UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
SME0510 - INTRODUÇÃO À PESQUISA OPERACIONAL

PROJETO

Heitor Pupim Assunção Toledo, nºUSP: 11372858
Rafael Scalon Peres Conti , nº USP: 11871181
Rafael Jun Teramae Dantas, nº USP: 12563686
Jade Bortot de Paiva , nº USP: 11372883

São Carlos

2022

SUMÁRIO

1 INTRODUÇÃO

O grupo escolheu modelar um problema de Sequenciamento de Tarefas (otimização linear). Para tal, o grupo utilizou a função **linprog** da biblioteca **scipy.optimize** , cujo código fonte pode ser encontrado no link a seguir: Código Fonte, e a documentação no link seguinte: Documentação.

Dessa forma, foi escolhido o problema de sequenciamento de tarefas para uma fábrica de bolachas cobertas por chocolate.

As variáveis são:

- t_i = Tempo de início da atividade i

As restrições são causadas pelos requisitos entre as variáveis, apresentadas na tabela a seguir (Figura 1):

	Atividades	Pré-requisitos	duração/horas
1	carregamento dos caminhoes com cacau	-	3
2	transporte do cacau	1	2
3	classificação dos grãos por qualidade	2	1
4	torrefação do cacau	3	1
5	moagem	4	1
6	prensagem da massa	5	2
7	adição de leite e açucar	6	1
8	refinação do chocolate	7	2
9	temperagem do chocolate	8	1
10	teste de qualidade do chocolate	9	3
11	dosagem dos ingredientes do biscoito	-	1
12	mistura dos ingredientes do biscoito	11	2
13	moldagem do biscoito	12	1
14	resfriamento do biscoito	13	4
15	teste de qualidade dos biscoitos	14	3
16	aplicação do chocolate no biscoito	10 e 15	2
17	teste de qualidade dos biscoitos cobertos por chocolate	16	3
18	transporte das embalagens	-	2
19	teste de qualidade das embalagens	18	3
20	embalagem dos biscoitos cobertos	17 e 19	1
21	teste de qualidade do produto final	20	3
22	carregamento dos caminhoes com o produto final	21	3
23	envio do produto para as distribuidoras	22	5

Figura 1 – Tabela de requisitos

2 SOBRE O SOLVER

A função utilizada pertence à biblioteca Open Source **Scipy** da linguagem **Python**, cujo propósito é trabalhar com arrays **Numpy** permitindo rotinas para otimização e resolução de problemas.

Além disso, o **scipy.optimize** possui funções que buscam resolver problemas relacionados à otimização, determinação de raizes e etc.

2.1 scipy.optimize.linprog:

O propósito desta função é minimizar uma função objetiva de um problema de otimização linear com restrições de igualdade ou desigualdade.

O **linprog** resolve problemas no formato:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.a:} \quad & A_{ub} x \leq b_{ub} \\ & A_{eq} X = b_{eq} \\ & l \leq x \leq u \end{aligned}$$

Em que x é um vetor de variáveis de decisão. c, b_{ub} b_{eq}, l e u são vetores. Ademais, A_{ub} e A_{eq} são matrizes.

O link para acesso do código fonte e da documentação do Solver: Código Fonte Documentação.

3 RESOLUÇÃO

Inspirados no problema de sequenciamento de tarefas apresentado em aula, o grupo decidiu modelar e resolver um problema de sequenciamento de tarefas para uma fábrica de produção de bolachas cobertas por chocolate. O problema escolhido foi este, pois buscávamos um modelo com grande quantidade de tarefas e com requisitos (dependências) múltiplos, além do grupo achar interessante um contexto industrial.

Desta forma, pensamos nas atividades necessárias para produção e nas interações entre elas de forma a desenvolver a tabela 1 supra-indicada. Ao final do processo de modelagem obtivemos 23 atividades; 22 restrições. O tempo das atividades foi estipulado em horas.

Com a modelagem finalizada, foi possível escrever as restrições e a função objetivo como se segue:

$$
\begin{aligned}
\min \quad & t_{23} \\
s.a. : & -1t_1 + 1t_2 \geq 3 \\
& -1t_2 + 1t_3 \geq 2 \\
& -1t_3 + 1t_4 \geq 1 \\
& -1t_4 + 1t_5 \geq 1 \\
& -1t_5 + 1t_6 \geq 1 \\
& -1t_6 + 1t_7 \geq 2 \\
& -1t_7 + 1t_8 \geq 1 \\
& -1t_8 + 1t_9 \geq 2 \\
& -1t_9 + 1t_{10} \geq 1 \\
& -1t_{11} + 1t_{12} \geq 1 \\
& -1t_{12} + 1t_{13} \geq 2 \\
& -1t_{13} + 1t_{14} \geq 1 \\
& -1t_{14} + 1t_{15} \geq 4 \\
& -1t_{10} + 1t_{16} \geq 3 \\
& -1t_{15} + 1t_{16} \geq 3 \\
& -1t_{16} + 1t_{17} \geq 2 \\
& -1t_{18} + 1t_{19} \geq 2 \\
& -1t_{17} + 1t_{20} \geq 3
\end{aligned}
$$

$$
\begin{aligned}
-1t_{19} + 1t_{20} &\geq 3 \\
-1t_{20} + 1t_{21} &\geq 1 \\
-1t_{21} + 1t_{22} &\geq 3 \\
-1t_{22} + 1t_{23} &\geq 3
\end{aligned}
$$

$$t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}, t_{14}, t_{15}, t_{16}, t_{17}, t_{18}, t_{19}, t_{20}, t_{21}, t_{22}, t_{23} \geq 0$$

Com isso, foi possível determinar a matriz A, que possui 22 linhas (restrições) e 23 colunas (variáveis), e a matriz b que possui 22 linhas (restrições) e uma coluna. Para tal, iniciamos as matrizes utilizando a função zeros da biblioteca **Numpy**, que cria matrizes ou vetores preenchidos por zero.

Em seguida, alteramos os valores da matriz A e b para representarem as restrições colocadas acima, além de criar um vetor c, também preenchido de zeros, somente com a último valor igual a 1 (de forma a representar a função objetivo).

A partir de A, b e c foi possível utilizar a **opt.linprog**, com parâmetros $c, -A, -b$ e $bounds = (0, None)$. É necessário utilizar $-A$ e $-b$, para transformar as retrições de maior ou igual ($\geq$) e bounds de zero a None para que todas as variáveis (t_i) sejam maiores que zero (sem apresentar valor máximo).

Essa função **linprog** foi chamada como igual a variável res, de forma que chamamos res para verificar que a otimização foi bem sucedida (message = 'Optimization terminated successfully.', success = True), em seguida somamos o valor de $res.fun$ (valor ótimo encontrado pelo solver, que representa o tempo de inicio da última tarefa, t_{23}) com o tempo da última tarefa, de forma a termos $res.fun + 5$, resultando em um valor ótimo de 34.00000015727363, ou seja, 34 horas.

4 CONCLUSÃO

Ao final do processo a otimização do modelo foi bem-sucedida (vide os conteúdos de message e success), o que indica que a modelagem do problema foi bem executada, além do uso do sinal negativo para A e b na entrada do solver ter sido uma escolha acertada. Ademais o valor ótimo obtido pelo solver (29.00000015727363) e o valor ótimo final, com o acréscimo do tempo para a realização da última tarefa (34.00000015727363) estão ambos dentro do esperado para o problema modelado (entre zero e o somatório do tempo de execução das tarefas). É importante salientar que os algarismos no fim de ambos os valores são resultados de erros devidos ao uso de variáveis do tipo **“float”**.

Portanto, o grupo conclui que tanto a modelagem do problema, quanto os resultados obtidos com o uso do solver foram satisfatórios e condizentes com o conteúdo apresentado nas aulas da disciplina.

REFERÊNCIAS

[1] SCIPY. **Scipy**. 2022. Documentação da função linprog. Disponível em: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linprog.html>. Acesso em: 11 de outubro de 2022.

[2] NUMPY. **Numpy**. 2022. Documentação da função zeros. Disponível em: <https://numpy.org/doc/stable/reference/generated/numpy.zeros.html>. Acesso em: 11 de outubro de 2022.

[3] SCIPY. **GitHub**. 2022. Código fonte do linprog. Disponível em: <https://github.com/scipy/scipy/blob/v1.9.2/scipy/optimize/$_l inprog.pyL168 - L673$>. Acesso em: 11 de outubro de 2022.

[4] SCIPY. **Scipy**. 2022. Documentação do optimize. Disponível em: <https://docs.scipy.org/doc/scipy/reference/optimize.html>. Acesso em: 11 de outubro de 2022.

[5] SCIPY. **Scipy**. 2022. Documentação do scipy. Disponível em: <https://docs.scipy.org/doc/scipy/tutorial/general.html>. Acesso em: 11 de outubro de 2022.