INSTITUTO UNIVERSITÁRIO DE LISBOA

Ano Letivo 2022/2023

26 de maio de 2023

Trabalho Individual II

Metaheurísticas

UC Otimização Heurística

Licenciatura Ciência de Dados

CDB1

Docente

Anabela Costa

André Filipe Gomes Silvestre Nº104532

Índice

Introdução

Metaheurísticas

O termo Metaheurística foi introduzido em 1986 por *Fred Glover* e, apesar de inúmeras definições para este termo, podemos referir como um *framework* para um algoritmo geral que pode ser aplicado a diferentes problemas de otimização, e com um número ínfimo de modificações, o algoritmo pode ser adaptado a um problema específico.

Assim, podemos considerar Metaheurísticas como uma técnica de otimização que procuram soluções admissíveis, sem compromisso de ser a solução ótima, que são especialmente úteis em problemas que não têm uma solução analítica ou que são muito complexos para serem resolvidos com técnicas tradicionais de otimização.

Uma das Metaheurísticas que pode ser utilizada é o Algoritmo de Pesquisa Tabu, que é uma técnica baseada no *Procedimento de Pesquisa Local* que utiliza uma lista tabu para evitar retornar a soluções que já foram visitadas anteriormente. A Pesquisa Tabu tem sido amplamente utilizada em problemas de otimização combinatória, incluindo alocação de recursos e programação de tarefas. [1]

Neste sentido, foi proposto no âmbito da UC de *Otimização Heurística* inserida na licenciatura de Ciência de Dados, este trabalho que visa dar resposta a um problema de otimização da farmacêutica Lusa_Med que pretende determinar a afetação dos cientistas aos projetos que maximiza a aptidão total.

Resolução

a) Solução Admissível

Neste problema, determinar uma solução admissível consiste em:

- Determinar a afetação de cada um dos 10 Cientistas seniores ($\mathbf{C1}, \ldots, \mathbf{C10}$) a um e um só Projeto I&D ($\mathbf{P1}, \ldots, \mathbf{P10}$) que maximiza a aptidão total.

Esta solução será admissível se cada projeto for liderado por um único cientista que não poderá liderar outro projeto simultaneamente, atendendo à exigência estabelecida.

b) e c) Heurística e Aplicação

Passo 1 – Selecionar os projetos com maior aptidão.

Calcular, para cada projeto, o máximo (máx.) de aptidão de afetação dos cientistas, salientando os que obtêm aptidão $\mathbf{100}$.

	Projetos I&D									
Cientistas	**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
Máx	100	95	100	79	100	81	93	100	100	90

Passo 2 – Alocar os cientistas aos projetos que tenham aptidão 100.

Para cada projeto de I&D, selecionar o coordenador com maior aptidão para liderá-lo, desde que esse coordenador ainda não tenha sido selecionado para liderar outro projeto.

Para maximizar a aptidão, vamos começar por favorecer as combinações (projeto, cientista) que tenham aptidão 100. Nos restantes casos, que já tenham sido alocados, passar ao cientista com maior valor seguinte que ainda não tenha sido alocado.

Seguindo este passo, começamos por afetar os cientistas C1, C2, C4, C5 e C8 aos projetos P9, P5, P8, P1 e P3, respetivamente.

- P1: C5
- P3: C8
- P5: C2
- P8: C4
- P9: C1

Fica a faltar a afetação dos cientistas C3, C6, C7, C9 e C10 aos projetos P2, P4, P6, P7, P10. Para tal, vamos encontrar nestas combinações de (projeto, cientista), para cada cientista, o projeto que tenha maior aptidão.

	Projetos I&D				
Cientistas	**P2**	**P4**	**P6**	**P7**	**P10**
C3	87	58	81	48	64
C6	55	65	47	60	60
C7	63	79	77	90	90
C9	45	48	68	80	67
C10	77	60	70	80	70

Este passo resulta na afetação dos cientistas C3, C6, C7, C9 e C10 aos projetos P2, P4, P7, P10 e P6, respetivamente.

Após a execução destes dois passos já temos uma **Solução Inicial Admissível** para o problema:

- **P1:** C5 (100)
- **P2:** C3 (87)
- **P3:** C8 (100)
- **P4:** C6 (65)
- **P5:** C2 (100)
- **P6:** C10 (70)
- **P7:** C7 (90)
- **P8:** C4 (100)
- **P9:** C1 (100)
- **P10:** C9 (67)

Aptidão Total de Afetação = $5 \times 100 + 87 + 65 + 70 + 90 + 67 = 879$

d) Vizinhança de uma Solução

Neste problema, a transição entre soluções adjacentes é direcionada pela mudança das alocações de cientistas orientadores.

Assim, podemos então definir como vizinhança:

- A troca de 2 cientistas orientadores de forma a obter todas as soluções distintas entre iterações.

e) Solução Vizinha (S.V.)

Tendo em conta a definição estabelecida na alínea **d)** uma possível solução vizinha pode ser obtida pela troca do coordenador do projeto **P9** de **C1** para **C3**. Assim a nova solução manter-se-á igual à inicial, alterando apenas o par **(P7, C7)** e **(P10, C9)** para **(P7, C9)** e **(P10, C7)**, respetivamente.

Aptidão Total de Afetação da S.V. = $879 - 67 - 90 + 80 + 90 = 892$

Por coincidência, esta **Solução Vizinha** apresenta uma aptidão total de afetação melhor do que a **Solução Inicial Admissível** calculada na alínea **c)**.

f) Lista Tabu

A Lista Tabu é uma lista de soluções que foram visitadas recentemente e que, portanto, não podem ser consideradas novamente durante um determinado período de iterações.

Esta é utilizada para evitar que o *Algoritmo de Pesquisa Tabu* fique preso em ciclos e para incentivar a exploração de outras soluções na vizinhança da solução atual.

Atendendo à definição de vizinhança, uma solução vizinha é gerada a partir de dois movimentos:

- Movimento de **remoção** de 2 pares de afetação (projeto, cientista);
- Movimento de **inserção** de 2 pares de afetação (projeto, cientista) diferentes dos removidos.

Na lista tabu, deve ficar um dos movimentos (qualquer um dos movimentos é válido).

Assim, podemos optar por definir a lista tabu como uma lista de pares (projeto, cientista) que, em cada iterada, são adicionados e originam a melhor solução vizinha da iterada.

Deste modo, no algoritmo estes pares serão proibidos de serem selecionados para liderar um projeto durante um determinado número de iterações.

g) Atualização da Lista Tabu

Assumindo que a solução vizinha apresentada na alínea **e)** passaria a ser a nova solução atual e que a lista tabu se encontra vazia, esta deveria ser atualizada com a inserção dos pares (P7, C9) e (P10, C7).

Assim, passará a ser:

Lista Tabu = {(P7, C9), (P10, C7)}

h) Movimentos Tabu

Contrariamente à Lista Tabu definida em **f)**, um *Movimento Tabu* é um movimento associado à remoção de 1 par de ligações da lista tabu.

> Por exemplo, tendo em consideração a lista tabu atualizada na alínea anterior, o movimento tabu será a remoção dos pares (P7, C9), (P10, C7) na iterada seguinte do algoritmo. Isto é, na iterada seguinte a melhor solução vizinha não pode ser resultante da remoção dos pares referidos.

i) Implementação do Algoritmo

Para implementar o Algoritmo de Pesquisa Tabu no *Python* criei as funções **calcular_aptidao_total**, **gerar_vizinhanca**, **encontrar_melhor_vizinho** e **pesquisa_tabu**.

Relativamente à função **calcular_aptidao_total**, recebendo como *input* uma solução em formato dicionário **{key:value}**, com os projetos a serem *chaves* e os cientistas *valores*, percorre todos os pares e calcula a aptidão total da solução, somando as aptidões individuais de cada projeto atribuído ao respetivo coordenador.

Quanto à função **gerar_vizinhanca**, esta gera a vizinhança de uma solução atual, dada como *input*, considerando todas as possibilidades de trocas entre dois cientistas atribuídos a projetos distintos. Neste caso, dado que apenas ocorrem 2 trocas em 10 possibilidades, o número de vizinhos será $C_2^{10} = 45$.

A função **encontrar_melhor_vizinho** escolhe o melhor vizinho dentro do conjunto de vizinhança, tendo em consideração uma lista tabu de trocas proibidas, garantindo que as mesmas não ocorram.

E por último, a função **pesquisa_tabu** implementa o *Algoritmo de Pesquisa Tabu*, recebendo como *input* uma solução inicial admissível. O algoritmo realiza iterações, analisa a vizinhança de soluções admissíveis, e mantem uma lista tabu para evitar movimentos repetidos. Nesta função define-se os critérios de paragem e atualiza-se a lista tabu, adicionado apenas as trocas (projeto, cientista) que deram origem à melhor solução vizinha.

Adicionalmente, na implementação da lista tabu remove-se os 1º e 2º elementos caso o tamanho da lista seja superior a 12 (valor escolhido após diversas tentativas de modo a tentar alcançar a melhor solução possível).

Após a execução do algoritmo, a função retorna a melhor solução encontrada, acompanhada da sua aptidão total, e ainda um `DataFrame` que contém todas as soluções iteradas durante o algoritmo, contendo a solução anterior, a solução atual, a aptidão total e a lista tabu utilizada em cada iteração.

Deste modo é possível analisar todas as soluções encontradas e a correta implementação do algoritmo.

Critério de Paragem Estipulados

Utilizando os critérios definidos no enunciado, o algoritmo terminará quando for atingido o número máximo de iterações de 100 ou a obtenção de uma solução admissível cujo valor de aptidão total seja de, pelo menos, 850.

Alteração dos Critérios de Paragem Estipulados

Após correr o código realizado pude observar que devido ao facto de minha solução inicial ter já aptidão total acima do valor de paragem, este algoritmo não chega a iterar nenhuma vez.

As 3 possíveis soluções para o problema referido passam por:

i. Retirar o critério de aptidão total, assumindo apenas o critério das iterações;
ii. Aumentar o valor do critério de aptidão mínima, de modo que a pesquisa tabu ocorra;
iii. Começar com uma solução com aptidão inferir a 850 (critério estipulado) para observar o funcionamento do algoritmo.

Tomei como solução alterar os critérios de paragem do algoritmo de modo a obter soluções melhores do que a inicial e puder ver o algoritmo funcionar.

Decidi, portanto, colocar como aptidão mínima **1000**, para que este critério não fizesse o algoritmo parar dado que não existe possibilidade de haver uma solução com esta aptidão; e, adicionalmente, adicionei um número máximo de iterações de **500** apenas para verificar e corroborar o correto comportamento do algoritmo. Isto apenas é possível, dada a reduzida dimensão do problema e a elevada capacidade computacional que temos atualmente à disposição.

Resultados

Após a execução do código com a solução inicial definida na alínea **c)**, e iterando 500 vezes o algoritmo obtemos como **Melhor Solução Admissível** para o problema a afetação:

- **P1:** C9 (90)
- **P2:** C2 (83)
- **P3:** C8 (100)
- **P4:** C6 (65)
- **P5:** C10 (100)
- **P6:** C3 (81)
- **P7:** C5 (93)
- **P8:** C4 (100)
- **P9:** C1 (100)
- **P10:** C7 (90)

Aptidão Total de Afetação = $90 + 83 + 65 + 81 + 93 + 90 + 4 \times 100 = 902$

Além desta melhor solução foram ainda encontradas soluções admissíveis com aptidões totais **900**, **899** e **897** que podem ser úteis para o agente decisor, caso haja alguma impossibilidade em afetar os pares (projetos, cientista) da melhor solução encontrada.

Conclusões

Sucintamente, o presente trabalho de otimização da afetação de cientistas a projetos na farmacêutica Lusa_Med foi realizado com sucesso, aplicando-se a metaheurística do *Algoritmo de Pesquisa Tabu*.

Esta abordagem permitiu encontrar soluções admissíveis com uma aptidão total significativamente melhor, maximizando assim a eficiência e a qualidade dos projetos desenvolvidos.

Ao utilizar uma técnica de otimização robusta e adaptável, foi possível superar as limitações das abordagens tradicionais e lidar eficazmente com problemas complexos e sem soluções analíticas.

Deste modo, o presente trabalho evidenciou a capacidade das metaheurística em fornecer soluções promissoras e viáveis, sem compromisso de serem soluções ótimas, porém, possibilitando futuras aplicações em diversas temáticas da vida quotidiana que pretenda utilizar a otimização nas tomadas de decisão.

Em suma, cumpri com sucesso o objetivo pretendido neste projeto de responder a um problema de metaheurística num caso de afetação de projetos de uma farmacêutica a cientistas orientadores.

Bibliografia

[1] Siarry, P. (Ed.) (2016). Metaheuristics, Springer.