Matheuristics

Jean-Louis Bouquard

Beijing Institute of Technology

Optimization Methods

Content

1. Generalities
2. Usual neighborhood
3. Other neighborhoods
   - The Shake model
   - The Swap model
   - The extended swap model
   - The EBSR and EFSR models
   - The Random model
4. Conclusion

1 Generalities

Matheuristics

- Matheuristics
- reference to Metaheuristics
- or Math + Heuristics
- First papers: 2009, Hybridizing Metaheuristics and Mathematical programming (V. Maniezzo, T. Stützle, S. Voß, Annals of Information Systems, Springer)

Matheuristics

- Matheuristics
- reference to Metaheuristics
- or Math + Heuristics
- First papers: 2009, Hybridizing Metaheuristics and Mathematical programming (V. Maniezzo, T. Stützle, S. Voß, Annals of Information Systems, Springer)

Matheuristics

- Matheuristics
- reference to Metaheuristics
- or Math $+$ Heuristics
- First papers: 2009, Hybridizing Metaheuristics and Mathematical programming (V. Maniezzo, T. Stützle, S. Voß, Annals of Information Systems, Springer)

Matheuristics

- Matheuristics
- reference to Metaheuristics
- or Math + Heuristics
- First papers: 2009, Hybridizing Metaheuristics and Mathematical programming (V. Maniezzo, T. Stützle, S. Voß, Annals of Information Systems, Springer)

Matheuristics

- In Iterated Local Search (ILS)
- Neighborhoods are searched by enumeration
- that is (local) brute force
- we could use MILP to do it!

Matheuristics

- In Iterated Local Search (ILS)
- Neighborhoods are searched by enumeration
- that is (local) brute force
- we could use MILP to do it!

Matheuristics

- In Iterated Local Search (ILS)
- Neighborhoods are searched by enumeration
- that is (local) brute force
- we could use MILP to do it!

Matheuristics

- In Iterated Local Search (ILS)
- Neighborhoods are searched by enumeration
- that is (local) brute force
- we could use MILP to do it!

2 Usual neighborhood

Defining a neighborhood

- Back to our F2//$\sum T_j$ problem
- Restrict the search to a reasonnable amount of jobs
- Given a solution $s = (s_1, \ldots, s_n)$
- and an interval $[h..h + w - 1] \subset [1..n]$, the width is w
- let us consider the neighborhood defined by:
- all the solutions that are equal to s except on $[h..h + w - 1]$
- $\mathcal{N}_{h,w}(s) = \{\sigma / \sigma_j = s_j \; for \; j \notin [h..h + w - 1]\}$

Defining a neighborhood

- Back to our F2//$\sum T_j$ problem
- Restrict the search to a reasonnable amount of jobs
- Given a solution $s = (s_1, \ldots, s_n)$
- and an interval $[h..h + w - 1] \subset [1..n]$, the width is w
- let us consider the neighborhood defined by:
- all the solutions that are equal to s except on $[h..h + w - 1]$
- $\mathcal{N}_{h,w}(s) = \{\sigma / \sigma_j = s_j \; for \; j \notin [h..h + w - 1]\}$

Defining a neighborhood

- Back to our F2//$\sum T_j$ problem
- Restrict the search to a reasonnable amount of jobs
- Given a solution $s = (s_1, \ldots, s_n)$
- and an interval $[h..h + w - 1] \subset [1..n]$, the width is w
- let us consider the neighborhood defined by:
- all the solutions that are equal to s except on $[h..h + w - 1]$
- $\mathcal{N}_{h,w}(s) = \{\sigma / \sigma_j = s_j \ for\ j \notin [h..h + w - 1]\}$

Defining a neighborhood

- Back to our F2//$\sum T_j$ problem
- Restrict the search to a reasonnable amount of jobs
- Given a solution $s = (s_1, \ldots, s_n)$
- and an interval $[h..h+w-1] \subset [1..n]$, the width is w
- let us consider the neighborhood defined by:
- all the solutions that are equal to s except on $[h..h+w-1]$
- $\mathcal{N}_{h,w}(s) = \{\sigma / \sigma_j = s_j \text{ for } j \notin [h..h+w-1]\}$

Defining a neighborhood

- Back to our F2//$\sum T_j$ problem
- Restrict the search to a reasonnable amount of jobs
- Given a solution $s = (s_1, \ldots, s_n)$
- and an interval $[h..h + w - 1] \subset [1..n]$, the width is w
- let us consider the neighborhood defined by:
- all the solutions that are equal to s except on $[h..h + w - 1]$
- $\mathcal{N}_{h,w}(s) = \{\sigma / \sigma_j = s_j \; for \; j \notin [h..h + w - 1]\}$

Defining a neighborhood

- Back to our F2//$\sum T_j$ problem
- Restrict the search to a reasonnable amount of jobs
- Given a solution $s = (s_1, \ldots, s_n)$
- and an interval $[h..h + w - 1] \subset [1..n]$, the width is w
- let us consider the neighborhood defined by:
- all the solutions that are equal to s except on $[h..h + w - 1]$
- $\mathcal{N}_{h,w}(s) = \{\sigma / \sigma_j = s_j \text{ for } j \notin [h..h + w - 1]\}$

Defining a neighborhood

- Back to our F2//$\sum T_j$ problem
- Restrict the search to a reasonnable amount of jobs
- Given a solution $s = (s_1, \ldots, s_n)$
- and an interval $[h..h + w - 1] \subset [1..n]$, the width is w
- let us consider the neighborhood defined by:
- all the solutions that are equal to s except on $[h..h + w - 1]$
- $\mathcal{N}_{h,w}(s) = \{\sigma / \sigma_j = s_j \textit{ for } j \notin [h..h + w - 1]\}$

Defining a neighborhood

- There are $w!$ elements in this neighborhood
- We will search for the best solution of this neighborhood
- a best local solution
- Not using brute force, but MILP
- For the model, we can use the lesson 3 model
- and fix all the variables we know
- or we can build a dedicated model

Defining a neighborhood

- There are $w!$ elements in this neighborhood
- We will search for the best solution of this neighborhood
- a best local solution
- Not using brute force, but MILP
- For the model, we can use the lesson 3 model
- and fix all the variables we know
- or we can build a dedicated model

Defining a neighborhood

- There are $w!$ elements in this neighborhood
- We will search for the best solution of this neighborhood
- a best local solution
- Not using brute force, but MILP
- For the model, we can use the lesson 3 model
- and fix all the variables we know
- or we can build a dedicated model

Defining a neighborhood

- There are $w!$ elements in this neighborhood
- We will search for the best solution of this neighborhood
- a best local solution
- Not using brute force, but MILP
- For the model, we can use the lesson 3 model
- and fix all the variables we know
- or we can build a dedicated model

Defining a neighborhood

- There are $w!$ elements in this neighborhood
- We will search for the best solution of this neighborhood
- a best local solution
- Not using brute force, but MILP
- For the model, we can use the lesson 3 model
- and fix all the variables we know
- or we can build a dedicated model

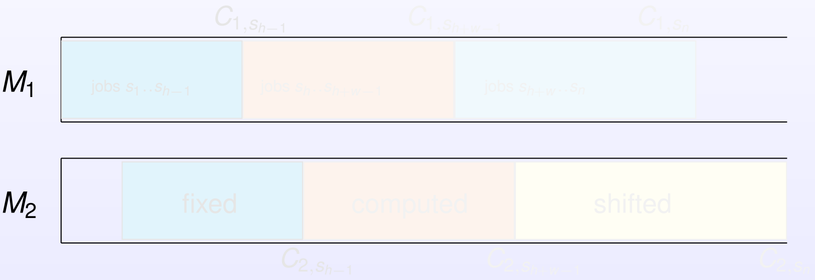
Defining a neighborhood

- There are $w!$ elements in this neighborhood
- We will search for the best solution of this neighborhood
- a best local solution
- Not using brute force, but MILP
- For the model, we can use the lesson 3 model
- and fix all the variables we know
- or we can build a dedicated model

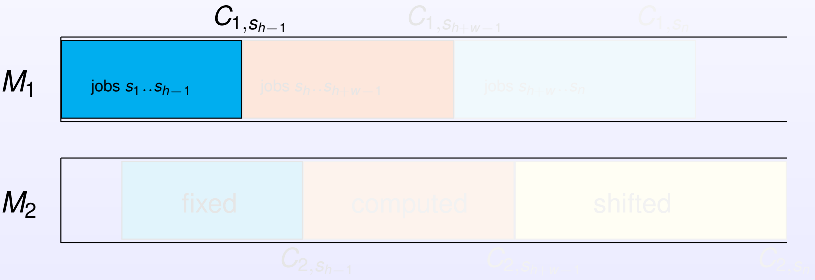
Defining a neighborhood

- There are $w!$ elements in this neighborhood
- We will search for the best solution of this neighborhood
- a best local solution
- Not using brute force, but MILP
- For the model, we can use the lesson 3 model
- and fix all the variables we know
- or we can build a dedicated model

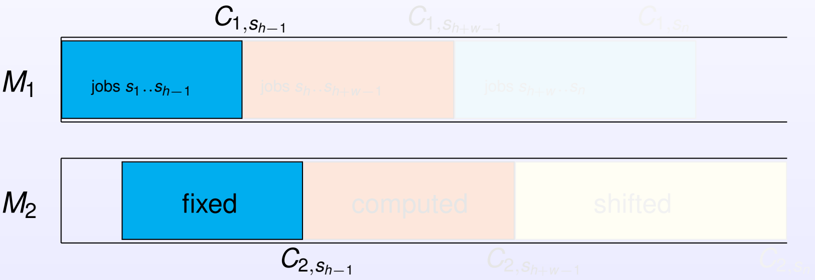
Defining a neighborhood



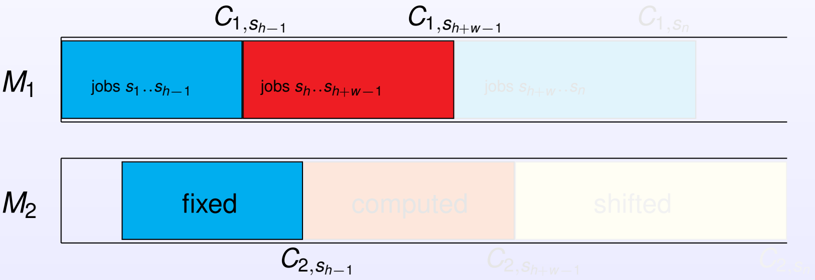
Defining a neighborhood



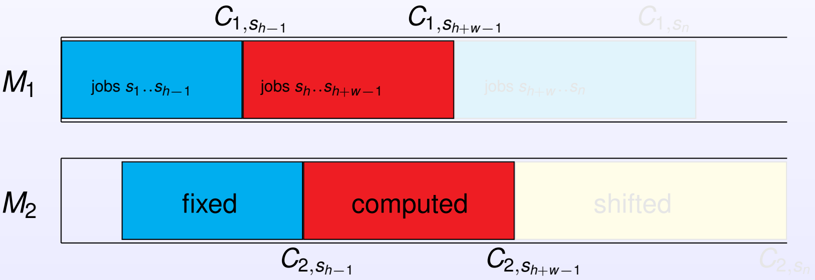
Defining a neighborhood



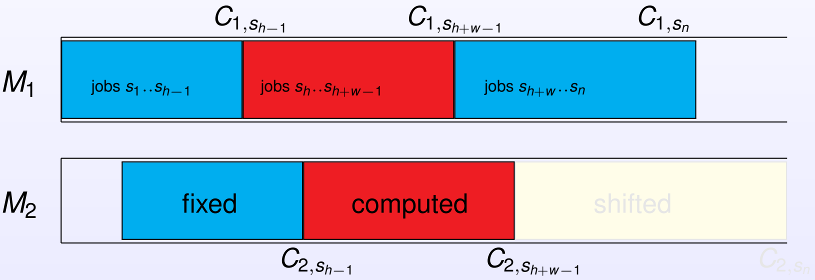
Defining a neighborhood



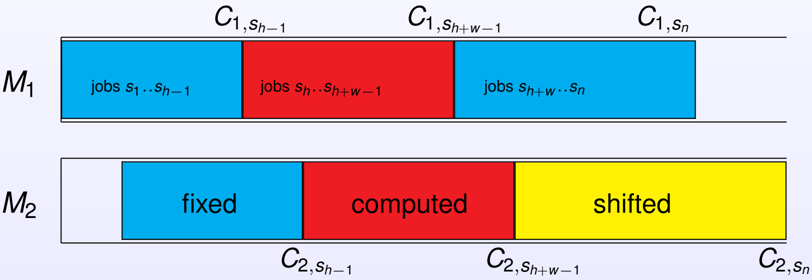
Defining a neighborhood



Defining a neighborhood



Defining a neighborhood



A model for the neighborhood
Variables

- Metadata: from s we compute useful values
- Variables are:
- - $X_{k,j}$ for $k \in [0..w-1]$ and $j \in [0..w-1]$.
  $X_{k,j} = 1$ if and only if the job $s[h+j]$ is in position $h+k$
- - $F1_k$, $F2_k$ and G_k for $k \in [0..w-1]$.
  They correspond to the job in position $h+k$
- - $C2_j$, T_j for the jobs in position $\in [h+w..n]$

A model for the neighborhood
Variables

- Metadata: from s we compute useful values
- Variables are:
  - $X_{k,j}$ for $k \in [0..w-1]$ and $j \in [0..w-1]$.
    $X_{k,j} = 1$ if and only if the job $s[h+j]$ is in position $h+k$
  - $F1_k$, $F2_k$ and G_k for $k \in [0..w-1]$.
    They correspond to the job in position $h+k$
  - $C2_j$, T_j for the jobs in position $\in [h+w..n]$

A model for the neighborhood
Variables

- Metadata: from s we compute useful values
- Variables are:
- • $X_{k,j}$ for $k \in [0..w-1]$ and $j \in [0..w-1]$.
  $X_{k,j} = 1$ if and only if the job $s[h+j]$ is in position $h+k$
- • $F1_k$, $F2_k$ and G_k for $k \in [0..w-1]$.
  They correspond to the job in position $h+k$
- • $C2_j$, T_j for the jobs in position $\in [h+w..n]$

A model for the neighborhood
Variables

- Metadata: from s we compute useful values
- Variables are:
- • $X_{k,j}$ for $k \in [0..w-1]$ and $j \in [0..w-1]$.
  $X_{k,j} = 1$ if and only if the job $s[h+j]$ is in position $h+k$
- • $F1_k$, $F2_k$ and G_k for $k \in [0..w-1]$.
  They correspond to the job in position $h+k$
- • $C2_j$, T_j for the jobs in position $\in [h+w..n]$

A model for the neighborhood
Variables

- Metadata: from s we compute useful values
- Variables are:
- • $X_{k,j}$ for $k \in [0..w-1]$ and $j \in [0..w-1]$.
  $X_{k,j} = 1$ if and only if the job $s[h+j]$ is in position $h+k$
- • $F1_k$, $F2_k$ and G_k for $k \in [0..w-1]$.
  They correspond to the job in position $h+k$
- • $C2_j$, T_j for the jobs in position $\in [h+w..n]$

A model for the neighborhood
Constraints

- Each rank is assigned to a unique job (rows of the matrix $(X_{k,j})$):

$$\forall k \in [0..w-1] \quad \sum_{j=0}^{w-1} X_{k,j} = 1 \qquad (1)$$

- Each job has got a unique rank (columns of the matrix $(X_{k,j})$):

$$\forall j \in [0..w-1] \quad \sum_{k=0}^{w-1} X_{k,j} = 1 \qquad (2)$$

A model for the neighborhood
Constraints

- Each rank is assigned to a unique job (rows of the matrix $(X_{k,j})$):

$$\forall k \in [0..w-1] \quad \sum_{j=0}^{w-1} X_{k,j} = 1 \qquad (1)$$

- Each job has got a unique rank (columns of the matrix $(X_{k,j})$):

$$\forall j \in [0..w-1] \quad \sum_{k=0}^{w-1} X_{k,j} = 1 \qquad (2)$$

A model for the neighborhood
Constraints

- Completion time on $M1$ of the job in position h (first red job)

$$F1_0 = f1_{h-1} + \sum_{j=0}^{w-1} p1_{s_{(h+j)}} X_{0,j} \qquad (3)$$

- Completion times on $M1$ of the jobs in position $h+1..h+w-1$ (other red jobs)

$$\forall k \in [1..w-1] \quad F1_k = F1_{k-1} + \sum_{j=0}^{w-1} p1_{s_{(h+j)}} X_{k,j} \qquad (4)$$

A model for the neighborhood
Constraints

- Completion time on $M1$ of the job in position h (first red job)

$$F1_0 = f1_{h-1} + \sum_{j=0}^{w-1} p1_{s_{(h+j)}} X_{0,j} \quad (3)$$

- Completion times on $M1$ of the jobs in position $h+1..h+w-1$ (other red jobs)

$$\forall k \in [1..w-1] \quad F1_k = F1_{k-1} + \sum_{j=0}^{w-1} p1_{s_{(h+j)}} X_{k,j} \quad (4)$$

A model for the neighborhood
Constraints

- Completion time on *M2* of the job in position *h* (first red job)

$$F2_0 \geq f2_{h-1} + \sum_{j=0}^{w-1} p2_{s_{(h+j)}} X_{0,j} \qquad (5)$$

- Completion times on *M2* of the jobs in position $h+1..h+w-1$ (other red jobs)

$$\forall k \in [1..w-1] \quad F2_k \geq F2_{k-1} + \sum_{j=0}^{w-1} p2_{s_{(h+j)}} X_{k,j} \qquad (6)$$

A model for the neighborhood
Constraints

- Completion time on $M2$ of the job in position h (first red job)

$$F2_0 \geq f2_{h-1} + \sum_{j=0}^{w-1} p2_{s_{(h+j)}} X_{0,j} \qquad (5)$$

- Completion times on $M2$ of the jobs in position $h+1..h+w-1$ (other red jobs)

$$\forall k \in [1..w-1] \quad F2_k \geq F2_{k-1} + \sum_{j=0}^{w-1} p2_{s_{(h+j)}} X_{k,j} \qquad (6)$$

A model for the neighborhood
Constraints

- Completion times on $M2$ of the jobs in position $h..h+w-1$ (red jobs): $M2$ is after $M1$

$$\forall k \in [0..w-1] \quad F2_k \geq F1_k + \sum_{j=0}^{w-1} p2_{s_{(h+j)}} X_{k,j} \qquad (7)$$

- Tardinesses of the jobs in position $h..h+w-1$ (red jobs):

$$\forall k \in [0..w-1] \quad G_k \geq F2_k - \sum_{j=0}^{w-1} d_{s_{(h+j)}} X_{k,j} \qquad (8)$$

A model for the neighborhood

Constraints

- Completion times on $M2$ of the jobs in position $h..h+w-1$ (red jobs): $M2$ is after $M1$

$$\forall k \in [0..w-1] \quad F2_k \geq F1_k + \sum_{j=0}^{w-1} p2_{s_{(h+j)}} X_{k,j} \qquad (7)$$

- Tardinesses of the jobs in position $h..h+w-1$ (red jobs):

$$\forall k \in [0..w-1] \quad G_k \geq F2_k - \sum_{j=0}^{w-1} d_{s_{(h+j)}} X_{k,j} \qquad (8)$$

A model for the neighborhood
Constraints

- Completion times on $M2$ of the jobs in position $h+w..n$ (yellow jobs):

$$\forall j \in [0..n-h-w] \quad C2_j \geq \pi^1_{h+w,h+w+j} \quad (9)$$

$$C2_j \geq F2_{w-1} + \pi^2_{h+w,h+w+j} \quad (10)$$

- Tardinesses of the jobs in position $h+w..n$ (yellow jobs):

$$\forall j \in [0..n-h-w] \quad T_j \geq C2_j - d_{s_{(h+w+j)}} \quad (11)$$

A model for the neighborhood
Constraints

- Completion times on $M2$ of the jobs in position $h+w..n$ (yellow jobs):

$$\forall j \in [0..n-h-w] \quad C2_j \geq \pi^1_{h+w,h+w+j} \qquad (9)$$

$$C2_j \geq F2_{w-1} + \pi^2_{h+w,h+w+j} \qquad (10)$$

- Tardinesses of the jobs in position $h+w..n$ (yellow jobs):

$$\forall j \in [0..n-h-w] \quad T_j \geq C2_j - d_{s_{(h+w+j)}} \qquad (11)$$

A model for the neighborhood

Constraint of improvement and objective function

- We want to improve the current solution:

$$\sum_{j=1}^{h-1} g_j + \sum_{k=0}^{w-1} G_k + \sum_{j=0}^{n-h-w} T_j \leq tbar(s) - 1 \qquad (12)$$

- Objective function:

$$Minimize \quad \sum_{k=0}^{w-1} G_k + \sum_{j=0}^{n-h-w} T_j$$

A model for the neighborhood

Constraint of improvement and objective function

- We want to improve the current solution:

$$\sum_{j=1}^{h-1} g_j + \sum_{k=0}^{w-1} G_k + \sum_{j=0}^{n-h-w} T_j \leq tbar(s) - 1 \qquad (12)$$

- Objective function:

$$Minimize \quad \sum_{k=0}^{w-1} G_k + \sum_{j=0}^{n-h-w} T_j$$

Running this model

Function One-Pass(w,step)

1: *improved* $\leftarrow$ **false**
2: **for** h **from** 0 **while**$(h + w \leq n)$ **by** *step* **do**
3:     Run the model
4:     **if** (there is a solution *news*) **then**
5:         $s \leftarrow$ *news*
6:         *improved* $\leftarrow$ **true**
7:     **end if**
8: **end for**
9: **return** *improved*

- w: width of the interval
- *step*: pace for the progression of the interval
- line 4: if there is a solution, then it is better than s
- s is modified by side effect

Running this model

In the function One-pass, the model is lauched $\left\lfloor \dfrac{n-w}{step} \right\rfloor$ times.

The function One-Pass is used while the solution is improved.

A fixed point is reached.

Function Several-Passes(w,step)

1: $s \leftarrow$ *initial solution*
2: **while** *One-Pass(w,step)* **do**
3: Pass
4: **end while**
5: **return** s

Running this model

In the function One-pass, the model is lauched $\left\lfloor \dfrac{n-w}{step} \right\rfloor$ times.

The function One-Pass is used while the solution is improved.

A fixed point is reached.

Function Several-Passes(w,step)

```
s ← initial solution
while One-Pass(w,step) do
    Pass
end while
return s
```

Running this model

In the function One-pass, the model is lauched $\left\lfloor \frac{n-w}{step} \right\rfloor$ times.

The function One-Pass is used while the solution is improved.

A fixed point is reached.

Function Several-Passes(w,step)

1: $s \leftarrow$ *initial solution*
2: **while** *One-Pass(w,step)* **do**
3: Pass
4: **end while**
5: **return** s

3 Other neighborhoods

- The Shake model
- The Swap model
- The extended swap model
- The EBSR and EFSR models
- The Random model

A second neighborhood

- In the local search, we considered swapping consecutive jobs
- Now we can allow jobs to change their positions in a limited way
- for each job, new-rank $\in$ [old-rank$-\delta$.. old-rank$+\delta$]
- This will be called the Shake model

A second neighborhood

- In the local search, we considered swapping consecutive jobs
- Now we can allow jobs to change their positions in a limited way
- for each job, new-rank $\in$ [old-rank$-\delta$.. old-rank$+\delta$]
- This will be called the Shake model

A second neighborhood

- In the local search, we considered swapping consecutive jobs
- Now we can allow jobs to change their positions in a limited way
- for each job, new-rank $\in$ [old-rank$-\delta$.. old-rank$+\delta$]
- This will be called the Shake model

A second neighborhood

- In the local search, we considered swapping consecutive jobs
- Now we can allow jobs to change their positions in a limited way
- for each job, new-rank $\in$ [old-rank$-\delta$.. old-rank$+\delta$]
- This will be called the Shake model

Defining a second neighborhood

$s()$

	0	...	δ									$n-1$
0	X	X	X	□	□	□	□	□	□	□	□	□
...	X	X	X	X	□	□	□	□	□	□	□	□
δ	X	X	X	X	X	□	□	□	□	□	□	□
	□	X	X	X	X	X	□	□	□	□	□	□
	□	□	X	X	X	X	X	□	□	□	□	□
	□	□	□	X	X	X	X	X	□	□	□	□
	□	□	□	□	X	X	X	X	X	□	□	□
	□	□	□	□	□	X	X	X	X	X	□	□
	□	□	□	□	□	□	X	X	X	X	X	□
	□	□	□	□	□	□	□	X	X	X	X	X
	□	□	□	□	□	□	□	□	X	X	X	X
$n-1$	□	□	□	□	□	□	□	□	□	X	X	X

The X correspond to possible moves

thus to binary variables.

The ranks of the jobs are recomputed.

Defining a second neighborhood

$s()$

	0	$\cdots$	h	$\cdots$	$h+\delta$					$h+w$		$n-1$
0	□	□	□	□	□	□	□	□	□	□	□	□
$\cdots$	□	□	□	□	□	□	□	□	□	□	□	□
h	□	□	X	X	X	□	□	□	□	□	□	□
$\cdots$	□	□	X	X	X	X	□	□	□	□	□	□
$h+\delta$	□	□	X	X	X	X	X	□	□	□	□	□
	□	□	□	X	X	X	X	X	□	□	□	□
	□	□	□	□	X	X	X	X	X	□	□	□
	□	□	□	□	□	X	X	X	X	□	□	□
	□	□	□	□	□	□	X	X	X	□	□	□
$h+w$	□	□	□	□	□	□	□	□	□	□	□	□
$\cdots$	□	□	□	□	□	□	□	□	□	□	□	□
$n-1$	□	□	□	□	□	□	□	□	□	□	□	□

The X correspond to possible moves

thus to binary variables.

The parameters of the method are:

- δ
- w: width of the interval
- *step*

A third neighborhood

- In the local search, we considered swapping any pairs of two jobs
- Now we can consider jobs within two (small) distinct intervals
- This will be called the Swap model

A third neighborhood

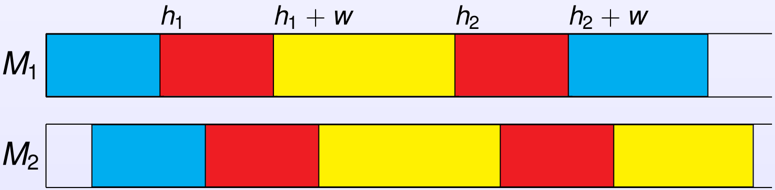
- In the local search, we considered swapping any pairs of two jobs
- Now we can consider jobs within two (small) distinct intervals
- This will be called the Swap model

A third neighborhood

- In the local search, we considered swapping any pairs of two jobs
- Now we can consider jobs within two (small) distinct intervals
- This will be called the Swap model

A third neighborhood

- In the local search, we considered swapping any pairs of two jobs
- Now we can consider jobs within two (small) distinct intervals
- This will be called the Swap model



Running the Swap model

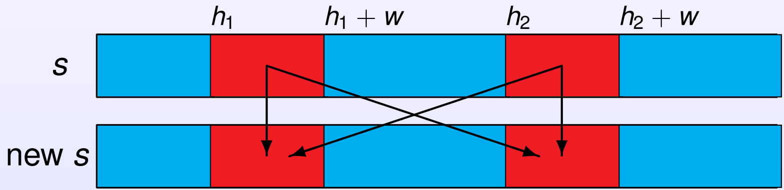
Function One-Pass(w,step)

improved ← **false**
for h_1 **from** 0 **while**$(h_1 + 2w \leq n)$ **by** $step_1$ **do**
    for h_2 **from** $h_1 + w$ **while**$(h_2 + w \leq n)$ **by** $step_2$ **do**
        Run the model
        if (there is a solution *news*) **then**
            $s \leftarrow news$
            improved ← **true**
        end if
    end for
end for
return *improved*

- w: width of the interval
- $step_1, step_2$: pace for the progression of h_1 and h_2
- the model is launched $\approx \frac{(n-2w)^2}{2 step_1 step_2}$ times

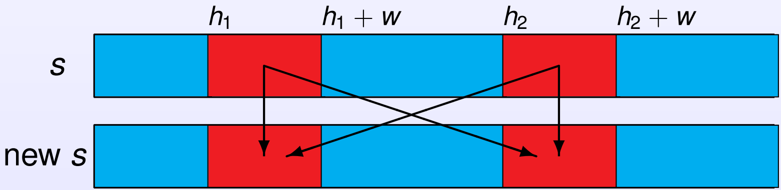
A variant of the Swap model

- In the Swap model, the (blue) jobs with the rank $[1..h_1 - 1]$, $[h_1 + w..h_2 - 1]$ and $[h_2 + w..n]$ do not change
- Only the (red) jobs with the ranks within the intervals $[h_1..h_1 + w - 1]$ and $[h_2..h_2 + w - 1]$ are recomputed.



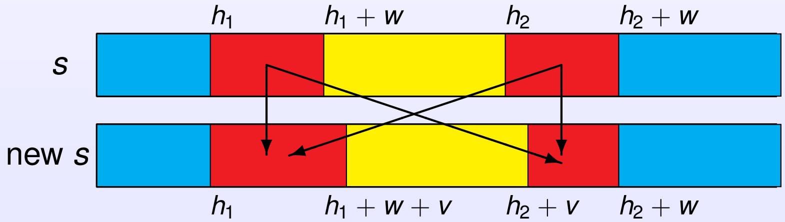
A variant of the Swap model

- In the Swap model, the (blue) jobs with the rank $[1..h_1 - 1]$, $[h_1 + w..h_2 - 1]$ and $[h_2 + w..n]$ do not change
- Only the (red) jobs with the ranks within the intervals $[h_1..h_1 + w - 1]$ and $[h_2..h_2 + w - 1]$ are recomputed.



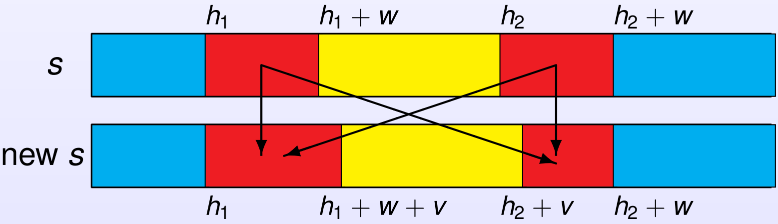
the Extended Swap model

- In the Extended Swap model, the (blue) jobs with the rank $[1..h_1 - 1]$ and $[h_2 + w..n]$ do not change
- The (red) jobs with the ranks within the intervals $[h_1..h_1 + w - 1] \cup [h_2..h_2 + w - 1]$ are recomputed
- within the intervals $[h_1..h_1 + w + v - 1] \cup [h_2 + v..h_2 + w - 1]$ where v is a small positive or negative variable.



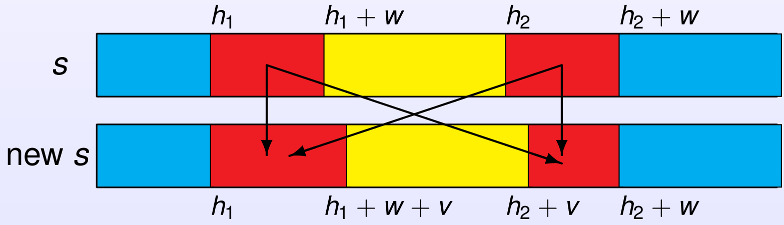
the Extended Swap model

- In the Extended Swap model, the (blue) jobs with the rank $[1..h_1 - 1]$ and $[h_2 + w..n]$ do not change
- The (red) jobs with the ranks within the intervals $[h_1..h_1 + w - 1] \cup [h_2..h_2 + w - 1]$ are recomputed
- within the intervals $[h_1..h_1 + w + v - 1] \cup [h_2 + v..h_2 + w - 1]$ where v is a small positive or negative variable.



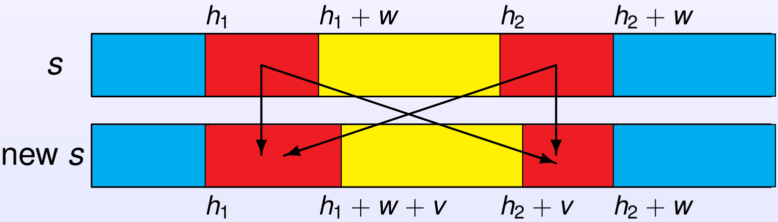
the Extended Swap model

- In the Extended Swap model, the (blue) jobs with the rank $[1..h_1 - 1]$ and $[h_2 + w..n]$ do not change
- The (red) jobs with the ranks within the intervals $[h_1..h_1 + w - 1] \cup [h_2..h_2 + w - 1]$ are recomputed
- within the intervals $[h_1..h_1 + w + v - 1] \cup [h_2 + v..h_2 + w - 1]$ where v is a small positive or negative variable.



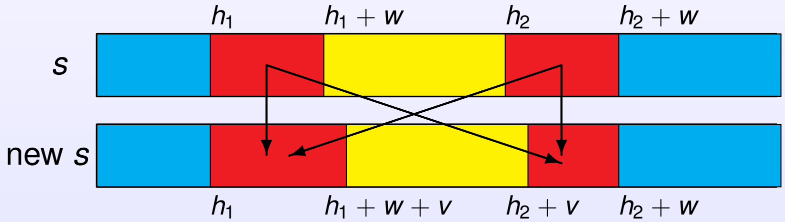
the Extended Swap model

- The subsequence of the (yellow) jobs in the interval $[h_1 + w..h_2 - 1]$ is shifted of v positions
- leftward or rightward, depending on the sign of v
- HV's are to be used in the model to enable this possibility (the price of the free v)



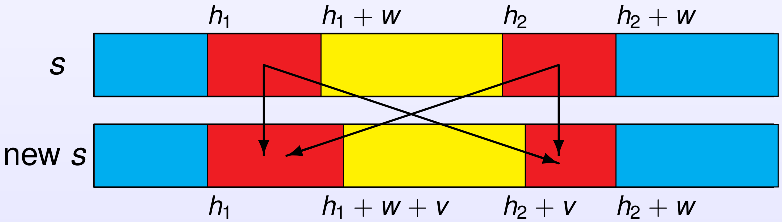
the Extended Swap model

- The subsequence of the (yellow) jobs in the interval $[h_1 + w..h_2 - 1]$ is shifted of v positions
- leftward or rightward, depending on the sign of v
- *HV*'s are to be used in the model to enable this possibility (the price of the free v)



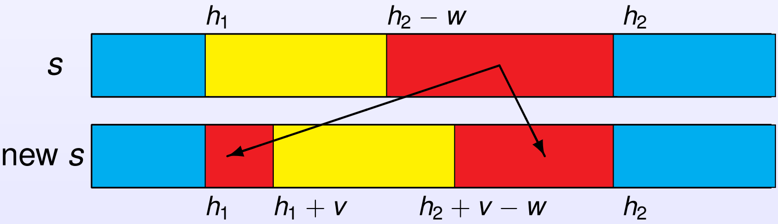
the Extended Swap model

- The subsequence of the (yellow) jobs in the interval $[h_1 + w..h_2 - 1]$ is shifted of v positions
- leftward or rightward, depending on the sign of v
- HV's are to be used in the model to enable this possibility (the price of the free v)



The EBSR model
Extraction and Backward Shifted Reinsertion

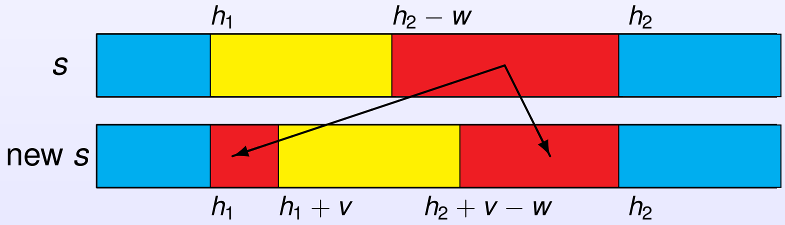
- The subsequence of the (red) jobs in the interval $[h_2 - w..h_2 - 1]$ gives place to the intervals $[h_1..h_1 + v - 1]$ and $[h_2 + v - w..h_2 - 1]$
- The subsequence of the (yellow) jobs in the interval $[h_1..h_2 - w - 1]$ is shifted rightward of v positions



The EBSR model

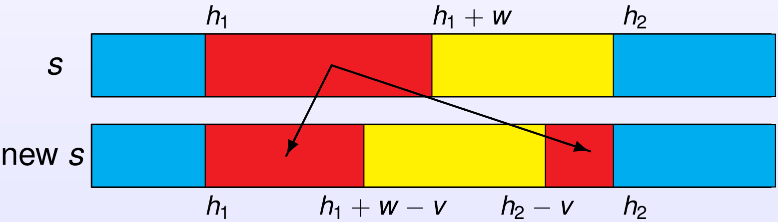
Extraction and Backward Shifted Reinsertion

- The subsequence of the (red) jobs in the interval $[h_2 - w..h_2 - 1]$ gives place to the intervals $[h_1..h_1 + v - 1]$ and $[h_2 + v - w..h_2 - 1]$
- The subsequence of the (yellow) jobs in the interval $[h_1..h_2 - w - 1]$ is shifted rightward of v positions



The EFSR model
Extraction and Forward Shifted Reinsertion

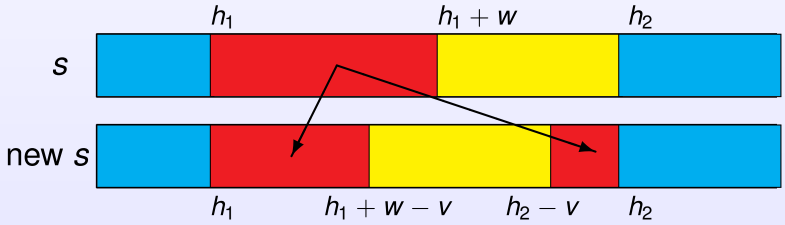
- The subsequence of the (red) jobs in the interval $[h_1..h_1 + w - 1]$ gives place to the intervals $[h_1..h_1 + w - v - 1]$ and $[h_2 - v..h_2 - 1]$
- The subsequence of the (yellow) jobs in the interval $[h_1 + w..h_2 - 1]$ is shifted leftward of $(w - v)$ positions



The EFSR model

Extraction and Forward Shifted Reinsertion

- The subsequence of the (red) jobs in the interval $[h_1..h_1 + w - 1]$ gives place to the intervals $[h_1..h_1 + w - v - 1]$ and $[h_2 - v..h_2 - 1]$
- The subsequence of the (yellow) jobs in the interval $[h_1 + w..h_2 - 1]$ is shifted leftward of $(w - v)$ positions



The Random model

- We choose w jobs
- The other $(n - w)$ jobs keep the same ranks
- We compute the optimal local solution

The Random model

- We choose w jobs
- The other $(n - w)$ jobs keep the same ranks
- We compute the optimal local solution

The Random model

- We choose w jobs
- The other $(n - w)$ jobs keep the same ranks
- We compute the optimal local solution

The Random model

- $\binom{n}{w}$ is huge!
- In practice impossible to enumerate
- We use only a fixed number of randomly generated subsets
- for this method, no fixed point to reach

The Random model

- $\binom{n}{w}$ is huge!
- In practice impossible to enumerate
- We use only a fixed number of randomly generated subsets
- for this method, no fixed point to reach

The Random model

- $\binom{n}{w}$ is huge!
- In practice impossible to enumerate
- We use only a fixed number of randomly generated subsets
- for this method, no fixed point to reach

The Random model

- $\binom{n}{w}$ is huge!
- In practice impossible to enumerate
- We use only a fixed number of randomly generated subsets
- for this method, no fixed point to reach

4 Conclusion

Various Matheuristic methods

1. The “classical” method
2. The Shake method
3. The Swap method
4. The Extended Swap method
5. The EBSR and EFSR methods
6. The Random method

Various Matheuristic methods

1. The “classical” method
2. The Shake method
3. The Swap method
4. The Extended Swap method
5. The EBSR and EFSR methods
6. The Random method

Various Matheuristic methods

1. The “classical” method
2. The Shake method
3. The Swap method
4. The Extended Swap method
5. The EBSR and EFSR methods
6. The Random method

Various Matheuristic methods

1. The “classical” method
2. The Shake method
3. The Swap method
4. The Extended Swap method
5. The EBSR and EFSR methods
6. The Random method

Various Matheuristic methods

1. The “classical” method
2. The Shake method
3. The Swap method
4. The Extended Swap method
5. The EBSR and EFSR methods
6. The Random method

Various Matheuristic methods

1. The “classical” method
2. The Shake method
3. The Swap method
4. The Extended Swap method
5. The EBSR and EFSR methods
6. The Random method

How to compare them?

- How to compare them?
- For each method
- How to choose the parameters?
- How to choose the way to make loops?
- Anyway, let's go!

How to compare them?

- How to compare them?
- For each method
- How to choose the parameters?
- How to choose the way to make loops?
- Anyway, let's go!

How to compare them?

- How to compare them?
- For each method
- How to choose the parameters?
- How to choose the way to make loops?
- Anyway, let's go!

How to compare them?

- How to compare them?
- For each method
- How to choose the parameters?
- How to choose the way to make loops?
- Anyway, let's go!

How to compare them?

- How to compare them?
- For each method
- How to choose the parameters?
- How to choose the way to make loops?
- Anyway, let's go!

What can we conclude from the experiment?
The ILS

- Beware of the *gray cat* theorem!
- EDD versus NEH ?
- Swap, EBSR or EFSR ?
- Compare with Edd23 and Neh23.
- Is it well worth it?

What can we conclude from the experiment?
The ILS

- Beware of the *gray cat* theorem!
- EDD versus NEH ?
- Swap, EBSR or EFSR ?
- Compare with Edd23 and Neh23.
- Is it well worth it?

What can we conclude from the experiment?

The ILS

- Beware of the *gray cat* theorem!
- EDD versus NEH ?
- Swap, EBSR or EFSR ?
- Compare with Edd23 and Neh23.
- Is it well worth it?

What can we conclude from the experiment?

The ILS

- Beware of the *gray cat* theorem!
- EDD versus NEH ?
- Swap, EBSR or EFSR ?
- Compare with Edd23 and Neh23.
- Is it well worth it?

What can we conclude from the experiment?
The ILS

- Beware of the *gray cat* theorem!
- EDD versus NEH ?
- Swap, EBSR or EFSR ?
- Compare with Edd23 and Neh23.
- Is it well worth it?

What can we conclude from the experiment?
The Matheuristics

- Classical, Swap, SwapEx, Shake, EBSR, EFSR, Random?
- Choice of the parameters!
- What should be done now?

What can we conclude from the experiment?

The Matheuristics

- Classical, Swap, SwapEx, Shake, EBSR, EFSR, Random?
- Choice of the parameters!
- What should be done now?

What can we conclude from the experiment?
The Matheuristics

- Classical, Swap, SwapEx, Shake, EBSR, EFSR, Random?
- Choice of the parameters!
- What should be done now?

What can we conclude from the experiment?

The Matheuristics

- Classical, Swap, SwapEx, Shake, EBSR, EFSR, Random?
- Choice of the parameters!
- What should be done now?