Solving the Non-permutation Flow Shop Scheduling Problem

Alexander J. Benavides

ajbenavides@unsa.edu.pe
ajbenavides@ucsp.edu.pe

October, 2019

Today we'll see...

How to solve very-difficult combinatorial-optimization problems by using computers to model problems and produce solutions.

Case Study: Flow Shop Scheduling Problem (FSSP)

Methods: constructive heuristics, local search, meta-heuristics, ...

Thinking out of the box!!! ...

Benavides A.J., & Ritt M., (2016), Two simple and effective heuristics for minimizing the makespan in non-permutation flowshop scheduling problems. Comput. Oper. Res. 60, 160–169.

Benavides A.J., & Ritt M., (2018), Fast heuristics for minimizing the makespan in non-permutation flow shops. Comput. Oper. Res. 100, 230–243.

Outline

Flow Shop Scheduling Problem (FSSP)

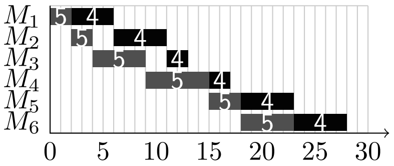
6×6 instance of the FSSP

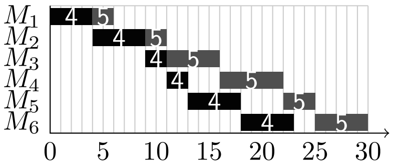
Jobs	Operations					
	M_1	M_2	M_3	M_4	M_5	M_6
J_1	3	6	3	3	4	3
J_2	4	3	5	3	5	2
J_3	6	5	2	2	2	4
J_4	4	5	2	2	5	5
J_5	2	2	5	6	3	5
J_6	2	3	5	5	3	3

A set of jobs $J_1, \ldots, J_n$ must be processed on a set of machines $M_1, \ldots, M_m$

with given processing times p_{ij} for each job J_j on machine M_i

Objective function:
min. $C_{\max} = \max C_j$ (makespan)
There are $n!$ possible solutions





Flow Shop Scheduling Problem (FSSP)

6! 720
10! 3628800
20! 2.43e+18
50! 3.04e+64
100! 9.33e+157
200! 7.88e+374
500! 1.22e+1134
800! 7.71e+1976
Grains of sand on Earth 7.5e+18
Stars in the observable universe 2e+20
4 GHz = 4e+9 op/s = 4 op/s
60 s * 60 m * 24 h * 365 d = 31536000
operations per year: 1.26144e+17
so 2.4e+18/1.2e+17 20 years.

Taillard (1993): 120 instances
$n \in 20, 50, 100, 200, 500$ jobs by
$m \in 5, 10, 20$ machines.

Vallada, Ruiz, Framinan (2015)
240 small instances
$n \in 10, 20, 30, 40, 50, 60$ jobs by
$m \in 5, 10, 15, 20$ machines.

240 large instances
$n \in 100, 200, 300, 400, 500, 600, 700, 800$ jobs by
$m \in 20, 40, 60$ machines.

Flow Shop Scheduling Problem (FSSP)

6! 720
10! 3628800
20! 2.43e+18
50! 3.04e+64
100! 9.33e+157
200! 7.88e+374
500! 1.22e+1134
800! 7.71e+1976
Grains of sand on Earth 7.5e+18
Stars in the observable universe 2e+20
4 GHz = 4e+9 op/s = 4 op/s
60 s * 60 m * 24 h * 365 d = 31536000
operations per year: 1.26144e+17
so 2.4e+18/1.2e+17 20 years.

Taillard (1993): 120 instances
$n \in 20, 50, 100, 200, 500$ jobs by
$m \in 5, 10, 20$ machines.

Vallada, Ruiz, Framinan (2015)
240 small instances
$n \in 10, 20, 30, 40, 50, 60$ jobs by
$m \in 5, 10, 15, 20$ machines.

240 large instances
$n \in 100, 200, 300, 400, 500, 600, 700, 800$ jobs by
$m \in 20, 40, 60$ machines.

Flow Shop Scheduling Problem (FSSP)

6! 720
10! 3628800
20! 2.43e+18
50! 3.04e+64
100! 9.33e+157
200! 7.88e+374
500! 1.22e+1134
800! 7.71e+1976
Grains of sand on Earth 7.5e+18
Stars in the observable universe 2e+20
4 GHz = 4e+9 op/s = 4 op/s
60 s * 60 m * 24 h * 365 d = 31536000
operations per year: 1.26144e+17
so 2.4e+18/1.2e+17 20 years.

Taillard (1993): 120 instances
$n \in 20, 50, 100, 200, 500$ jobs by
$m \in 5, 10, 20$ machines.

Vallada, Ruiz, Framinan (2015)
240 small instances
$n \in 10, 20, 30, 40, 50, 60$ jobs by
$m \in 5, 10, 15, 20$ machines.

240 large instances
$n \in 100, 200, 300, 400, 500, 600, 700, 800$ jobs by
$m \in 20, 40, 60$ machines.

Flow Shop Scheduling Problem (FSSP)

6! 720
10! 3628800
20! 2.43e+18
50! 3.04e+64
100! 9.33e+157
200! 7.88e+374
500! 1.22e+1134
800! 7.71e+1976
Grains of sand on Earth 7.5e+18
Stars in the observable universe 2e+20
4 GHz = 4e+9 op/s = 4 op/s
60 s * 60 m * 24 h * 365 d = 31536000
operations per year: 1.26144e+17
so 2.4e+18/1.2e+17   20 years.

Taillard (1993): 120 instances
$n \in 20, 50, 100, 200, 500$ jobs by
$m \in 5, 10, 20$ machines.

Vallada, Ruiz, Framinan (2015)
240 small instances
$n \in 10, 20, 30, 40, 50, 60$ jobs by
$m \in 5, 10, 15, 20$ machines.

240 large instances
$n \in 100, 200, 300, 400, 500, 600, 700, 800$ jobs by
$m \in 20, 40, 60$ machines.

Flow Shop Scheduling Problem (FSSP)

6! 720
10! 3628800
20! 2.43e+18
50! 3.04e+64
100! 9.33e+157
200! 7.88e+374
500! 1.22e+1134
800! 7.71e+1976
Grains of sand on Earth 7.5e+18
Stars in the observable universe 2e+20
4 GHz = 4e+9 op/s = 4 op/s
60 s * 60 m * 24 h * 365 d = 31536000
operations per year: 1.26144e+17
so 2.4e+18/1.2e+17   20 years.

Taillard (1993): 120 instances
$n \in 20, 50, 100, 200, 500$ jobs by
$m \in 5, 10, 20$ machines.

Vallada, Ruiz, Framinan (2015)
240 small instances
$n \in 10, 20, 30, 40, 50, 60$ jobs by
$m \in 5, 10, 15, 20$ machines.

240 large instances
$n \in 100, 200, 300, 400, 500, 600, 700, 800$ jobs by
$m \in 20, 40, 60$ machines.

Nawaz, Enscore & Ham (1983) NEH constructive heuristic

6×6 instance of the FSSP

Jobs	Operations						
	M_1	M_2	M_3	M_4	M_5	M_6	Total
J_1	3	6	3	3	4	3	22
J_2	4	3	5	3	5	2	22
J_3	6	5	2	2	2	4	21
J_4	4	5	2	2	5	5	23
J_5	2	2	5	6	3	5	23
J_6	2	3	5	5	3	3	21

First, determine insertion order:

$\pi_o = (J_4, J_5, J_1, J_2, J_3, J_6)$

The, insert one by one

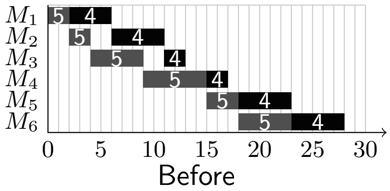
at the best position

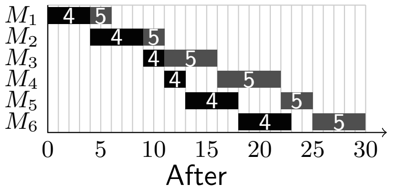
starting with $\pi = (J_4)$

```
1: function NEH_Constructive_Heuristic( )
2:     $\pi_o := (\pi_o(1), \dots, \pi_o(n))$ from large to small
3:     $\pi := (\pi_o(1))$
4:     for $\pi_o(j), j \in [2, n]$ do
5:         evaluate all the insertion positions of job $\pi_o(j)$ into $\pi$
6:         insert job $\pi_o(j)$ into $\pi$ at the position which minimizes $C_{\max}$
7:     end for
8:     return $\pi$
9: end function
```

Nawaz, Enscore & Ham (1983) NEH constructive heuristic

$\pi_o = (J_4, J_5, J_1, J_2, J_3, J_6)$ $\pi = (J_4)$ Next job: J_5



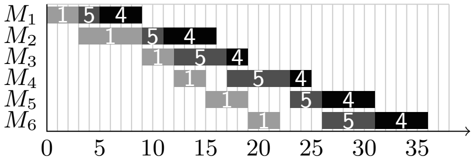


```
1: function NEH_Constructive_Heuristic( )
2:     $\pi_o := (\pi_o(1), \dots, \pi_o(n))$ from large to small
3:     $\pi := (\pi_o(1))$
4:     for $\pi_o(j), j \in [2, n]$ do
5:         evaluate all the insertion positions of job $\pi_o(j)$ into $\pi$
6:         insert job $\pi_o(j)$ into $\pi$ at the position which minimizes $C_{\max}$
7:     end for
8:     return $\pi$
9: end function
```

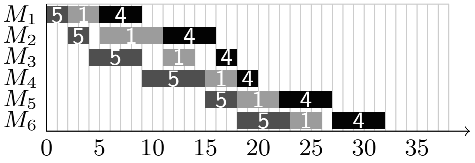
Nawaz, Enscore & Ham (1983) NEH constructive heuristic

$\pi_o = (J_4, J_5, J_1, J_2, J_3, J_6)$ $\pi = (J_5, J_4)$ Next job: J_1

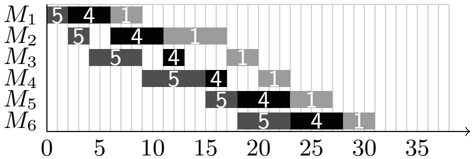
Before



Middle

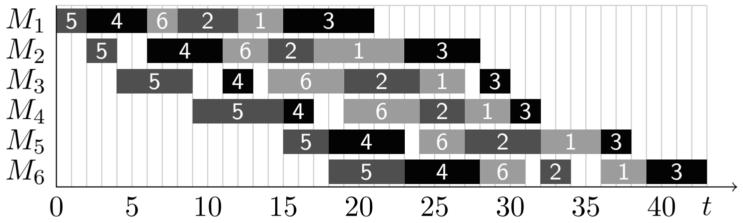


After



Nawaz, Enscore & Ham (1983) NEH constructive heuristic

And so on … until all jobs are inserted: $\pi = (J_5, J_4, J_6, J_2, J_1, J_3)$



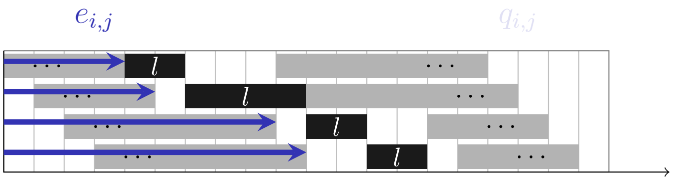
Original NEH has a time complexity of $O(n^3m)$

NEH inserts n jobs, evaluates $O(n)$ insertion positions, (exactly $n(n+1)/2 - 1$ evaluations) and each evaluation has a time complexity of $O(nm)$

Nawaz, Enscore & Ham (1983) NEH$_T$ heuristic with Taillard (1990) acceleration technique for $C_{\max}$

Earliest completion times $e_{i,j}$ before insertion position remain unchanged

Also $q_{i,j}$ times after insertion position remain unchanged



Taillard defines:

$e_{i,j} = \max\{e_{i,j-1}, e_{i-1,j}\} + p_{i,\pi(j)}$, for $i \in [m], j \in [|\pi|]$, with $e_{0,j} = 0$ and $e_{i,0} = 0$

$q_{i,j} = \max\{q_{i,j+1}, q_{i+1,j}\} + p_{i,\pi(j)}$, for $i \in [m], j \in [|\pi|]$, with $q_{m+1,j} = 0$ and $q_{i,k+1} = 0$

$f_{i,j} = \max\{f_{i-1,j}, e_{i,j-1}\} + p_{i,l}$, for $i \in [m], j \in [|\pi| + 1]$, with $f_{0,j} = 0$

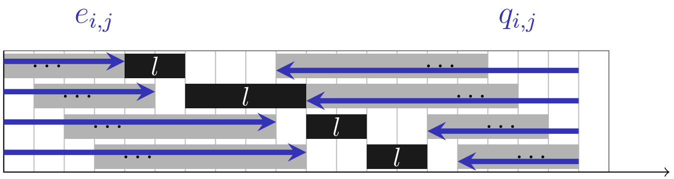
$M_j = \max_{i \in [m]}\{f_{i,j} + q_{i,j}\}$, for $j \in [|\pi| + 1]$

These calculations evaluate n insertion positions in time $O(nm)$

This reduces the time complexity of NEH$_T$ from $O(n^3 m)$ to $O(n^2 m)$

Nawaz, Enscore & Ham (1983) NEH$_T$ heuristic with Taillard (1990) acceleration technique for $C_{\max}$

Earliest completion times $e_{i,j}$ before insertion position remain unchanged
Also $q_{i,j}$ times after insertion position remain unchanged



Taillard defines:

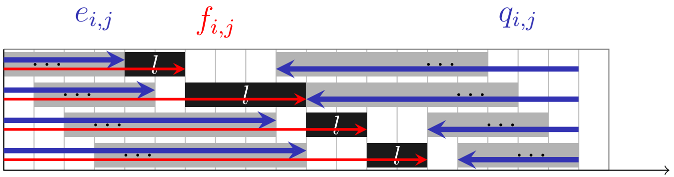
$e_{i,j} = \max\{e_{i,j-1}, e_{i-1,j}\} + p_{i,\pi(j)}$, for $i \in [m], j \in [|\pi|]$, with $e_{0,j} = 0$ and $e_{i,0} = 0$
$q_{i,j} = \max\{q_{i,j+1}, q_{i+1,j}\} + p_{i,\pi(j)}$, for $i \in [m], j \in [|\pi|]$, with $q_{m+1,j} = 0$ and $q_{i,k+1} = 0$
$f_{i,j} = \max\{f_{i-1,j}, e_{i,j-1}\} + p_{i,l}$, for $i \in [m], j \in [|\pi|+1]$, with $f_{0,j} = 0$
$M_j = \max_{i\in[m]}\{f_{i,j} + q_{i,j}\}$, for $j \in [|\pi|+1]$

These calculations evaluate n insertion positions in time $O(nm)$

This reduces the time complexity of NEH$_T$ from $O(n^3m)$ to $O(n^2m)$

Nawaz, Enscore & Ham (1983) NEH$_T$ heuristic with Taillard (1990) acceleration technique for $C_{\max}$

Earliest completion times $e_{i,j}$ before insertion position remain unchanged
Also $q_{i,j}$ times after insertion position remain unchanged



Taillard defines:

$e_{i,j} = \max\{e_{i,j-1}, e_{i-1,j}\} + p_{i,\pi(j)}$, for $i \in [m], j \in [|\pi|]$, with $e_{0,j} = 0$ and $e_{i,0} = 0$
$q_{i,j} = \max\{q_{i,j+1}, q_{i+1,j}\} + p_{i,\pi(j)}$, for $i \in [m], j \in [|\pi|]$, with $q_{m+1,j} = 0$ and $q_{i,k+1} = 0$
$f_{i,j} = \max\{f_{i-1,j}, e_{i,j-1}\} + p_{i,l}$, for $i \in [m], j \in [|\pi| + 1]$, with $f_{0,j} = 0$
$M_j = \max_{i\in[m]}\{f_{i,j} + q_{i,j}\}$, for $j \in [|\pi| + 1]$

These calculations evaluate n insertion positions in time $O(nm)$

This reduces the time complexity of NEH$_T$ from $O(n^3m)$ to $O(n^2m)$

Neighborhoods for local search for permutation schedules

Swapping adjacent jobs ($n - 1$ neighbors)

Swapping arbitrary pairs of jobs ($\binom{n}{2}$ neighbors)

Reinserting a job into another position ($(n-1)^2$ neighbors)

Neighborhoods for local search for permutation schedules

Swapping adjacent jobs ($n-1$ neighbors)

$$\begin{tikzcd}[column sep=small] \pi = (& J_1, & J_2, & J_3, & J_4, \arrow[r, bend left, leftrightarrow] & J_5, & J_6 &) \end{tikzcd}$$

Swapping arbitrary pairs of jobs ($\binom{n}{2}$ neighbors)

Reinserting a job into another position ($(n-1)^2$ neighbors)

Neighborhoods for local search for permutation schedules

Swapping adjacent jobs ($n-1$ neighbors)

$$\begin{tikzcd}[column sep=small] \pi = (& J_1, & J_2, \arrow[r, bend left, leftrightarrow] & J_3, & J_4, & J_5, & J_6 &) \end{tikzcd}$$

Swapping arbitrary pairs of jobs ($\binom{n}{2}$ neighbors)

Reinserting a job into another position ($(n-1)^2$ neighbors)

Neighborhoods for local search for permutation schedules

Swapping adjacent jobs ($n-1$ neighbors)

$$\begin{tikzcd}[column sep=small] \pi = (& J_1, \arrow[r, bend left, leftrightarrow] & J_2, \arrow[r, bend left, leftrightarrow] & J_3, \arrow[r, bend left, leftrightarrow] & J_4, \arrow[r, bend left, leftrightarrow] & J_5, \arrow[r, bend left, leftrightarrow] & J_6 &) \end{tikzcd}$$

Swapping arbitrary pairs of jobs ($\binom{n}{2}$ neighbors)

Reinserting a job into another position ($(n-1)^2$ neighbors)

Neighborhoods for local search for permutation schedules

Swapping adjacent jobs ($n-1$ neighbors)

$$\begin{tikzcd}[column sep=small]
\pi = (& J_1, \arrow[r, bend left, leftrightarrow] & J_2, \arrow[r, bend left, leftrightarrow] & J_3, \arrow[r, bend left, leftrightarrow] & J_4, \arrow[r, bend left, leftrightarrow] & J_5, \arrow[r, bend left, leftrightarrow] & J_6 &)
\end{tikzcd}$$

Swapping arbitrary pairs of jobs ($\binom{n}{2}$ neighbors)

$$\begin{tikzcd}[column sep=small]
\pi = (& J_1, & J_2, & J_3, \arrow[rr, bend left, leftrightarrow] & J_4, & J_5, & J_6 &)
\end{tikzcd}$$

Reinserting a job into another position ($(n-1)^2$ neighbors)

Neighborhoods for local search for permutation schedules

Swapping adjacent jobs ($n-1$ neighbors)

$$\begin{tikzcd}[column sep=small]
\pi = (& J_1, \arrow[r, bend left, leftrightarrow] & J_2, \arrow[r, bend left, leftrightarrow] & J_3, \arrow[r, bend left, leftrightarrow] & J_4, \arrow[r, bend left, leftrightarrow] & J_5, \arrow[r, bend left, leftrightarrow] & J_6 &)
\end{tikzcd}$$

Swapping arbitrary pairs of jobs ($\binom{n}{2}$ neighbors)

$$\begin{tikzcd}[column sep=small]
\pi = (& J_1, & J_2, \arrow[rrrr, bend left, leftrightarrow] & J_3, & J_4, & J_5, & J_6 &)
\end{tikzcd}$$

Reinserting a job into another position ($(n-1)^2$ neighbors)

Neighborhoods for local search for permutation schedules

Swapping adjacent jobs ($n-1$ neighbors)

$$\begin{tikzcd}
\pi = (& J_1, \arrow[r, bend left, <->] & J_2, \arrow[r, bend left, <->] & J_3, \arrow[r, bend left, <->] & J_4, \arrow[r, bend left, <->] & J_5, \arrow[r, bend left, <->] & J_6 &)
\end{tikzcd}$$

Swapping arbitrary pairs of jobs ($\binom{n}{2}$ neighbors)

$$\begin{tikzcd}
\pi = (& J_1, \arrow[r, bend left, <->] \arrow[rr, bend left, <->] \arrow[rrr, bend left, <->] \arrow[rrrr, bend left, <->] \arrow[rrrrr, bend left, <->] & J_2, \arrow[r, bend left, <->] \arrow[rr, bend left, <->] \arrow[rrr, bend left, <->] \arrow[rrrr, bend left, <->] & J_3, \arrow[r, bend left, <->] \arrow[rr, bend left, <->] \arrow[rrr, bend left, <->] & J_4, \arrow[r, bend left, <->] \arrow[rr, bend left, <->] & J_5, \arrow[r, bend left, <->] & J_6 &)
\end{tikzcd}$$

Reinserting a job into another position ($(n-1)^2$ neighbors)

Neighborhoods for local search for permutation schedules

Swapping adjacent jobs ($n - 1$ neighbors)

$$\begin{tikzcd}
\pi = (& J_1, \arrow[r, bend left, leftrightarrow] & J_2, \arrow[r, bend left, leftrightarrow] & J_3, \arrow[r, bend left, leftrightarrow] & J_4, \arrow[r, bend left, leftrightarrow] & J_5, \arrow[r, bend left, leftrightarrow] & J_6 &)
\end{tikzcd}$$

Swapping arbitrary pairs of jobs ($\binom{n}{2}$ neighbors)

$$\begin{tikzcd}
\pi = (& J_1, \arrow[r, bend left, leftrightarrow] \arrow[rr, bend left, leftrightarrow] \arrow[rrr, bend left, leftrightarrow] \arrow[rrrr, bend left, leftrightarrow] \arrow[rrrrr, bend left, leftrightarrow] & J_2, \arrow[r, bend left, leftrightarrow] \arrow[rr, bend left, leftrightarrow] \arrow[rrr, bend left, leftrightarrow] \arrow[rrrr, bend left, leftrightarrow] & J_3, \arrow[r, bend left, leftrightarrow] \arrow[rr, bend left, leftrightarrow] \arrow[rrr, bend left, leftrightarrow] & J_4, \arrow[r, bend left, leftrightarrow] \arrow[rr, bend left, leftrightarrow] & J_5, \arrow[r, bend left, leftrightarrow] & J_6 &)
\end{tikzcd}$$

Reinserting a job into another position ($(n-1)^2$ neighbors)

$$\begin{tikzcd}
\pi = (& {} & J_1, & {} & J_2, & {} & J_3, & {} & J_4, & {} & J_5, \arrow[lllllllll, bend right] \arrow[lllllll, bend right] \arrow[lllll, bend right] \arrow[lll, bend right] \arrow[rrr, bend left] & {} & J_6 & {} &)
\end{tikzcd}$$

Neighborhoods for local search for permutation schedules

Swapping adjacent jobs ($n-1$ neighbors)

$$\begin{tikzcd}
\pi = (& J_1, \arrow[r, leftrightarrow, bend left] & J_2, \arrow[r, leftrightarrow, bend left] & J_3, \arrow[r, leftrightarrow, bend left] & J_4, \arrow[r, leftrightarrow, bend left] & J_5, \arrow[r, leftrightarrow, bend left] & J_6 &)
\end{tikzcd}$$

Swapping arbitrary pairs of jobs ($\binom{n}{2}$ neighbors)

$$\begin{tikzcd}
\pi = (& J_1, \arrow[r, leftrightarrow, bend left] \arrow[rr, leftrightarrow, bend left] \arrow[rrr, leftrightarrow, bend left] \arrow[rrrr, leftrightarrow, bend left] \arrow[rrrrr, leftrightarrow, bend left] & J_2, \arrow[r, leftrightarrow, bend left] \arrow[rr, leftrightarrow, bend left] \arrow[rrr, leftrightarrow, bend left] \arrow[rrrr, leftrightarrow, bend left] & J_3, \arrow[r, leftrightarrow, bend left] \arrow[rr, leftrightarrow, bend left] \arrow[rrr, leftrightarrow, bend left] & J_4, \arrow[r, leftrightarrow, bend left] \arrow[rr, leftrightarrow, bend left] & J_5, \arrow[r, leftrightarrow, bend left] & J_6 &)
\end{tikzcd}$$

Reinserting a job into another position ($(n-1)^2$ neighbors)

$$\begin{tikzcd}
\pi = (& J_1, & J_2, & J_3, \arrow[ll, bend right] \arrow[l, bend right] \arrow[r, bend left] \arrow[rr, bend left] \arrow[rrr, bend left] & J_4, & J_5, & J_6 &)
\end{tikzcd}$$

Neighborhoods for local search for permutation schedules

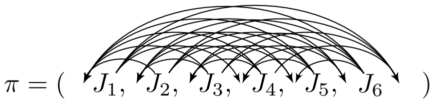
Swapping adjacent jobs ($n-1$ neighbors)

$$\begin{tikzcd}
\pi = (& J_1, \arrow[r, bend left, leftrightarrow] & J_2, \arrow[r, bend left, leftrightarrow] & J_3, \arrow[r, bend left, leftrightarrow] & J_4, \arrow[r, bend left, leftrightarrow] & J_5, \arrow[r, bend left, leftrightarrow] & J_6 &)
\end{tikzcd}$$

Swapping arbitrary pairs of jobs ($\binom{n}{2}$ neighbors)

$$\begin{tikzcd}
\pi = (& J_1, \arrow[r, bend left, leftrightarrow] \arrow[rr, bend left, leftrightarrow] \arrow[rrr, bend left, leftrightarrow] \arrow[rrrr, bend left, leftrightarrow] \arrow[rrrrr, bend left, leftrightarrow] & J_2, \arrow[r, bend left, leftrightarrow] \arrow[rr, bend left, leftrightarrow] \arrow[rrr, bend left, leftrightarrow] \arrow[rrrr, bend left, leftrightarrow] & J_3, \arrow[r, bend left, leftrightarrow] \arrow[rr, bend left, leftrightarrow] \arrow[rrr, bend left, leftrightarrow] & J_4, \arrow[r, bend left, leftrightarrow] \arrow[rr, bend left, leftrightarrow] & J_5, \arrow[r, bend left, leftrightarrow] & J_6 &)
\end{tikzcd}$$

Reinserting a job into another position ($(n-1)^2$ neighbors)


$\pi = (\; J_1, \; J_2, \; J_3, \; J_4, \; J_5, \; J_6 \;)$

Neighborhoods for local search for permutation schedules

Swapping adjacent jobs ($n-1$ neighbors)

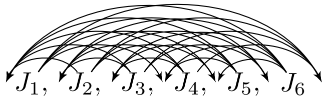
$$\begin{tikzcd}
\pi = (& J_1, \arrow[r, bend left, <->] & J_2, \arrow[r, bend left, <->] & J_3, \arrow[r, bend left, <->] & J_4, \arrow[r, bend left, <->] & J_5, \arrow[r, bend left, <->] & J_6 &)
\end{tikzcd}$$

part of:

Swapping arbitrary pairs of jobs ($\binom{n}{2}$ neighbors)

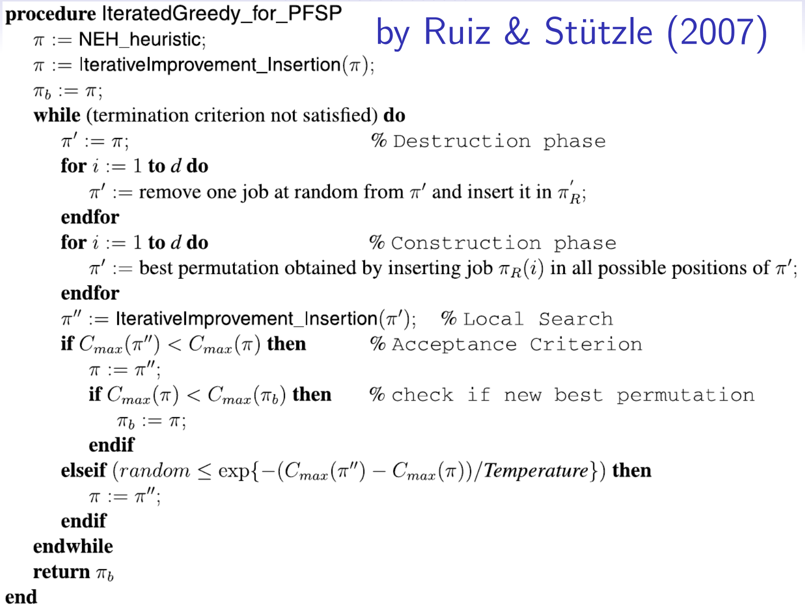
$$\begin{tikzcd}
\pi = (& J_1, \arrow[r, bend left, <->] \arrow[rr, bend left, <->] \arrow[rrr, bend left, <->] \arrow[rrrr, bend left, <->] \arrow[rrrrr, bend left, <->] & J_2, \arrow[r, bend left, <->] \arrow[rr, bend left, <->] \arrow[rrr, bend left, <->] \arrow[rrrr, bend left, <->] & J_3, \arrow[r, bend left, <->] \arrow[rr, bend left, <->] \arrow[rrr, bend left, <->] & J_4, \arrow[r, bend left, <->] \arrow[rr, bend left, <->] & J_5, \arrow[r, bend left, <->] & J_6 &)
\end{tikzcd}$$

Reinserting a job into another position ($(n-1)^2$ neighbors)



$\pi = (\quad J_1, \; J_2, \; J_3, \; J_4, \; J_5, \; J_6 \quad)$

Taillard acc. $O(n^2m)$



by Ruiz & Stützle (2007)

```
procedure IteratedGreedy_for_PFSP
    $\pi$ := NEH_heuristic;
    $\pi$ := IterativeImprovement_Insertion($\pi$);
    $\pi_b := \pi$;
    while (termination criterion not satisfied) do
        $\pi' := \pi$;                                   % Destruction phase
        for $i := 1$ to $d$ do
            $\pi'$ := remove one job at random from $\pi'$ and insert it in $\pi'_R$;
        endfor
        for $i := 1$ to $d$ do                           % Construction phase
            $\pi'$ := best permutation obtained by inserting job $\pi_R(i)$ in all possible positions of $\pi'$;
        endfor
        $\pi''$ := IterativeImprovement_Insertion($\pi'$);   % Local Search
        if $C_{max}(\pi'') < C_{max}(\pi)$ then          % Acceptance Criterion
            $\pi := \pi''$;
            if $C_{max}(\pi) < C_{max}(\pi_b)$ then      % check if new best permutation
                $\pi_b := \pi$;
            endif
        elseif ($random \leq \exp\{-(C_{max}(\pi'') - C_{max}(\pi))/Temperature\}$) then
            $\pi := \pi''$;
        endif
    endwhile
    return $\pi_b$
end
```

Outline

Permutation FSSP vs. Non-permutation FSSP

Practically are the same problem!

All machines have the same processing order

Simplified problem:

- Possible solutions: $n!$ disregarding the number of machines
- 99% of the literature

Excludes better (optimal) non-permutation schedules

Some machines may have different processing orders

Harder problem

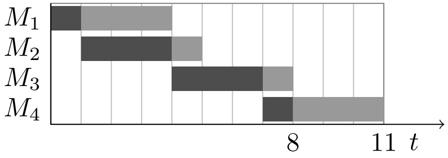
- Possible solutions:
  $n!^{(m-2)}$ for **min.** $C_{\max}$
  $n!^{(m-1)}$ for **min.** C_{sum}
- 1% of the literature

How to solve this harder problem with the same effort?

Permutation FSSP vs. Non-permutation FSSP

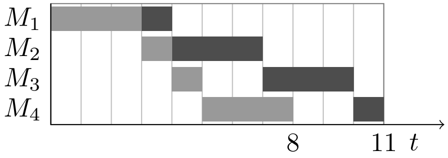
Permutation

(J_1, J_2)



$C_{\max} = 11$

(J_2, J_1)



$C_{\max} = 11$

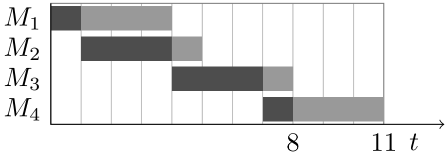
FSSP 2×4 instance.

Jobs	Operations			
	M_1	M_2	M_3	M_4
J_1	1	3	3	1
J_2	3	1	1	3

Permutation FSSP vs. Non-permutation FSSP

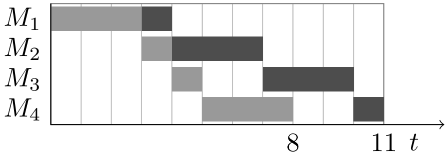
Permutation

(J_1, J_2)



$C_{\max} = 11$

(J_2, J_1)

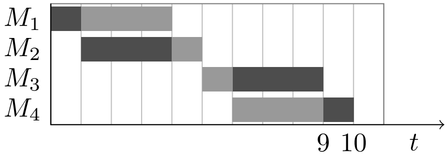


$C_{\max} = 11$

Non-permutation

(J_1, J_2)

(J_2, J_1)

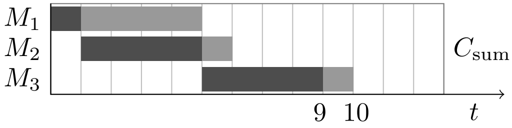


$C_{\max} = 10$

Permutation FSSP vs. Non-permutation FSSP

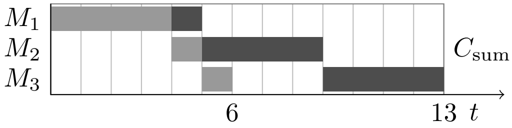
Permutation

(J_1, J_2)



$C_{\text{sum}} = 19$

(J_2, J_1)

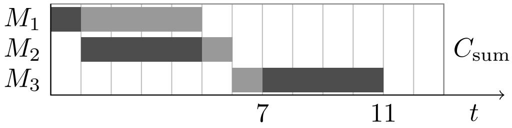


$C_{\text{sum}} = 19$

Non-permutation

(J_1, J_2)

(J_2, J_1)



$C_{\text{sum}} = 18$

Permutation FSSP vs. Non-permutation FSSP

Practically are the same problem!

All machines have the same processing order

Simplified problem:

- Possible solutions: $n!$ disregarding the number of machines
- 99% of the literature

Excludes better (optimal) non-permutation schedules

Some machines may have different processing orders

Harder problem

- Possible solutions:
  $n!^{(m-2)}$ for **min.** C_{max}
  $n!^{(m-1)}$ for **min.** C_{sum}
- 1% of the literature

How to solve this harder problem with the same effort?

Permutation FSSP vs. Non-permutation FSSP

Practically are the same problem!

All machines have the same processing order

Some machines may have different processing orders

Simplified problem:

- Possible solutions: $n!$ disregarding the number of machines
- 99% of the literature

Harder problem

- Possible solutions:
  $n!^{(m-2)}$ for **min.** $C_{\max}$
  $n!^{(m-1)}$ for **min.** C_{sum}
- 1% of the literature

Excludes better (optimal) non-permutation schedules

How to solve this harder problem with the same effort?

Permutation FSSP vs. Non-permutation FSSP

Practically are the same problem!

All machines have the same processing order

Simplified problem:

- Possible solutions: $n!$ disregarding the number of machines
- 99% of the literature

Excludes better (optimal) non-permutation schedules

Some machines may have different processing orders

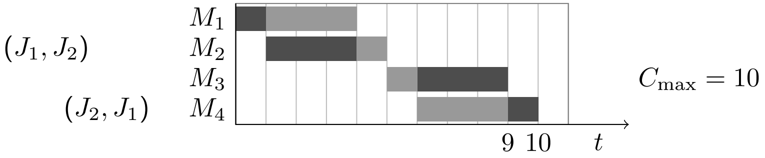
Harder problem

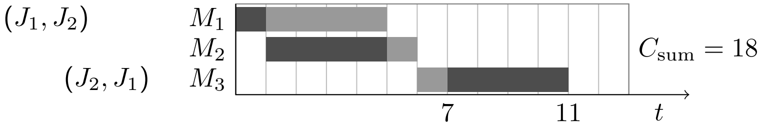
- Possible solutions:
  $n!^{(m-2)}$ for **min.** $C_{\max}$
  $n!^{(m-1)}$ for **min.** C_{sum}
- 1% of the literature

How to solve this harder problem with the same effort?

Job insertion for non-permutation FSSP

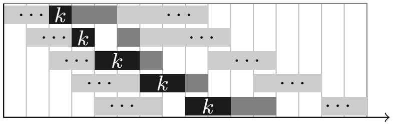
Optimal schedules have small differences in the processing order of subsequent machines.

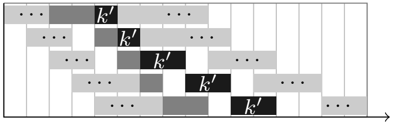




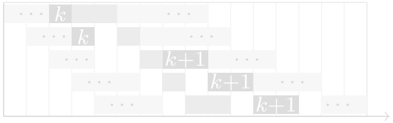
Job insertion for non-permutation FSSP
with anticipation and delay after an intermediate machine

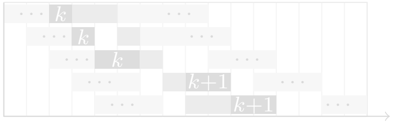
Original NEH inserts jobs only into straight positions



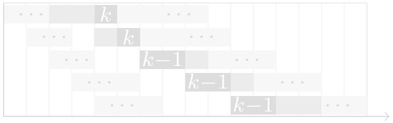


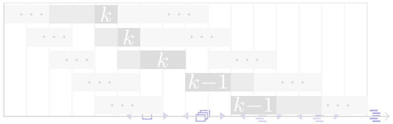
We also insert jobs with delay after an intermediate machine





and with anticipation after an intermediate machine

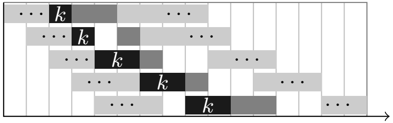


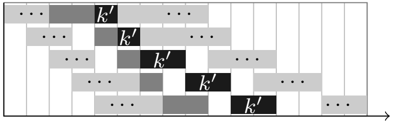


Job insertion for non-permutation FSSP

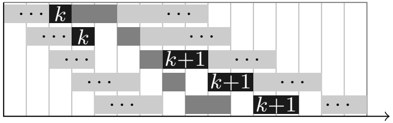
with anticipation and delay after an intermediate machine

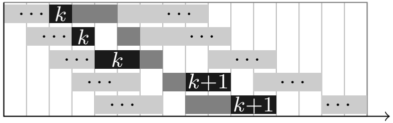
Original NEH inserts jobs only into straight positions



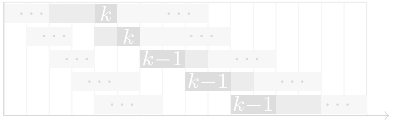


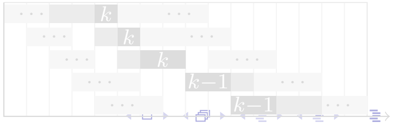
We also insert jobs with delay after an intermediate machine





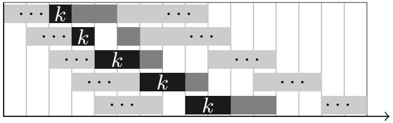
and with anticipation after an intermediate machine

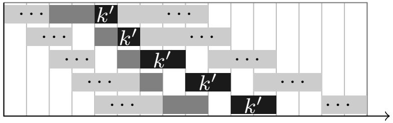




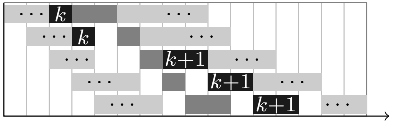
Job insertion for non-permutation FSSP
with anticipation and delay after an intermediate machine

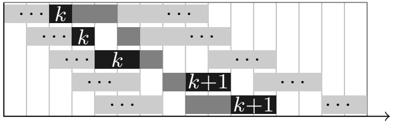
Original NEH inserts jobs only into straight positions



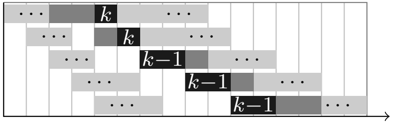


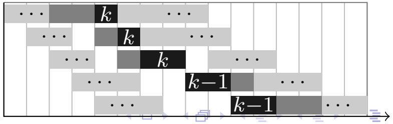
We also insert jobs with delay after an intermediate machine





and with anticipation after an intermediate machine





Job insertion for non-permutation FSSP

NEH-like heuristics for non-permutation FSSP

```
1: function NEH_like_Constructive_Heuristic( )
2:     $\pi_o := (\pi_o(1), \ldots, \pi_o(n))$ from large to small
3:     $\pi := (\pi_o(1))$
4:     for $\pi_o(j), j \in [2, n]$ do
5:         for all insertion positions $k \in [j]$ do
6:             evaluate insertion of $\pi_o(j)$ at $k$ with anticipation after $M_i$ with $i \in [2, m-2]$
7:             evaluate insertion of $\pi_o(j)$ at $k$ with delay after $M_i$ with $i \in [2, m-2]$
8:             evaluate insertion of $J_j$ at $k$ straight
9:         end for
10:        Apply the best insertion of job $\rho_o(j)$ into $\pi$ which minimizes $C_{\max}$
11:    end for
12:    return $\pi$
13: end function
```

The number of insertion possibilities goes from $O(n)$ to $O(nm)$

Inserts n jobs in time $O(n^3m^2)$ for Csum (cannot use Taillard acceleration)

Job insertion for non-permutation FSSP

NEH-like heuristics for non-permutation FSSP

```
1: function NEH_like_Constructive_Heuristic( )
2:     $\pi_o := (\pi_o(1), \ldots, \pi_o(n))$ from large to small
3:     $\pi := (\pi_o(1))$
4:     for $\pi_o(j), j \in [2, n]$ do
5:         for all insertion positions $k \in [j]$ do
6:             evaluate insertion of $\pi_o(j)$ at $k$ with anticipation after $M_i$ with $i \in [2, m-2]$
7:             evaluate insertion of $\pi_o(j)$ at $k$ with delay after $M_i$ with $i \in [2, m-2]$
8:             evaluate insertion of $J_j$ at $k$ straight
9:         end for
10:        Apply the best insertion of job $\rho_o(j)$ into $\pi$ which minimizes $C_{\max}$
11:    end for
12:    return $\pi$
13: end function
```

The number of insertion possibilities goes from $O(n)$ to $O(nm)$

Inserts n jobs in time $O(n^3m^2)$ for Csum (cannot use Taillard acceleration)

Job insertion for non-permutation FSSP

Non-permutation insertions with Taillard acceleration

Taillard acceleration technique needs adjustments because...

Non-permutation insertions produces invalid $e_{i,j}$ and $q_{i,j}$ when used with m-permutation representation, e.g.:

$$\begin{array}{l} \pi_1 = (4,\ 3) \\ \pi_2 = (4,\ 3) \\ \pi_3 = (4,\ 3) \\ \pi_4 = (3,\ 4) \\ \pi_5 = (3,\ 4) \end{array} \Rightarrow \begin{array}{l} \pi_1 = (4,\ 3,\ 1) \\ \pi_2 = (4,\ 3,\ 1) \\ \pi_3 = (4,\ 1,\ 3) \\ \pi_4 = (3,\ 1,\ 4) \\ \pi_5 = (3,\ 1,\ 4) \end{array} \Rightarrow \begin{array}{l} \pi_1 = (4,\ 3,\ 2,|1) \\ \pi_2 = (4,\ 3,\ 2,|1) \\ \pi_3 = (4,\ 1,\ 3,|2) \\ \pi_4 = (3,\ 1,\ 4,|2) \\ \pi_5 = (3,\ 1,\ 4,|2) \end{array}$$

Two possible alternative solutions:

Update invalid $e_{i,j}$ and $q_{i,j}$ efficiently

NFS constructive heuristic $O(n^2m^2W)$ (Benavides & Ritt, 2016)

Propose a new representation that supports Taillard acceleration

NEH$_{\text{BR}}$ constructive heuristic $O(n^2m)$ (same as NEH$_T$, Benavides & Ritt, 2018)

Job insertion for non-permutation FSSP

Non-permutation insertions with Taillard acceleration

Taillard acceleration technique needs adjustments because...

Non-permutation insertions produces invalid $e_{i,j}$ and $q_{i,j}$ when used with m-permutation representation, e.g.:

$$\begin{array}{l} \pi_1 = (4,\ 3) \\ \pi_2 = (4,\ 3) \\ \pi_3 = (4,\ 3) \\ \pi_4 = (3,\ 4) \\ \pi_5 = (3,\ 4) \end{array} \Rightarrow \begin{array}{l} \pi_1 = (4,\ 3,\ 1) \\ \pi_2 = (4,\ 3,\ 1) \\ \pi_3 = (4,\ 1,\ 3) \\ \pi_4 = (3,\ 1,\ 4) \\ \pi_5 = (3,\ 1,\ 4) \end{array} \Rightarrow \begin{array}{l} \pi_1 = (4,\ 3,\ 2, | 1) \\ \pi_2 = (4,\ 3,\ 2, | 1) \\ \pi_3 = (4,\ 1,\ 3, | 2) \\ \pi_4 = (3,\ 1,\ 4, | 2) \\ \pi_5 = (3,\ 1,\ 4, | 2) \end{array}$$

Two possible alternative solutions:

Update invalid $e_{i,j}$ and $q_{i,j}$ efficiently
NFS constructive heuristic $O(n^2m^2W)$ (Benavides & Ritt, 2016)

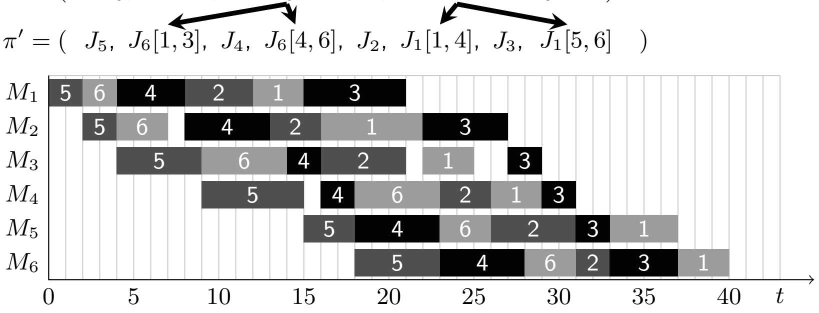
Propose a new representation that supports Taillard acceleration
NEH$_{\text{BR}}$ constructive heuristic $O(n^2m)$ (same as NEH$_T$, Benavides & Ritt, 2018)

New representation for non-permutation schedules: Permutation of pseudo-jobs

Pseudo-job $J_j[i, i']$: operations of job J_j from M_i to $M_{i'}$, others are missing

$$\pi = (\quad J_5, \quad J_4, \quad J_6, \quad J_2, \quad J_1, \quad J_3 \quad)$$

$$\pi' = (\; J_5, \; J_6[1,3], \; J_4, \; J_6[4,6], \; J_2, \; J_1[1,4], \; J_3, \; J_1[5,6] \;)$$



Times $e_{i,j}$ and $q_{i,j}$ are valid, but some operations are missing

Taillard acceleration redefinition: straight insertion

$$e_{i,j} = \begin{cases} \max\{e_{i,j-1}, e_{i-1,j}\} + p_{i,\pi(j)}, & \text{if } \exists\ p_{i,\pi(j)} \\ e_{i,j-1}, & \text{if } \nexists\ p_{i,\pi(j)} \end{cases} \quad \text{for } i \in [m], j \in [|\pi|],$$

with $e_{0,j} = 0$ and $e_{i,0} = 0$

$$q_{i,j} = \begin{cases} \max\{q_{i,j+1}, q_{i+1,j}\} + p_{i,\pi(j)}, & \text{if } \exists\ p_{i,\pi(j)} \\ q_{i,j+1}, & \text{if } \nexists\ p_{i,\pi(j)} \end{cases} \quad \text{for } i \in [m], j \in [|\pi|],$$

with $q_{m+1,j} = 0$ and $q_{i,k+1} = 0$

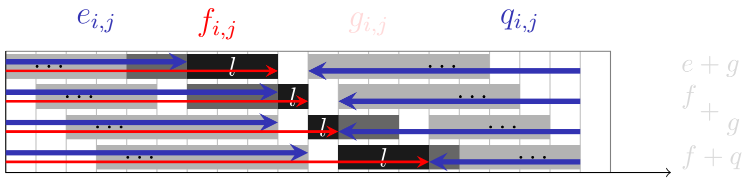
$$f_{i,j} = \max\{f_{i-1,j}, e_{i,j-1}\} + p_{i,\pi_o(l)}, \quad \text{for } i \in [m], j \in [|\pi| + 1] \quad \text{with } f_{0,j} = 0$$

$$MC_j = \max_{i\in[m]}\{f_{i,j} + q_{i,j}\}, \quad \text{for } j \in [|\pi| + 1]$$

Taillard acceleration extension: insertion with anticipation

$$g_{i,j} = \max\{g_{i+1,j}, q_{i,j}\} + p_{i,\pi_o(l)}, \quad \text{for } i \in [m], j \in [|\pi|+1] \quad \text{with } g_{m+1,j} = 0$$

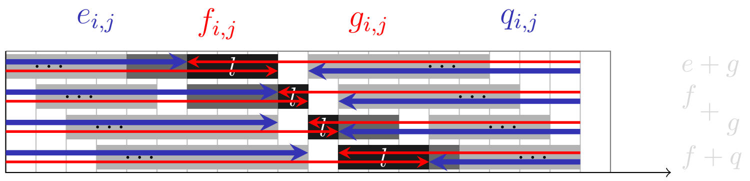
$$MC'_{i,j} = \begin{cases} \max\{f_{i,j+1} + g_{i+1,j}, \\ \qquad \max_{i' \in [i]}\{g_{i',j+1} + e_{i',j}\}, \\ \qquad \max_{i'' \in [i+1,m]}\{f_{i'',j} + q_{i'',j}\}\}, & \text{if } \exists p_{i,\pi(j)} \wedge \exists p_{i+1,\pi(j)} \\ \infty, & \text{if } \nexists p_{i,\pi(j)} \vee \nexists p_{i+1,\pi(j)} \end{cases} \quad \text{for } i \in [2, m-2], j \in [|\pi|]$$



Taillard acceleration extension: insertion with anticipation

$$g_{i,j} = \max\{g_{i+1,j}, q_{i,j}\} + p_{i,\pi_o(l)}, \quad \text{for } i \in [m], j \in [|\pi|+1] \quad \text{with } g_{m+1,j} = 0$$

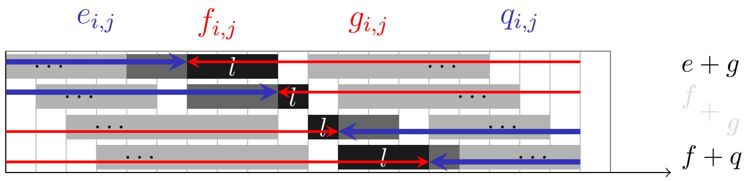
$$MC'_{i,j} = \begin{cases} \max\{f_{i,j+1} + g_{i+1,j}, \\ \qquad \max_{i' \in [i]}\{g_{i',j+1} + e_{i',j}\}, \\ \qquad \max_{i'' \in [i+1,m]}\{f_{i'',j} + q_{i'',j}\}\}, & \text{if } \exists p_{i,\pi(j)} \wedge \exists p_{i+1,\pi(j)} \\ \infty, & \text{if } \nexists p_{i,\pi(j)} \vee \nexists p_{i+1,\pi(j)} \end{cases} \quad \text{for } i \in [2, m-2], j \in [|\pi|]$$



Taillard acceleration extension: insertion with anticipation

$$g_{i,j} = \max\{g_{i+1,j}, q_{i,j}\} + p_{i,\pi_o(l)}, \quad \text{for } i \in [m], j \in [|\pi| + 1] \quad \text{with } g_{m+1,j} = 0$$

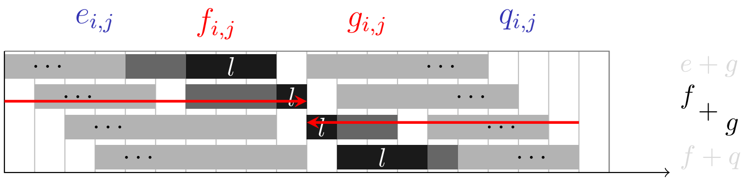
$$MC'_{i,j} = \begin{cases} \max\{f_{i,j+1} + g_{i+1,j}, \\ \qquad \max_{i' \in [i]}\{g_{i',j+1} + e_{i',j}\}, \\ \qquad \max_{i'' \in [i+1,m]}\{f_{i'',j} + q_{i'',j}\}\}, & \text{if } \exists p_{i,\pi(j)} \wedge \exists p_{i+1,\pi(j)} \\ \infty, & \text{if } \nexists p_{i,\pi(j)} \vee \nexists p_{i+1,\pi(j)} \end{cases} \quad \text{for } i \in [2, m-2], j \in [|\pi|]$$



Taillard acceleration extension: insertion with anticipation

$$g_{i,j} = \max\{g_{i+1,j}, q_{i,j}\} + p_{i,\pi_o(l)}, \quad \text{for } i \in [m], j \in [|\pi|+1] \quad \text{with } g_{m+1,j} = 0$$

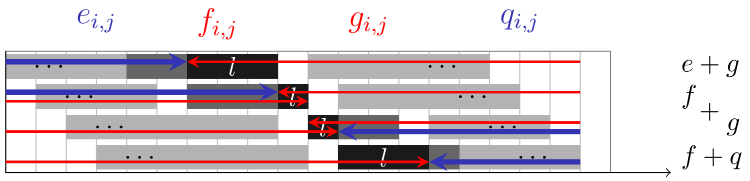
$$MC'_{i,j} = \begin{cases} \max\{f_{i,j+1} + g_{i+1,j}, \\ \qquad \max\limits_{i' \in [i]}\{g_{i',j+1} + e_{i',j}\}, \\ \qquad \max\limits_{i'' \in [i+1,m]}\{f_{i'',j} + q_{i'',j}\}\}, & \text{if } \exists p_{i,\pi(j)} \wedge \exists p_{i+1,\pi(j)} \\ \infty, & \text{if } \nexists p_{i,\pi(j)} \vee \nexists p_{i+1,\pi(j)} \end{cases} \quad \text{for } i \in [2, m-2], j \in [|\pi|]$$



Taillard acceleration extension: insertion with anticipation

$$g_{i,j} = \max\{g_{i+1,j}, q_{i,j}\} + p_{i,\pi_o(l)}, \quad \text{for } i \in [m], j \in [|\pi|+1] \quad \text{with } g_{m+1,j} = 0$$

$$MC'_{i,j} = \begin{cases} \max\{f_{i,j+1} + g_{i+1,j}, & \\ \qquad \max_{i' \in [i]}\{g_{i',j+1} + e_{i',j}\}, & \\ \qquad \max_{i'' \in [i+1,m]}\{f_{i'',j} + q_{i'',j}\}\}, & \text{if } \exists p_{i,\pi(j)} \wedge \exists p_{i+1,\pi(j)} \\ \infty, & \text{if } \nexists p_{i,\pi(j)} \vee \nexists p_{i+1,\pi(j)} \end{cases} \quad \text{for } i \in [2, m-2], j \in [|\pi|]$$

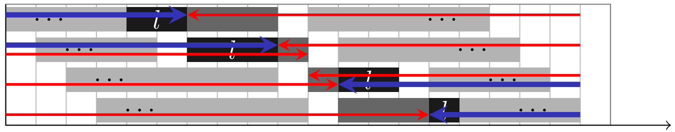


Taillard acceleration extension: insertion with delay

$$e'_{i,j} = \begin{cases} \max\{e'_{i-1,j}, f_{i,j}\} + p_{i,\pi(j)}, & \text{if } \exists\ p_{i,\pi(j)} \\ f_{i,j}, & \text{if } \nexists\ p_{i,\pi(j)} \end{cases} \quad \text{for } i \in [m], j \in [|\pi|] \quad \text{with } e'_{0,j} = 0$$

$$q'_{i,j} = \begin{cases} \max\{q'_{i+1,j}, g_{i,j+1}\} + p_{i,\pi(j)}, & \text{if } \exists\ p_{i,\pi(j)} \\ g_{i,j+1}, & \text{if } \nexists\ p_{i,\pi(j)} \end{cases} \quad \text{for } i \in [m], j \in [|\pi|] \quad \text{with } q'_{m+1,j} = 0$$

$$MC''_{i,j} = \begin{cases} \max\{e'_{i,j} + q'_{i+1,j}, \\ \qquad \max_{i' \in [i]}\{f_{i',j} + q_{i',j}\}, \\ \qquad \max_{i'' \in [i+1,m]}\{g_{i'',j+1} + e_{i'',j}\}\}, & \text{if } \exists p_{i,\pi(j)} \wedge \exists p_{i+1,\pi(j)} \\ \infty, & \text{if } \nexists p_{i,\pi(j)} \vee \nexists p_{i+1,\pi(j)} \end{cases} \quad \text{for } i \in [2, m-2], j \in [|\pi|]$$



$f + q$

$e' + q'$

$e + g$

Constructive heuristic NEH$_{\text{BR}}$ has time complexity of $O(n^2m)$

Besides calculating

MC: makespan for $O(n)$ straight insertions (like NEH)

Calculations are triplicated to obtain:

MC': makespan for $O(nm)$ insertions with anticipation

MC'': makespan for $O(nm)$ insertions with delay

Calculations have time complexity of $O(|\pi|m)$, $n \leq |\pi| \leq 2n$

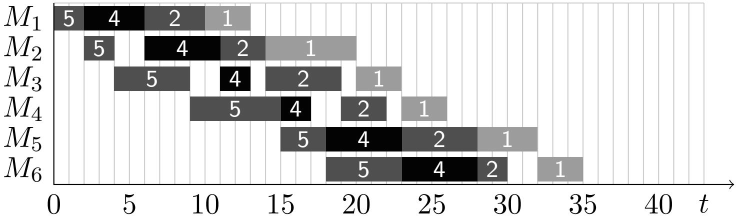
NEH$_{\text{BR}}$ evaluates $O(nm)$ insertion possibilities in time $O(nm)$

NEH$_{\text{BR}}$ has time complexity of $O(n^2m)$

Same time complexity but three times more expensive than NEH$_T$ for permutation FSSP

Constructive heuristic NEH$_{BR}$

$\pi = (J_5, J_4, J_2, J_1)$ Next job: J_3



j	MC_j
1	44
2	41
3	40
4	40
5	39

straight

j	$MC'_{2,j}$	$MC'_{3,j}$	$MC'_{4,j}$
1	46	43	41
2	41	40	40
3	43	40	40
4	40	39	**38**

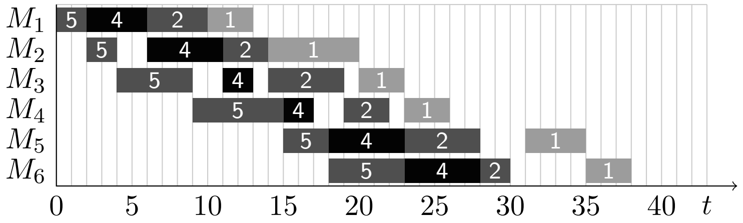
anticipation

j	$MC''_{2,j}$	$MC''_{3,j}$	$MC''_{4,j}$
1	46	46	46
2	42	41	41
3	42	42	42
4	44	44	44

delay

Constructive heuristic NEH$_{BR}$

$\pi = (J_5, J_4, J_2, J_1[1,4], J_3, J_1[5,6])$ with anticipation



j	MC_j
1	44
2	41
3	40
4	40
5	39

straight

j	$MC'_{2,j}$	$MC'_{3,j}$	$MC'_{4,j}$
1	46	43	41
2	41	40	40
3	43	40	40
4	40	39	**38**

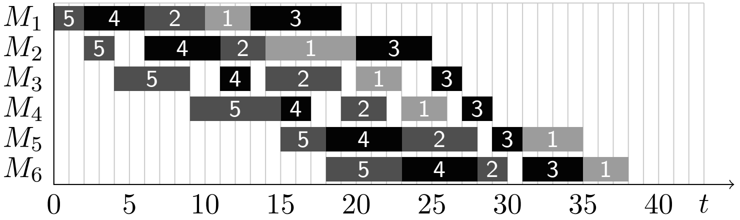
anticipation

j	$MC''_{2,j}$	$MC''_{3,j}$	$MC''_{4,j}$
1	46	46	46
2	42	41	41
3	42	42	42
4	44	44	44

delay

Constructive heuristic NEH$_{\text{BR}}$

$\pi = (J_5, J_4, J_2, J_1[1,4], J_3, J_1[5,6])$ with anticipation



j	MC_j
1	44
2	41
3	40
4	40
5	39

straight

j	$MC'_{2,j}$	$MC'_{3,j}$	$MC'_{4,j}$
1	46	43	41
2	41	40	40
3	43	40	40
4	40	39	**38**

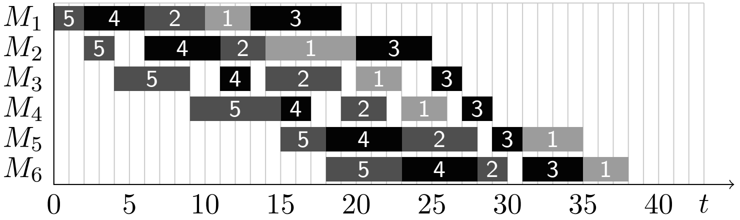
anticipation

j	$MC''_{2,j}$	$MC''_{3,j}$	$MC''_{4,j}$
1	46	46	46
2	42	41	41
3	42	42	42
4	44	44	44

delay

Constructive heuristic NEH$_{\text{BR}}$

$\pi = (J_5, J_4, J_2, J_1[1,4], J_3, J_1[5,6])$ Next job: J_6



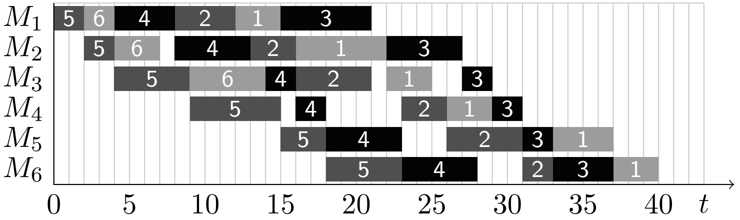
j	MC_j
1	43
2	42
3	41
4	43
5	46
6	48
7	44

j	$MC'_{2,j}$	$MC'_{3,j}$	$MC'_{4,j}$
1	45	47	45
2	46	46	46
3	44	46	46
4	47	47	46
5	51	51	51
6	48	48	48

j	$MC''_{2,j}$	$MC''_{3,j}$	$MC''_{4,j}$
1	45	48	46
2	42	**40**	44
3	46	46	45
4	49	48	45
5	50	47	47
6	44	44	48

Constructive heuristic NEH$_{BR}$

$\pi = (J_5, J_6[1,3], J_4, J_6[4,6], J_2, J_1[1,4], J_3, J_1[5,6])$ with delay



j	MC_j
1	43
2	42
3	41
4	43
5	46
6	48
7	44

straight

j	$MC'_{2,j}$	$MC'_{3,j}$	$MC'_{4,j}$
1	45	47	45
2	46	46	46
3	44	46	46
4	47	47	46
5	51	51	51
6	48	48	48

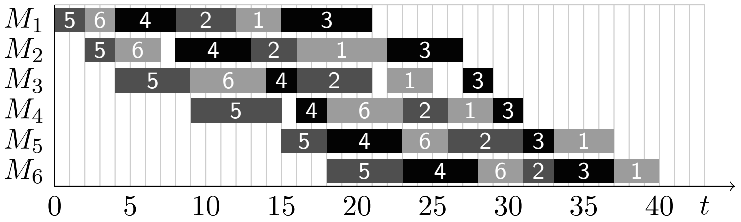
anticipation

j	$MC''_{2,j}$	$MC''_{3,j}$	$MC''_{4,j}$
1	45	48	46
2	42	**40**	44
3	46	46	45
4	49	48	45
5	50	47	47
6	44	44	48

delay

Constructive heuristic NEH$_{\text{BR}}$

$\pi = (J_5, J_6[1,3], J_4, J_6[4,6], J_2, J_1[1,4], J_3, J_1[5,6])$ with delay



j	MC_j
1	43
2	42
3	41
4	43
5	46
6	48
7	44

straight

j	$MC'_{2,j}$	$MC'_{3,j}$	$MC'_{4,j}$
1	45	47	45
2	46	46	46
3	44	46	46
4	47	47	46
5	51	51	51
6	48	48	48

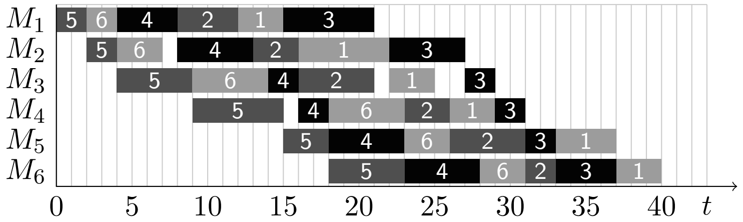
anticipation

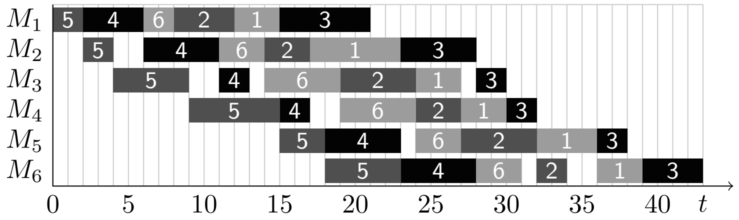
j	$MC''_{2,j}$	$MC''_{3,j}$	$MC''_{4,j}$
1	45	48	46
2	42	**40**	44
3	46	46	45
4	49	48	45
5	50	47	47
6	44	44	48

delay

Constructive heuristic NEH$_{\text{BR}}$

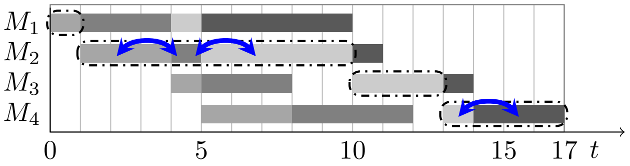
$$\pi = (J_5, J_6[1,3], J_4, J_6[4,6], J_2, J_1[1,4], J_3, J_1[5,6])$$





NEH produces
$\pi' = (J_5, J_4, J_6, J_2, J_1, J_3)$

Local search heuristics for non-permutation FSSP



Extended Neighbourhood of Nowicki & Smutnicki (1996)

Used in (Benavides & Ritt, 2016)

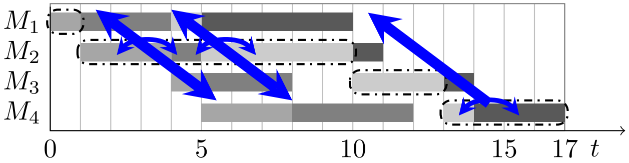
Interchange the first two (or the last two) operations in a critical block

- Evaluate the interchange only on critical machine M_i
- Evaluate the interchange on machines $M_1, \ldots, M_i'$ for all $i' \geq i$
- Evaluate the interchange on machines $M_i'', \ldots, M_m$ for all $i'' \leq i$

Evaluates $O(nm)$ neighbours in time $O(n^2m^2)$

- proposed before pseudo-jobs permutation representation

Local search heuristics for non-permutation FSSP



Extended Neighbourhood of Nowicki & Smutnicki (1996)

Used in (Benavides & Ritt, 2016)

Interchange the first two (or the last two) operations in a critical block

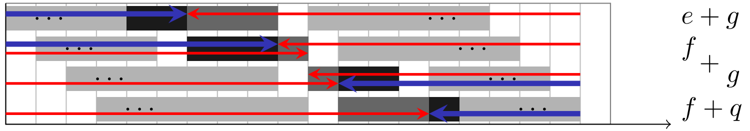
- Evaluate the interchange only on critical machine M_i
- Evaluate the interchange on machines $M_1, \ldots, M_i'$ for all $i' \geq i$
- Evaluate the interchange on machines $M_i'', \ldots, M_m$ for all $i'' \leq i$

Evaluates $O(nm)$ neighbours in time $O(n^2m^2)$

- proposed before pseudo-jobs permutation representation

Local search heuristics for non-permutation FSSP with pseudo-jobs and acceleration

$\pi = (\dots, J_a[1,2], J_b, J_a[3,4], \dots)$



Non-permutation insertion local search $\pi = (J_1, J_2, J_3, J_4, J_5, J_6)$

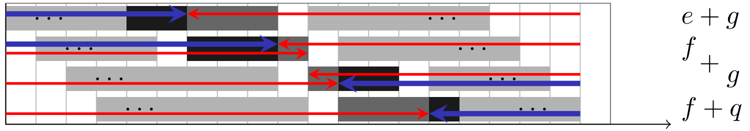
evaluates $(n-1)^2(2m-5)$ non-permutation neighbours in time $O(n^2m)$ same as the insertion local search for $(n-1)^2$ permutation neighbours

New BRN local search $\pi = (J_1, J_2, J_3, J_4, J_5, J_6)$

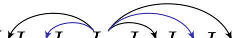
based on swapping adjacent jobs completely or partially
evaluates $(n-1)(2m-5)$ non-permutation neighbours in time $O(nm)$

Local search heuristics for non-permutation FSSP with pseudo-jobs and acceleration

$\pi = (\ldots, J_a[1,2], J_b, J_a[3,4], \ldots)$

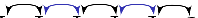


Non-permutation insertion local search $\pi = (J_1, J_2, J_3, J_4, J_5, J_6)$



evaluates $(n-1)^2(2m-5)$ non-permutation neighbours in time $O(n^2m)$ same as the insertion local search for $(n-1)^2$ permutation neighbours

New BRN local search $\pi = (J_1, J_2, J_3, J_4, J_5, J_6)$



based on swapping adjacent jobs completely or partially
evaluates $(n-1)(2m-5)$ non-permutation neighbours in time $O(nm)$

BRN local search

First calculates $e_{i,j}$ and $q_{i,j}$ in a time of complexity $O(nm)$

Best-improvement

chooses the best in the adjacent job swap neighbourhood

Reduced-neighbourhood

$(\pi(j), \pi(j+1)) \in R \iff e_{i,j} + q_{i+1,j} = C_{\max}(\pi) \vee e_{i,j+1} + q_{i+1,j+1} = C_{\max}(\pi)$

Either $\pi(j)$ or $\pi(j+1)$ has critical operations on consecutive machines

Like Nowicki & Smutnicki but considering all the critical paths

Non-permutation

Calculates the makespan of swapping two consecutive jobs $\pi(j), \pi(j+1)$

- MC: swap completely
- MC': swap on the first machines (like insertion with anticipation)
- MC'': swap on the last machines (like insertion with delay)

with a time complexity of $O(m)$ for each $(\pi(j), \pi(j+1)) \in R$, with $|R| < |\pi|$

BRN local search

First calculates $e_{i,j}$ and $q_{i,j}$ in a time of complexity $O(nm)$

Best-improvement

chooses the best in the adjacent job swap neighbourhood

Reduced-neighbourhood

$(\pi(j), \pi(j+1)) \in R \iff e_{i,j} + q_{i+1,j} = C_{\max}(\pi) \vee e_{i,j+1} + q_{i+1,j+1} = C_{\max}(\pi)$

Either $\pi(j)$ or $\pi(j+1)$ has critical operations on consecutive machines

Like Nowicki & Smutnicki but considering all the critical paths

Non-permutation

Calculates the makespan of swapping two consecutive jobs $\pi(j), \pi(j+1)$

- MC: swap completely
- MC': swap on the first machines (like insertion with anticipation)
- MC'': swap on the last machines (like insertion with delay)

with a time complexity of $O(m)$ for each $(\pi(j), \pi(j+1)) \in R$, with $|R| < |\pi|$

BRN local search

First calculates $e_{i,j}$ and $q_{i,j}$ in a time of complexity $O(nm)$

Best-improvement

chooses the best in the adjacent job swap neighbourhood

Reduced-neighbourhood

$(\pi(j), \pi(j+1)) \in R \iff e_{i,j} + q_{i+1,j} = C_{\max}(\pi) \vee e_{i,j+1} + q_{i+1,j+1} = C_{\max}(\pi)$

Either $\pi(j)$ or $\pi(j+1)$ has critical operations on consecutive machines

Like Nowicki & Smutnicki but considering all the critical paths

Non-permutation

Calculates the makespan of swapping two consecutive jobs $\pi(j), \pi(j+1)$

- MC: swap completely
- MC': swap on the first machines (like insertion with anticipation)
- MC'': swap on the last machines (like insertion with delay)

with a time complexity of $O(m)$ for each $(\pi(j), \pi(j+1)) \in R$, with $|R| < |\pi|$

BRN local search

First calculates $e_{i,j}$ and $q_{i,j}$ in a time of complexity $O(nm)$

Best-improvement

chooses the best in the adjacent job swap neighbourhood

Reduced-neighbourhood

$(\pi(j), \pi(j+1)) \in R \iff e_{i,j} + q_{i+1,j} = C_{\max}(\pi) \lor e_{i,j+1} + q_{i+1,j+1} = C_{\max}(\pi)$

Either $\pi(j)$ or $\pi(j+1)$ has critical operations on consecutive machines

Like Nowicki & Smutnicki but considering all the critical paths

Non-permutation

Calculates the makespan of swapping two consecutive jobs $\pi(j), \pi(j+1)$

- MC: swap completely
- MC': swap on the first machines (like insertion with anticipation)
- MC'': swap on the last machines (like insertion with delay)

with a time complexity of $O(m)$ for each $(\pi(j), \pi(j+1)) \in R$, with $|R| < |\pi|$

Outline

Non-permutation FSSP with Cmax (2016)

Benavides, A. J.; Ritt, M. (2016). (first attempt)
Two simple and effective heuristics for minimizing the makespan in non-permutation flow shops.
Computers & Operations Research, Elsevier, v. 66, p. 160–169.
CAPES WebQualis A1; Impact Factor 1.861; 5-Year Impact Factor 2.454

Iterated greedy algorithm for non-permutation FSSP with Cmax

Greedy Reconstruction Perturbation scheme:
Based on NFS, $O(nm^2W)$ per insertion

Local search scheme:
Extended Neighbourhood of Nowicki & Smutnicki

Non-permutation FSSP with Cmax (2016)

Demirkol instances	Lin & Ying	Rossi & Lanzetta	Our IGA min	Our IGA avg	Our IGA max
Averages	0.00	7.99	-1.98	-1.57	-1.13

Our IGA is better in the same adjusted time

Our IGA finds new BKV for the 40 instances

28 Taillard instances	Yagmahan & Yenisey	Rossi & Lanzetta min	Rossi & Lanzetta avg	Our IGA min	Our IGA avg	Our IGA max
Averages	6.86	5.02	5.98	-0.69	-0.51	-0.25

Our IGA is better in the less than their adjusted time

Our IGA finds new BKV for 13 of those 28 instances and 32 of all 120

Better results for $30nm^2$ ms in both cases

Non-permutation FSSP with Cmax (2016)

Demirkol instances	Lin & Ying	Rossi & Lanzetta	Our IGA min	Our IGA avg	Our IGA max
Averages	0.00	7.99	-1.98	-1.57	-1.13

Our IGA is better in the same adjusted time

Our IGA finds new BKV for the 40 instances

28 Taillard instances	Yagmahan & Yenisey	Rossi & Lanzetta min	Rossi & Lanzetta avg	Our IGA min	Our IGA avg	Our IGA max
Averages	6.86	5.02	5.98	-0.69	-0.51	-0.25

Our IGA is better in the less than their adjusted time

Our IGA finds new BKV for 13 of those 28 instances and 32 of all 120

Better results for $30nm^2$ ms in both cases

Non-permutation FSSP with Cmax (2016) compared to permutation FSSP

Taillard	Permutation		Our IGA $30nm$ ms			Our IGA $30nm^2$ ms		
instances	RS	FF	min	avg	max	min	avg	max
Averages	0.44	0.38	-0.22	-0.03	0.21	-0.41	-0.32	-0.20

Fernandez-Viagas & Framinan (2014) is 0.06% better than Ruiz & Stützle (2007)

Our IGA is 0.4% better than Fernandez-Viagas & Framinan (2014) in the same time, and is 0.7% better in $30nm^2$ ms

Small job reordering

Buffer sizes

Non-permutation schedules require slightly smaller buffers

Non-permutation FSSP with Cmax using pseudo-jobs

Benavides, A. J.; Ritt, M. (2018).

Novel pseudo-jobs permutation representation
for non-permutation flow shop schedules

Extended acceleration tech. with the same time complexity

NEH_T, NEH_BR: $O(n^2m)$	(Permutation and non-permutation)
BRN local search: $O(nm)$ per neighbourhood	(non-permutation)
Insertion local search: $O(n^2m)$ per neighbourhood	(Permutation and non-permutation)

FRB_BR based on Farahmand Rad, Ruiz & Boroojerdian (2009)

- produces better results than NEH_BR, more complex and expensive
- different initial solutions slightly affect IG_b

Non-permutation FSSP with Cmax using pseudo-jobs

Benavides, A. J.; Ritt, M. (2018).

Novel pseudo-jobs permutation representation
for non-permutation flow shop schedules

Extended acceleration tech. with the same time complexity

NEH_T, NEH_BR: $O(n^2m)$	(Permutation and non-permutation)
BRN local search: $O(nm)$ per neighbourhood	(non-permutation)
Insertion local search: $O(n^2m)$ per neighbourhood	(Permutation and non-permutation)

FRB_BR based on Farahmand Rad, Ruiz & Boroojerdian (2009)

- produces better results than NEH_BR, more complex and expensive
- different initial solutions slightly affect IG_b

Non-permutation FSSP with Cmax using pseudo-jobs

Benavides, A. J.; Ritt, M. (2018).
Makespan in non-permutation flow shop scheduling problem by the price of permutation.

Iterated greedy algorithms

IGA	Reconstr.	Local search
IG_b	NEH_BR	BRN
IG_i	NEH_BR	Insertion
IG_bi	NEH_BR	BRN, Insertion
IG_p	NEH	Permutation insertion

NEH_T, NEH_BR: $O(n^2m)$ (Permutation and non-permutation)

BRN local search: $O(nm)$ per neighbourhood (non-permutation)

Insertion local search: $O(n^2m)$ per neighbourhood (Permutation and non-permutation)

IG_b is the best combination for non-permutation FSSP

Non-permutation FSSP with Cmax using pseudo-jobs

Benavides, A. J.; Ritt, M. (2018).

Novel pseudo-jobs permutation representation
for non-permutation flow shop schedules

Extended acceleration tech. with the same time complexity

NEH$_\mathsf{T}$, NEH$_\mathsf{BR}$: $O(n^2m)$ (Permutation and non-permutation)

BRN local search: $O(nm)$ per neighbourhood (non-permutation)

Insertion local search: $O(n^2m)$ per neighbourhood (Permutation and non-permutation)

FRB$_\mathsf{BR}$ based on Farahmand Rad, Ruiz & Boroojerdian (2009)

- produces better results than NEH$_\mathsf{BR}$, more complex and expensive
- different initial solutions slightly affect IG$_\mathsf{b}$

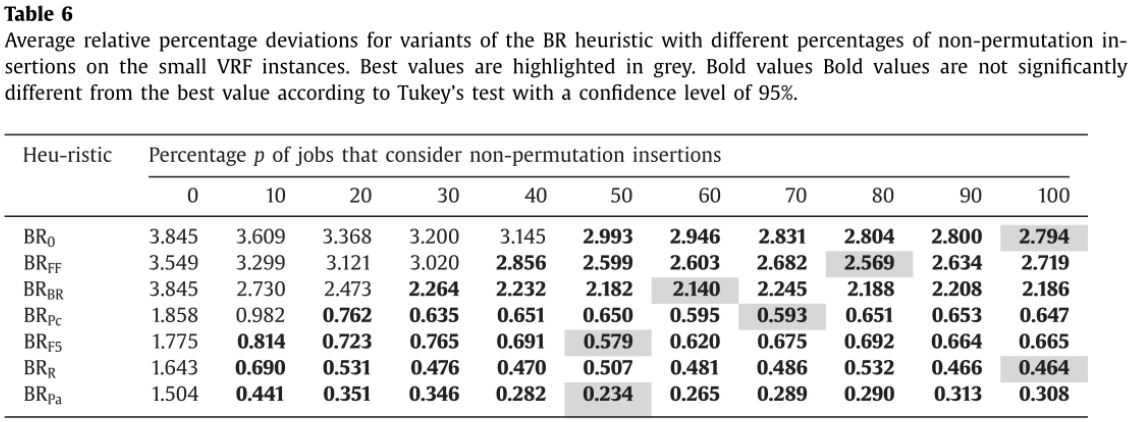


Table 6

Average relative percentage deviations for variants of the BR heuristic with different percentages of non-permutation insertions on the small VRF instances. Best values are highlighted in grey. Bold values Bold values are not significantly different from the best value according to Tukey's test with a confidence level of 95%.

Heu-ristic	Percentage *p* of jobs that consider non-permutation insertions										
	0	10	20	30	40	50	60	70	80	90	100
BR_0	3.845	3.609	3.368	3.200	3.145	**2.993**	**2.946**	**2.831**	**2.804**	**2.800**	**2.794**
BR_{FF}	3.549	3.299	3.121	3.020	**2.856**	**2.599**	**2.603**	**2.682**	**2.569**	**2.634**	**2.719**
BR_{BR}	3.845	2.730	2.473	**2.264**	**2.232**	**2.182**	**2.140**	**2.245**	**2.188**	**2.208**	**2.186**
BR_{Pc}	1.858	0.982	**0.762**	**0.635**	**0.651**	**0.650**	**0.595**	**0.593**	**0.651**	**0.653**	**0.647**
BR_{F5}	1.775	**0.814**	**0.723**	**0.765**	**0.691**	**0.579**	**0.620**	**0.675**	**0.692**	**0.664**	**0.665**
BR_R	1.643	**0.690**	**0.531**	**0.476**	**0.470**	**0.507**	**0.481**	**0.486**	**0.532**	**0.466**	**0.464**
BR_{Pa}	1.504	**0.441**	**0.351**	**0.346**	**0.282**	**0.234**	**0.265**	**0.289**	**0.290**	**0.313**	**0.308**

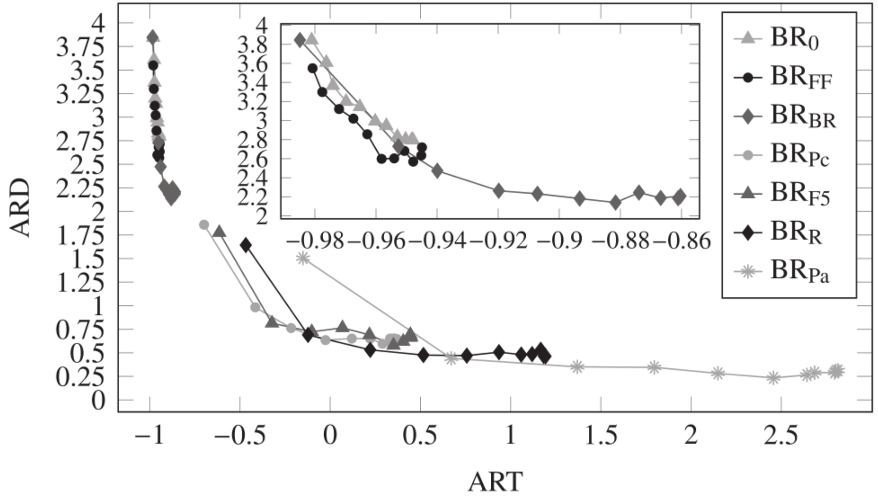


Fig. 5. Computational efficiency of the constructive heuristics on the smaller VRF instances.

Table 9
ARD of the permutation variants $IG_{c,\,l}$ for different time limits, constructive heuristics and local searches on the Taillard instances.

$IG_{c,\,l}$	Time limit 15*nm* ms				Time limit 30*nm* ms				Time limit 45*nm* ms			
	Pa	Pc	Ins	Avg.	Pa	Pc	Ins	Avg.	Pa	Pc	Ins	Avg.
BR_0	0.320	0.291	0.294	0.302	0.269	0.247	0.247	0.254	0.243	0.225	0.224	0.230
BR_{FF}	0.305	0.292	0.299	0.299	0.257	0.242	0.253	0.251	0.231	**0.218**	0.230	0.226
BR_{Pc}	0.309	**0.284**	0.287	0.293	0.265	0.240	0.244	0.250	0.243	**0.219**	0.222	0.228
BR_{F5}	0.315	0.287	0.294	0.299	0.270	0.242	0.248	0.254	0.245	**0.219**	0.225	0.230
BR_R	0.311	**0.281**	0.290	0.294	0.265	**0.236**	0.244	0.248	0.238	**0.211**	0.219	0.223
BR_{Pa}	0.293	0.264	**0.273**	0.277	0.248	0.224	**0.230**	0.234	0.226	0.205	**0.209**	0.213
Avg.	0.309	0.283	0.290		0.262	0.239	0.244		0.238	0.216	0.221	

Table 10
ARD of the non-permutation variants of $IG_{c,\,l}$ for different time limits, constructive heuristics, and local searches on the Taillard instances.

$IG_{c,\,l}$	Time limit 15*nm* ms					Time limit 30*nm* ms					Time limit 45*nm* ms				
	Pa	Pc	Ins	RNB	Avg.	Pa	Pc	Ins	RNB	Avg.	Pa	Pc	Ins	RNB	Avg.
BR_0	−0.103	−0.147	−0.144	−0.246	−0.160	−0.179	−0.216	−0.217	−0.315	−0.232	−0.218	−0.252	−0.258	**−0.348**	−0.269
BR_{FF}	−0.103	−0.147	−0.145	−0.245	−0.160	−0.176	−0.216	−0.220	**−0.317**	−0.232	−0.215	−0.251	−0.258	**−0.353**	−0.269
BR_{BR}	−0.110	−0.156	−0.150	**−0.251**	−0.167	−0.185	−0.225	−0.225	**−0.319**	−0.238	−0.223	−0.257	−0.262	**−0.353**	−0.273
BR_{Pc}	−0.124	−0.162	−0.162	**−0.261**	−0.177	−0.189	−0.225	−0.228	**−0.324**	−0.242	−0.223	−0.256	−0.263	**−0.354**	−0.274
BR_{F5}	−0.120	−0.160	−0.162	**−0.251**	−0.173	−0.189	−0.224	−0.226	**−0.315**	−0.239	−0.225	−0.256	−0.262	**−0.351**	−0.273
BR_R	−0.126	−0.166	−0.195	**−0.257**	−0.186	−0.193	−0.227	−0.286	**−0.319**	−0.256	−0.233	−0.260	−0.340	**−0.350**	−0.296
BR_{Pa}	−0.146	−0.179	−0.187	−0.268	−0.195	−0.207	−0.241	−0.251	−0.330	−0.257	−0.241	−0.273	−0.285	−0.360	−0.290
Avg.	−0.119	−0.160	−0.164	−0.254		−0.188	−0.225	−0.236	−0.320		−0.225	−0.258	−0.275	−0.353	

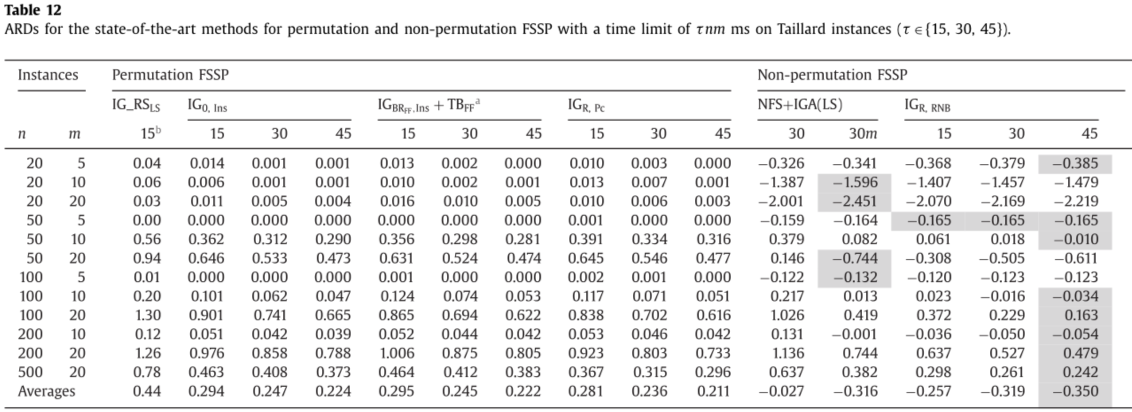


Table 12
ARDs for the state-of-the-art methods for permutation and non-permutation FSSP with a time limit of τnm ms on Taillard instances ($\tau \in \{15, 30, 45\}$).

Instances		Permutation FSSP										Non-permutation FSSP				
		IG_RS$_{LS}$	IG$_{0,\ Ins}$			IG$_{BR_{FF},Ins}$ + TB$_{FF}$[a]			IG$_{R,\ Pc}$			NFS+IGA(LS)		IG$_{R,\ RNB}$		
n	m	15[b]	15	30	45	15	30	45	15	30	45	30	30m	15	30	45
20	5	0.04	0.014	0.001	0.001	0.013	0.002	0.000	0.010	0.003	0.000	−0.326	−0.341	−0.368	−0.379	−0.385
20	10	0.06	0.006	0.001	0.001	0.010	0.002	0.001	0.013	0.007	0.001	−1.387	−1.596	−1.407	−1.457	−1.479
20	20	0.03	0.011	0.005	0.004	0.016	0.010	0.005	0.010	0.006	0.003	−2.001	−2.451	−2.070	−2.169	−2.219
50	5	0.00	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.000	−0.159	−0.164	−0.165	−0.165	−0.165
50	10	0.56	0.362	0.312	0.290	0.356	0.298	0.281	0.391	0.334	0.316	0.379	0.082	0.061	0.018	−0.010
50	20	0.94	0.646	0.533	0.473	0.631	0.524	0.474	0.645	0.546	0.477	0.146	−0.744	−0.308	−0.505	−0.611
100	5	0.01	0.000	0.000	0.000	0.001	0.000	0.000	0.002	0.001	0.000	−0.122	−0.132	−0.120	−0.123	−0.123
100	10	0.20	0.101	0.062	0.047	0.124	0.074	0.053	0.117	0.071	0.051	0.217	0.013	0.023	−0.016	−0.034
100	20	1.30	0.901	0.741	0.665	0.865	0.694	0.622	0.838	0.702	0.616	1.026	0.419	0.372	0.229	0.163
200	10	0.12	0.051	0.042	0.039	0.052	0.044	0.042	0.053	0.046	0.042	0.131	−0.001	−0.036	−0.050	−0.054
200	20	1.26	0.976	0.858	0.788	1.006	0.875	0.805	0.923	0.803	0.733	1.136	0.744	0.637	0.527	0.479
500	20	0.78	0.463	0.408	0.373	0.464	0.412	0.383	0.367	0.315	0.296	0.637	0.382	0.298	0.261	0.242
Averages		0.44	0.294	0.247	0.224	0.295	0.245	0.222	0.281	0.236	0.211	−0.027	−0.316	−0.257	−0.319	−0.350

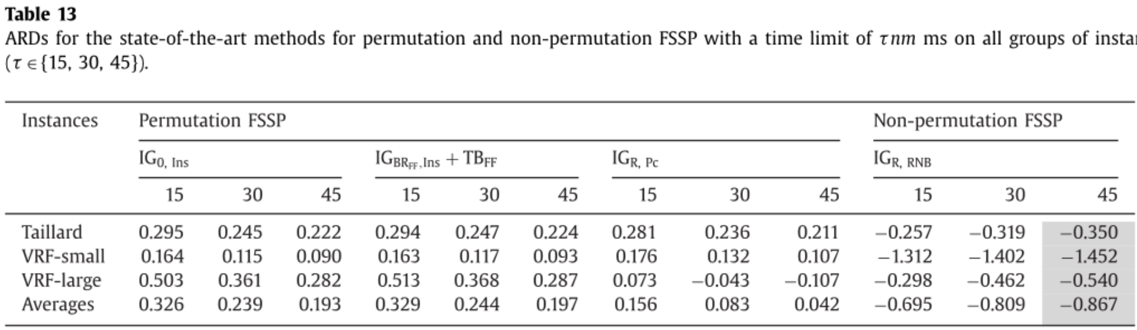


Table 13
ARDs for the state-of-the-art methods for permutation and non-permutation FSSP with a time limit of τnm ms on all groups of insta
($\tau \in \{15, 30, 45\}$).

Instances	Permutation FSSP									Non-permutation FSSP		
	$IG_{0,\ Ins}$			$IG_{BR_{FF},Ins} + TB_{FF}$			$IG_{R,\ Pc}$			$IG_{R,\ RNB}$		
	15	30	45	15	30	45	15	30	45	15	30	45
Taillard	0.295	0.245	0.222	0.294	0.247	0.224	0.281	0.236	0.211	−0.257	−0.319	−0.350
VRF-small	0.164	0.115	0.090	0.163	0.117	0.093	0.176	0.132	0.107	−1.312	−1.402	−1.452
VRF-large	0.503	0.361	0.282	0.513	0.368	0.287	0.073	−0.043	−0.107	−0.298	−0.462	−0.540
Averages	0.326	0.239	0.193	0.329	0.244	0.197	0.156	0.083	0.042	−0.695	−0.809	−0.867

Non-permutation FSSP
Concluding Remarks

Non-permutation schedules can be represented as a permutation of pseudo-jobs, and this allows the use of an extended taillard acceleration and a BRN local search.

Strategic operation reordering leads to non-permutation schedules with better quality than the best possible permutation schedules.

Non-permutation schedules can be found using the same computational effort than the used for permutation schedules
with the makespan and the total completion time criteria.

Non-permutation schedules can be implemented in practice without strong technological differences.

Solving the Non-permutation Flow Shop Scheduling Problem

Alexander J. Benavides

ajbenavides@unsa.edu.pe
ajbenavides@ucsp.edu.pe

Thank you!
Questions?