A New Approach to Scheduling in Flowshops

Lancaster University Management School

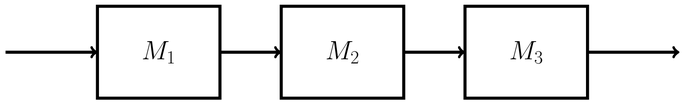
Sebastian Cáceres Gelvez [1, 2] Adam N. Letchford [1] Thu H. Dang [1]

[1]Lancaster University Management School, Lancaster, UK [2]Universidad de Santander, Cúcuta, Colombia

What is Flowshop Scheduling . . . ?

Assume a manufacturer has received an order:

- To process a number of n jobs
- Jobs are processed on m machines, arranged in a *flowline*, (i.e., a **flowshop**)
- Each machine processes one job at a time



The problem is **to find a sequence** (i.e., a *schedule*) that optimises some criteria.

The Permutation Flowshop with Makespan Criterion (PFM)

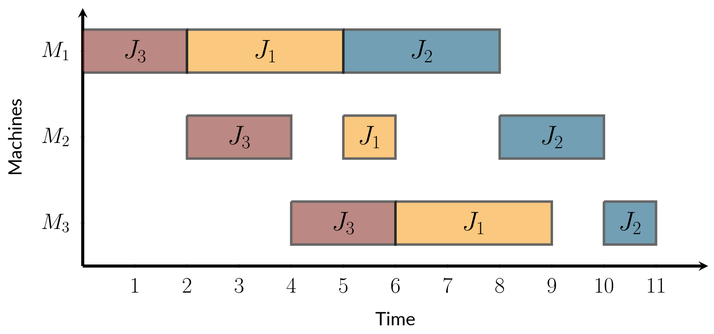
Suppose an order consists of 3 jobs: $\{J_1, J_2, J_3\}$

- A feasible solution is a *permutation* of jobs: i.e, a sequence

3	1	2

- Once defined, the sequence is the same for all the machines
- The objective is to minimise the completion time of the last job on the last machine: the **makespan**.

For $m = 3$, we can represent the sequence $\{J_3, J_1, J_2\}$ using a Gantt chart, where we can check that the *makespan* value is 11:



New Neighbourhoods for the PFM

We propose **five new neighbourhoods** for the PFM:

- Let us assume we have a feasible solution for an instance with $n = 5$:

4	1	2	5	3

1. Position Blocks: `block_size` = 2 (consecutive).

4	1	2	5	3

Jobs in `block_size` can move

2. Generalised Swap: `set_size` = 2.

4	1	2	5	3

Positions in `set_size` = $\{1, 4\}$ can move

3. Delta (Deterministic): `delta` (δ) = 2.

4	1	2	5	3

Jobs can move up to $\delta = 2$

4. Randomised Delta: `Delta` (Δ) = `random` $(1, 2 * \delta - 1)$.

4	1	2	5	3

Jobs can move up to Δ (different for each job)

5. Extended: `set_size` = 2.

4	1	2	5	3

Jobs in `set_size` = $\{2, 4\}$ can move freely; other jobs can move up to δ

Acknowledgements

The first author gratefully acknowledges funding from the Engineering and Physical Sciences Research Council (EPSRC) through the grant EP/V520214/1, and from the Ministerio de Ciencia, Tecnología e Innovación of Colombia (MINCIENCIAS) through the call "885 de 2020—Doctorados en el Exterior".

A *Matheuristic* Algorithm

1. **Initialise sequence**: obtained using the NEH algorithm [1]
2. **Build a MIP model**: proposed by Stafford *et al.* [2]
3. **Perform search**: In a loop, we apply two approaches:
   - **"Shift" local search**, as proposed in Taillard [4], and
   - **Any of the proposed neighbourhoods** by solving reduced MIPs using `IBM CPLEX` and Python 3.10.

Some Results So Far . . .

Table 1. Computational results by neighbourhood operator for Taillard instances ($m = 5$) [3]

Neighbourhood	Avg. %gap	Avg. time(s)
Position block	0.71	3.17
Generalised swap	0.60	3.84
Delta	0.17	11.45
Randomised delta	0.08	50.66
Extended	0.12	41.99

Table 2. Computational results by neighbourhood operator for Taillard instances ($m = 10$) [3]

Neighbourhood	Avg. %gap	Avg. time(s)
Delta	1.67	388.19
Randomised delta	1.34	2343.78
Extended	1.28	1012.18

We are currently working on the bigger instances with $m = 20$!

References

[1] M. Nawaz, E. Enscore, and I. Ham. "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem". In: *Omega* 11 (1983), pp. 91–95.

[2] E.F. Stafford, F.T. Tseng, and J.N. Gupta. "Comparative evaluation of MILP flowshop models". In: *J. Oper. Res. Soc.* 56 (2005), pp. 88–101.

[3] E. Taillard. "Benchmarks for basic scheduling problems". In: *Eur. J. Oper. Res.* 64 (1993), pp. 278–285.

[4] E. Taillard. "Some efficient heuristic methods for the flow shop sequencing problem". In: *Eur. J. Oper. Res* 47 (1990), pp. 65–74.

PhD in Management Science: s.caceresgelvez@lancaster.ac.uk