo Tabu

É pretendido implementar este algoritmo, de forma a atingir o máximo global da aptidão para este problema da **Lusa_Med**. A imagem seguinte, retirada deste link, sugere isso mesmo :

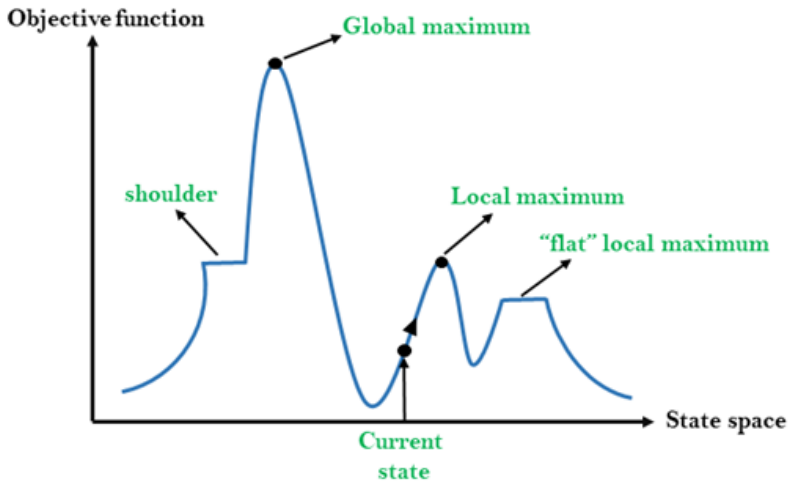

FIGURE 1 – Algoritmo de procura tabu e máximo global

Para esta alínea será utilizado o *python* com o intuito de aplicar o algoritmo tabu para o problema em questão. Abaixo é apresentado o pseudo-código para demonstrar de uma forma mais intuitiva como foi planeado e executado o algoritmo.

Algorithm 1 : Algoritmo de pesquisa tabu

```
Solução atual ← Variável
Tamanho da lista tabu ← Variável
Lista tabu ← Lista Vazia
Melhor solução atual ← Começa com a solução inicial
Aptidão melhor solução atual ← Começa com a aptidão da solução inicial
Melhor troca ← Começa com um par (0,0)
Iteração ← 0
while not critério de paragem do
    Estrutura de vizinhança ← Todos os vizinhos da estrutura de vizinhança definida
    Melhor vizinho ← Começa vazio
    Melhor aptidão vizinhança ← Valor muito baixo
    for trocas efetuadas na estrutura de vizinhança do
        if troca não está na lista tabu then
            vizinho ← cópia da solução atual fazer trocas de projetos dos cientistas e alterar em vizinho
            Aptidão do vizinho ← Calcular aptidão do vizinho
            if Melhor vizinho está vazio ou Aptidão do vizinho maior que Melhor aptidão vizinhança
then
                Melhor troca ← Troca
                Melhor vizinho ← Vizinho
                Melhor aptidão vizinhança ← Aptidão vizinho
    Adicionar à Lista tabu a Melhor troca
    if tamanho da Lista tabu maior que o Tamanho da lista tabu definido na variável then remover
primeiro elemento da lista
    Solução atual ← Melhor vizinho encontrado
    if Melhor aptidão vizinhança maior que Melhor aptidão total then
        Melhor solução ← cópia do melhor vizinho
        Melhor aptidão total ← Melhor aptidão vizinhança
        Iteração máxima ← Iteração somado a 1 para corrigir o índice de 0
    Iteração ← Incrementar uma Iteração à variável guardada
return Melhor solução encontrada durante o ciclo e a sua aptidão
```

Através do algoritmo apresentado, há três componentes que podem ser alteráveis, sendo estas o tamanho da lista tabu, a solução inicial (que, neste caso, foi a obtida em *c)*) e o critério de paragem. Esta última é-nos dita no enunciado na forma em que ou o algoritmo atinge 100 iterações, ou a aptidão atinje um valor superior a 850 de aptidão total. Contudo, como a solução inicial já teria sido superior a 850 (886), iremos ignorar esta restrição para o critério de paragem, deixando apenas o número de iterações.

Com isto dito, serão mostradas abaixo execuções deste algoritmo para tamanhos de lista tabu com 2, 3 e 4 elementos, sempre para a solução inicial definida na heurística em *b)* e mostrada passo a passo em *c)*.

Tamanho da lista tabu : 2

Para um tamanho de lista igual a 2, na primeira iteração será colocada uma troca de projetos entre cientistas, tal como na segunda iteração, estando a lista tabu com tamanho 2. Contudo, a partir da 3ª iteração, a lista apenas guardará os últimos dois movimentos, apelidando um nome de algoritmo de memória curta.

Para este tamanho de lista o valor de aptidão máximo atingido foi na segunda iteração, com um valor de 894 de aptidão total. A alocação cientista-projeto ficou da seguinte forma :

$[(C_1 : P_{10}), (C_2 : P_2), (C_3 : P_9), (C_4 : P_8), (C_5 : P_7), (C_6 : P_5), (C_7 : P_4), (C_8 : P_3), (C_9 : P_1), (C_{10} : P_6)]$

Em anexo conseguimos ver a tabela até onde esta atinge o tamanho máximo, mas que é facilmente visualizada no gráfico abaixo :

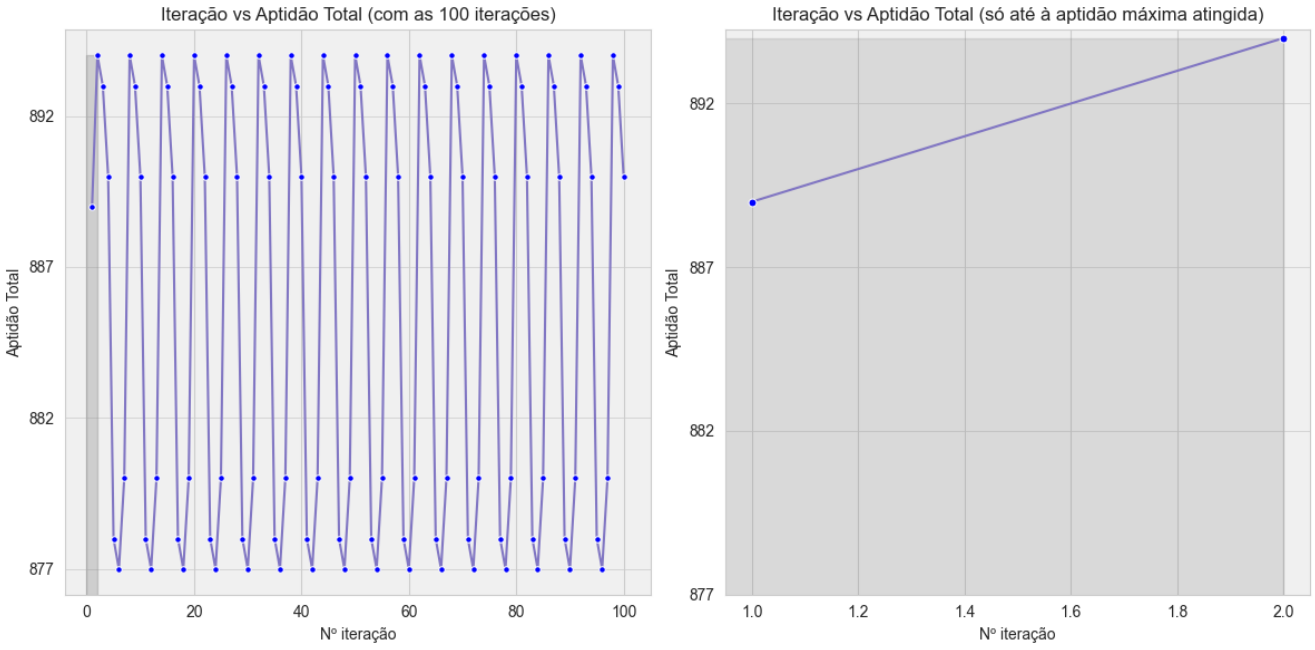

FIGURE 2 – Aptidão total ao longo das iterações (tamanho lista tabu igual a 2)

Tal como foi dito, a aptidão máxima atingida para um tamanho de lista tabu igual a 2 foi na segunda iteração e, por essa mesma razão, o gráfico ampliado (da direita) só mostra 2 iterações percorridas.

No gráfico à esquerda, com as 100 iterações, verificamos que o processo entra num ciclo que pode inferir que o tamanho da lista é demasiado pequeno e por isso acha que a solução que é retirada deve ser a melhor a ser colocada.

Tamanho da lista tabu : 3

Se o tamanho da lista tabu for mudado para 3, o algoritmo passa a ter uma memória um pouco maior, guardando mais elementos na lista tabu. Os resultados para esta mudança foram bastante diferentes dos obtidos para um tamanho de 2, onde a aptidão total agora chegou a 902, na 14ª iteração. Para mais detalhes observamos o gráfico abaixo com o comportamento da aptidão até atingir o melhor valor desta nas 100 iterações. Os outputs encontram-se em anexo.

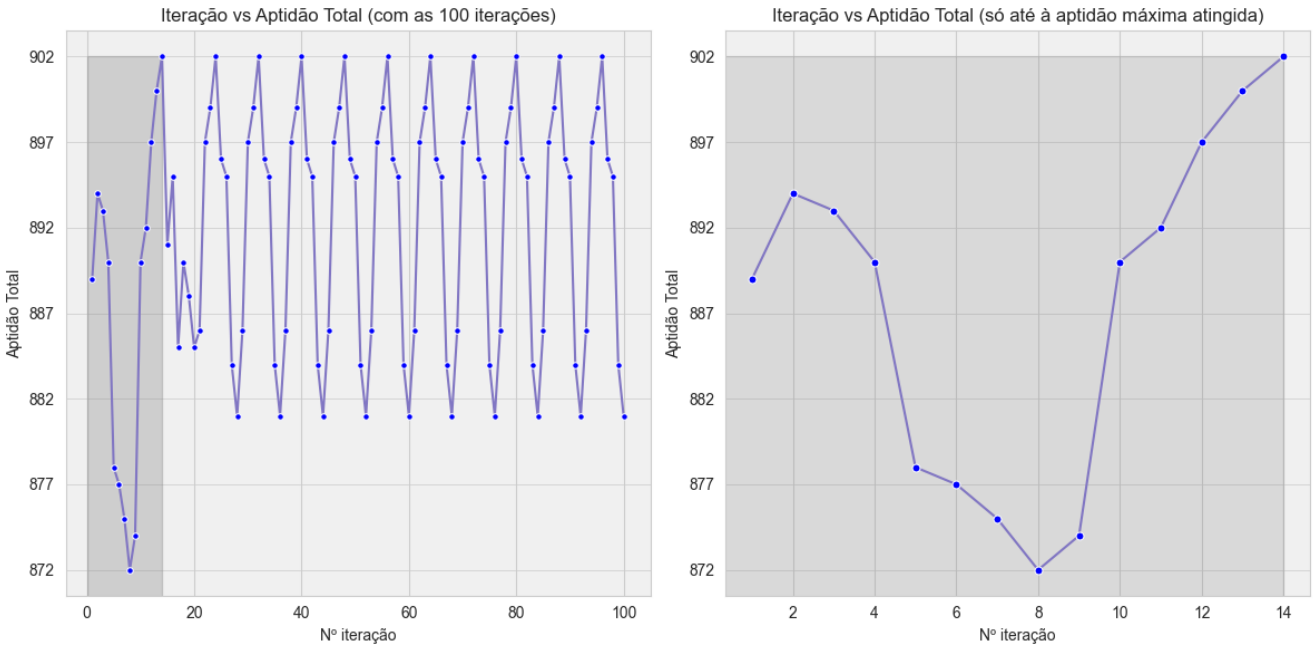

FIGURE 3 – Aptidão total ao longo das iterações (tamanho lista tabu igual a 3)

Tal como para um tamanho da lista tabu de 2, identificamos um ciclo após o algoritmo percorrer um certo número de iterações. Isto ocorre porque o algoritmo retira os valores da lista rapidamente (em 3 iterações) e, por isso, tende a atingir um novo máximo, mas sem sucesso, visto que fica "preso" nesse ciclo.

Testamos agora para um tamanho da lista tabu em 4.

Tamanho da lista tabu : 4

Para um tamanho de lista tabu com 4 elementos os resultados serão bastante semelhantes aos obtidos com um tamanho de 3, diferenciando só na iteração em que o valor de aptidão de 902 foi obtido. Este valor foi o máximo atingido nas 100 iterações, sendo atingido pela primeira vez na 16ª iteração. Assim como nos tamanhos anteriores será mostrado o gráfico com a evolução da aptidão durante as iterações/atualização da lista tabu.

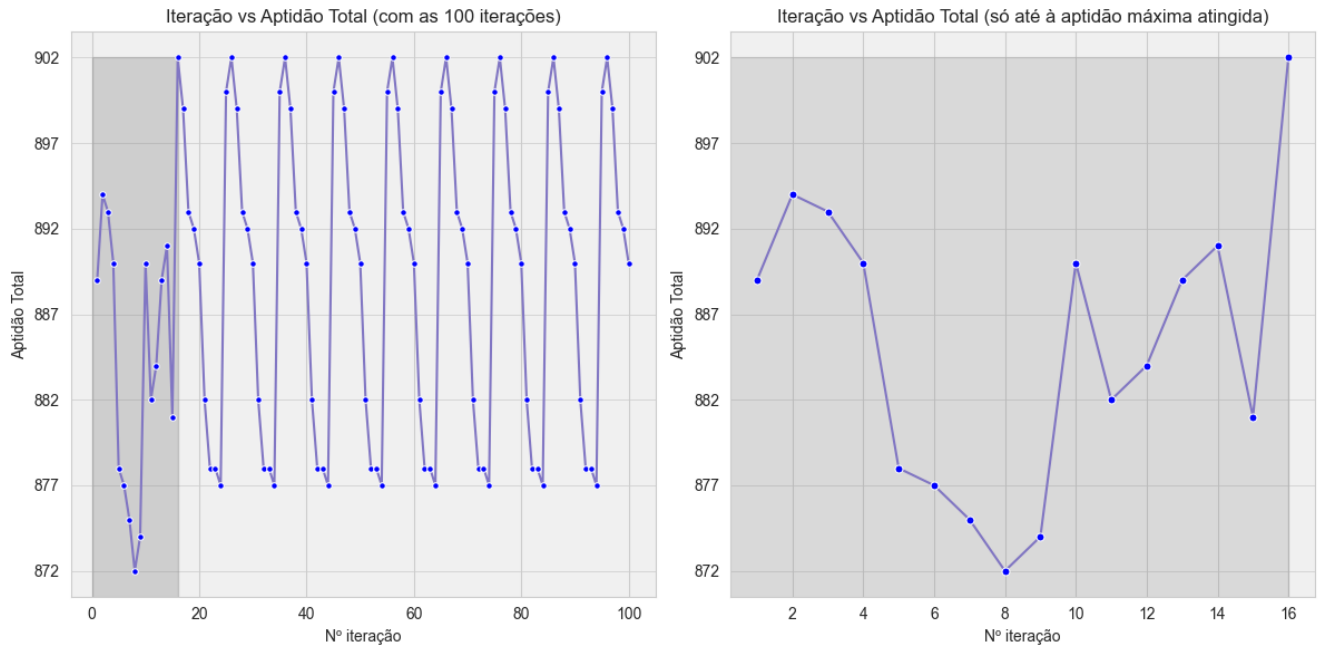

FIGURE 4 – Aptidão total ao longo das iterações (tamanho lista tabu igual a 4)

Do mesmo modo que para os outros tamanhos de listas, a partir de um certo momento, os movimentos tabu inseridos e retirados são os mesmos, entrando num ciclo repetitivo. Contudo, por outro lado, vimos que para este tamanho o algoritmo demorou mais 2 iterações a atingir o valor de aptidão 902. Por isso, não iremos aumentar mais o tamanho da lista, mesmo que o ciclo enunciado seja quebrado para tamanhos de listas maiores. Como chegamos a um valor igual para dimensões da lista tabu diferentes, pode concluir-se que o máximo global deste problema é 902, pelo que paramos o processo aqui. Os restantes resultados encontram-se em anexo.

NOTA : A heurística e a estrutura de vizinhança foram também criadas no *script python* enviado de forma a facilitar a compreensão dos mesmos. Foi também utilizada uma versão do *python* superior à 3.9 para executar o código desenvolvido.

Anexos

```
============================================================
Solução inicial: [10, 2, 9, 8, 4, 5, 6, 3, 1, 7]
Nível de aptidão total da solução inicial: 886
Começando na 1ª iteração...
============================================================
Melhor solução encontrada: [10, 2, 9, 8, 7, 5, 4, 3, 1, 6]
Nível de aptidão total da melhor solução: 894
Iteração em que a melhor aptidão foi gerada: 2
Demorou cerca de 0.066 segundos a realizar as 100 iterações.
============================================================
```

(a) Output algoritmo para tamanho da lista em 2

```
============================================================
Solução inicial: [10, 2, 9, 8, 4, 5, 6, 3, 1, 7]
Nível de aptidão total da solução inicial: 886
Começando na 1ª iteração...
============================================================
Melhor solução encontrada: [9, 2, 6, 8, 7, 4, 10, 3, 1, 5]
Nível de aptidão total da melhor solução: 902
Iteração em que a melhor aptidão foi gerada: 14
Demorou cerca de 0.05 segundos a realizar as 100 iterações.
============================================================
```

(b) Output algoritmo para tamanho da lista em 3

```
============================================================
Solução inicial: [10, 2, 9, 8, 4, 5, 6, 3, 1, 7]
Nível de aptidão total da solução inicial: 886
Começando na 1ª iteração...
============================================================
Melhor solução encontrada: [9, 2, 6, 8, 7, 4, 10, 3, 1, 5]
Nível de aptidão total da melhor solução: 902
Iteração em que a melhor aptidão foi gerada: 16
Demorou cerca de 0.055 segundos a realizar as 100 iterações.
============================================================
```

(c) Output algoritmo para tamanho da lista em 4

Iteração	Lista Tabu	Solução Atual	Aptidão Total
1	['(C7, C10)']	[10, 2, 9, 8, 4, 5, 7, 3, 1, 6]	889
2	['(C7, C10)', '(C5, C7)']	[10, 2, 9, 8, 7, 5, 4, 3, 1, 6]	894

(d) Tabela com as iterações até à aptidão máxima para tamanho da lista em 2

Iteração	Lista Tabu	Solução Atual	Aptidão Total
1	['(C7, C10)']	[10, 2, 9, 8, 4, 5, 7, 3, 1, 6]	889
2	['(C7, C10)', '(C5, C7)']	[10, 2, 9, 8, 7, 5, 4, 3, 1, 6]	894
3	['(C7, C10)', '(C5, C7)', '(C2, C10)']	[10, 6, 9, 8, 7, 5, 4, 3, 1, 2]	893
4	['(C5, C7)', '(C2, C10)', '(C5, C9)']	[10, 6, 9, 8, 1, 5, 4, 3, 7, 2]	890
5	['(C2, C10)', '(C5, C9)', '(C2, C3)']	[10, 9, 6, 8, 1, 5, 4, 3, 7, 2]	878
6	['(C5, C9)', '(C2, C3)', '(C3, C10)']	[10, 9, 2, 8, 1, 5, 4, 3, 7, 6]	877
7	['(C2, C3)', '(C3, C10)', '(C9, C10)']	[10, 9, 2, 8, 1, 5, 4, 3, 6, 7]	875
8	['(C3, C10)', '(C9, C10)', '(C2, C6)']	[10, 5, 2, 8, 1, 9, 4, 3, 6, 7]	872
9	['(C9, C10)', '(C2, C6)', '(C1, C6)']	[9, 5, 2, 8, 1, 10, 4, 3, 6, 7]	874
10	['(C2, C6)', '(C1, C6)', '(C6, C7)']	[9, 5, 2, 8, 1, 4, 10, 3, 6, 7]	890
11	['(C1, C6)', '(C6, C7)', '(C9, C10)']	[9, 5, 2, 8, 1, 4, 10, 3, 7, 6]	892
12	['(C6, C7)', '(C9, C10)', '(C2, C10)']	[9, 6, 2, 8, 1, 4, 10, 3, 7, 5]	897
13	['(C9, C10)', '(C2, C10)', '(C5, C9)']	[9, 6, 2, 8, 7, 4, 10, 3, 1, 5]	900
14	['(C2, C10)', '(C5, C9)', '(C2, C3)']	[9, 2, 6, 8, 7, 4, 10, 3, 1, 5]	902

(a) Tabela com as iterações até à aptidão máxima para tamanho da lista em 3

Iteração	Lista Tabu	Solução Atual	Aptidão Total
1	['(C7, C10)']	[10, 2, 9, 8, 4, 5, 7, 3, 1, 6]	889
2	['(C7, C10)', '(C5, C7)']	[10, 2, 9, 8, 7, 5, 4, 3, 1, 6]	894
3	['(C7, C10)', '(C5, C7)', '(C2, C10)']	[10, 6, 9, 8, 7, 5, 4, 3, 1, 2]	893
4	['(C7, C10)', '(C5, C7)', '(C2, C10)', '(C5, C9)']	[10, 6, 9, 8, 1, 5, 4, 3, 7, 2]	890
5	['(C5, C7)', '(C2, C10)', '(C5, C9)', '(C2, C3)']	[10, 9, 6, 8, 1, 5, 4, 3, 7, 2]	878
6	['(C2, C10)', '(C5, C9)', '(C2, C3)', '(C3, C10)']	[10, 9, 2, 8, 1, 5, 4, 3, 7, 6]	877
7	['(C5, C9)', '(C2, C3)', '(C3, C10)', '(C9, C10)']	[10, 9, 2, 8, 1, 5, 4, 3, 6, 7]	875
8	['(C2, C3)', '(C3, C10)', '(C9, C10)', '(C2, C6)']	[10, 5, 2, 8, 1, 9, 4, 3, 6, 7]	872
9	['(C3, C10)', '(C9, C10)', '(C2, C6)', '(C1, C6)']	[9, 5, 2, 8, 1, 10, 4, 3, 6, 7]	874
10	['(C9, C10)', '(C2, C6)', '(C1, C6)', '(C6, C7)']	[9, 5, 2, 8, 1, 4, 10, 3, 6, 7]	890
11	['(C2, C6)', '(C1, C6)', '(C6, C7)', '(C5, C6)']	[9, 5, 2, 8, 4, 1, 10, 3, 6, 7]	882
12	['(C1, C6)', '(C6, C7)', '(C5, C6)', '(C9, C10)']	[9, 5, 2, 8, 4, 1, 10, 3, 7, 6]	884
13	['(C6, C7)', '(C5, C6)', '(C9, C10)', '(C2, C10)']	[9, 6, 2, 8, 4, 1, 10, 3, 7, 5]	889
14	['(C5, C6)', '(C9, C10)', '(C2, C10)', '(C2, C3)']	[9, 2, 6, 8, 4, 1, 10, 3, 7, 5]	891
15	['(C9, C10)', '(C2, C10)', '(C2, C3)', '(C6, C9)']	[9, 2, 6, 8, 4, 7, 10, 3, 1, 5]	881
16	['(C2, C10)', '(C2, C3)', '(C6, C9)', '(C5, C6)']	[9, 2, 6, 8, 7, 4, 10, 3, 1, 5]	902

(b) Tabela com as iterações até à aptidão máxima para tamanho da lista em 4

Licenciatura em Ciência de Dados
Optimização Heurística – 2022/ 23
Trabalho Individual 2

Enunciado

A companhia farmacêutica **Lusa_Med** detém os direitos sobre dez projetos I&D, **P1**, …, **P10**, e pretende iniciá-los na tentativa de desenvolver novos medicamentos para o tratamento de dez tipos específicos de doenças. Cada projeto necessita de um coordenador distinto para o liderar e, atendendo à exigência dos projetos, cada coordenador só poderá liderar um só projeto. A **Lusa_Med** já selecionou dez cientistas seniores, **C1**, …, **C10**, e pretende saber como deve alocar os cientistas aos projetos. Para o efeito, a companhia elaborou uma tabela com a aptidão de cada cientista para liderar cada um dos projetos (medida na escala 0-100):

Cientistas	**Projetos I&D**									
	P1	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**	**P10**
C1	70	65	55	50	90	67	80	62	100	85
C2	74	83	54	60	100	75	50	76	82	45
C3	71	87	66	58	74	81	48	52	100	64
C4	50	89	78	63	51	40	48	100	68	75
C5	100	66	83	77	54	58	93	89	53	48
C6	80	55	70	65	94	47	60	88	73	60
C7	87	63	90	79	47	77	90	76	85	90
C8	67	95	100	40	70	54	70	100	65	57
C9	90	45	88	48	65	68	80	46	71	67
C10	67	77	50	60	100	70	80	60	65	70

A **Lusa_Med** pretende determinar a alocação dos cientistas aos projetos que maximiza a aptidão total. Na tentativa de atingir este objetivo, a companhia irá definir e implementar um algoritmo de Pesquisa Tabu. Para o efeito:

a) Descreva por palavras em que consiste uma solução admissível para o problema.

[1.5 valores]

b) Defina uma heurística que lhe permita determinar uma solução admissível para o problema. **[3.0 valores]**

c) Tendo em conta a alínea **b)**, determine uma solução admissível para o problema.

[1.0 valor]

d) Defina a estrutura de vizinhança de uma solução. **[2.0 valores]**

e) Tendo em conta a alínea **d)**, determine uma solução vizinha da solução que apresentou na alínea **c)**. **[1.0 valor]**

f) Tendo em conta as opções tomadas para responder às alíneas anteriores, defina a Lista Tabu. **[1.5 valores]**

g) Assumindo que a solução vizinha apresentada na alínea **e)** passaria a ser a nova solução atual, indique como deveria ser atualizada a Lista Tabu. **[1.0 valor]**

h) Tendo em conta as opções tomadas para responder às alíneas anteriores, defina os movimentos Tabu. **[1.5 valores]**

i) Tendo em conta as opções tomadas para responder às alíneas anteriores, implemente o algoritmo de Pesquisa Tabu, tomando como critérios de paragem um número máximo de 100 iterações ou a obtenção de uma solução admissível cujo valor de aptidão total seja pelo menos 850. **[7.5 valores]**