

Brief: A large car manufacturer, BWM, approaches you to optimise their production facilities. The production of a car involves it passing through a series of workstations, with workstations modifying the appearance of the car in some way (e.g. adding an engine/doors/lights/etc). Regardless of their type (e.g. saloon, estate, electric, convertible cars, etc), all cars go through the same workstations and in the same order. BWM wants you to find out what the most efficient processing order (permutation) of model types is within a single production facility. For example, should BWM first produce all saloon cars, then the estate cars, and then the electric ones, or should electric cars be produced before saloon cars and estate cars afterwards. This type of problem is known as Permutation Flow Shop Problem (PFSP).

Formally, a PFSP instance is given by a set of m machines $M_1, \ldots, M_m$ (m workstations in our case) and a set of n jobs $J_1, \ldots, J_n$ (n car model types in our case), where each job J_i has to be processed on machines $M_1, \ldots, M_m$ in that order, with processing time p_{ij} for processing job i on machine j. The objective is to find a job sequence $s = (s_1, s_2, \ldots., s_n)$ that minimises the total duration it takes until all the jobs have finished processing (or the completion time of all jobs, also known as the makespan). To compute the makespan ($C_{s_n m}$), we apply the following set of recursive equations:

$$C_{s_1 j} := \sum_{h=1}^{j} p_{s_1 h} \qquad j = 1, \ldots, m$$

$$C_{s_k 1} := \sum_{h=1}^{k} p_{s_h 1} \qquad k = 1, \ldots, n$$

$$C_{s_k j} := \max\{C_{s_{k-1} j}, C_{s_k j-1}\} + p_{s_k j} \quad k = 2, \ldots, n, \; j = 2, \ldots, m$$

The goal of an optimization algorithm applied to the PFSP is to find a sequence (permutation) of jobs S such that the makespan is as minimal as possible.

We demonstrate the computation of the makespan for a simple PFSP instance in the context of our car production problem: Assume we have n = 5 car models, m = 3 workstations, and processing times as shown in the left plot of Figure 1.

	J_1	J_2	J_3	J_4	J_5
p_{i1}	3	3	4	2	3
p_{i2}	2	1	3	3	1
p_{i3}	4	2	1	2	3

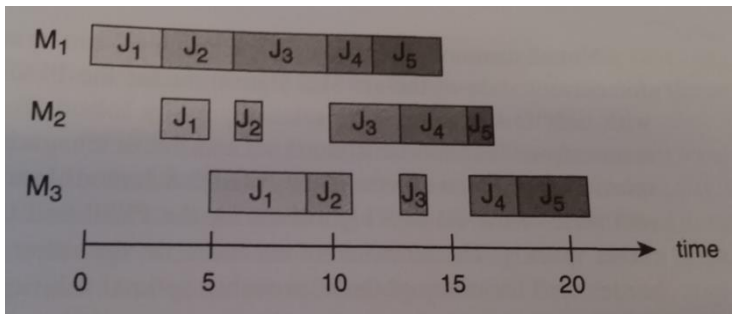

Figure 1: Processing times of jobs on 3 machines (left) and, on the right, a Gantt chart illustrating the schedule for the sequence S := (J_1, J_2, J_3, J_4, J_5). The three machines are numbered M_1, M_2 and M_3; job numbers are indicated in the boxes.

Using the recurrence relations from above, the makespan of the sequence S := (J_1, J_2, J_3, J_4, J_5) can be computed to be 21. The resulting schedule is graphically represented as a Gantt-chart in the right plot of Figure 1.

1. To get a feeling for the problem, you decide to test a simple algorithm (Random Search) and execute it for a number of solution evaluations.
   - Implement the problem and the algorithm in Python.
   - Apply the algorithm to each dataset using a maximum of $1000 \times n$ solution evaluations for each run.[1] Each solution evaluation (Line 3 in the Algorithm below) involves computing the makespan using the equations provided above.
   - Repeat each run with a different initial random seed 30 times and collect the makespan values of the best solution returned by each run.
   - Report in a table minimum, maximum, mean, and standard deviation of the makespan for each dataset.

[1] There are 5 datasets in total, all of which can be found in the file *"flowshop_bman73701.txt"* in the designated Blackboard Coursework folder.

- Discuss what can be observed from the results.

```
Algorithm: Random Search

Input: maximum number of function evaluations

Output: a single solution to the problem

1. generate and evaluate one random solution s_best
2. while maximum solution evaluations not reached do
3.       generate and evaluate one random solution s
4.       if s is better than s_best, replace s_best with s
5. end while
6. return s_best
```

2. You carry out a literature review and find a research paper from the 80s (Nawaz et al, 1983) that proposes one of the first heuristics, called NEH (Nawaz-Enscore-Ham), for the flow shop problem. Following this paper, you decide to implement and test NEH.
   - Implement the algorithm in Python following the description of the paper (page 92 in the paper). Bear in mind that the paper may use a different notation to the ones we use in this brief.
   - Apply the algorithm to each dataset using the stopping criteria reported in the paper (i.e. $n(n+1)/2 - 1$ enumerations).
   - Report in a table the makespan for each dataset.
   - Compare with the results obtained by Random Search and discuss what can be observed from the results.

3. In the hope of obtaining even better results you decide to implement a different heuristic that hopefully leads to a better performance. You discover the paper of Reeves (1995) that discusses four different versions of a Genetic Algorithm (GA) to cope with the PFSP:
   - Algo1: 1-point crossover (C1) with exchange mutation (ExM)
   - Algo2: 2-point crossover (C2) with exchange mutation (ExM)
   - Algo3: 1-point crossover (C1) with shift mutation (SM)
   - Algo4: 2-point crossover (C1) with shift mutation (SM)

- All four algorithms use the same initialization procedure (one solution from NEH and remaining solutions are generated at random) and selection operator (fitness rank and uniform distribution for parental selection + solution below median for deletion) as described in the paper.

Use only the algorithm corresponding to your group number; there will be no extra marks if you implement any of the other algorithms as well. The pseudocode in the Appendix of the research paper (Reeves, 1995) gives you an idea of how the GA can be implemented. Please bear in mind that the paper may use a different notation to the one used in this brief.

- Implement the algorithm in Python following the description of the paper. Use the algorithm parameters provided by Reeves (1995) in the pseudocode in the Appendix.
- Apply the algorithm to each dataset using a maximum of $1000 \times n$ solution evaluations for each run.
- Repeat each run 30 times with a different initial random seed and collect the value of the makespan of the best solution returned by each run.
- Report in a table minimum, maximum, mean, and standard deviation of the makespan for each dataset.
- Compare with the results obtained by Random Search and NEH on its own, and discuss what can be observed from the results.

4. After analysing the above results, you realise that there are several important parameters in the above GA variants: you have the population size (P = 30),[2] crossover probability (P_c = 1.0), initial mutation probability (P_m^{init} = 0.8), and a threshold parameter (D = 0.95). You decide to examine how the performance of the heuristic changes when you change those parameters:
   - Repeat the steps in Question 3 for each value of:

$$P = \{5, 10, 20, 50, 100\}$$
$$P_c = \{0.0, 0.5, 0.7, 0.9\}$$
$$P_m^{\text{init}} = \{0.0, 0.2, 0.4, 0.6, 1.0\}$$
$$D = \{0.0, 0.5, 1.0\}$$

[2] This parameter was denoted as M in Reeves (1995).

- Use Python (not Excel!) to visualize the effect of varying the above parameters on the makespan for each of the datasets. Include the performance of Random Search and NEH in your visualization as a reference.
  Hint: You may want to use boxplots with boxes showing the final objective function values obtained by all runs for each value of the parameter (P, P_c, P_m^{init}, or D) and two further boxes for the objective function values obtained by Random Search (from Question 1) and NEH (from Question 2).
- Discuss what can be observed from the results.
- Given these plots, which value of each parameter should be used and why?

References:

M. Nawaz, E.E. Enscore Jr, and I. Ham, A heuristic algorithm for the *m*-machine, *n*-job flow-shop sequencing problem, Omega, 11(1):91-95, 1983, https://doi.org/10.1016/0305-0483(83)90088-9

C.R. Reeves, A genetic algorithm for flowshop sequencing, Computers & Operations Research, 22(1):5-13, 1995, https://doi.org/10.1016/0305-0548(93)E0014-K

Hints:

a) Implement your algorithms in Python. Read carefully the brief and the research papers. The latter contain much more information than you actually need for answering the questions above.

b) Use Python to compute statistics, generate tables and plots. Do NOT use Excel. If you program those operations in an abstract way, you should be able to generate all tables and plots by calling the same function on different input data.

c) Document your code.