Mini-projet

Comparaison de méthodes
exactes et approchées

General VNS

Initialization. Select the set of neighborhood structures $\mathcal{N}_k$, for $k = 1, \ldots, k_{max}$, that will be used in the shaking phase, and the set of neighborhood structures N_ℓ for $\ell = 1, \ldots, \ell_{max}$ that will be used in the local search; find an initial solution x and improve it by using RVNS; choose a stopping condition;

Repeat the following sequence until the stopping condition is met:

(1) Set $k \leftarrow 1$;

(2) *Repeat* the following steps until $k = k_{max}$:

(a) Shaking. Generate a point x' at random from the k^{th} neighborhood $\mathcal{N}_k(x)$ of x;

(b) Local search by VND.

(b1) Set $\ell \leftarrow 1$;

(b2) Repeat the following steps until $\ell = \ell_{max}$;

· *Exploration of neighborhood*. Find the best neighbor x'' of x' in $N_\ell(x')$;

· *Move or not*. If $f(x'') < f(x')$ set $x' \leftarrow x''$ and $\ell \leftarrow 1$; otherwise set $\ell \leftarrow \ell + 1$;

(c) Move or not. If this local optimum is better than the incumbent, move there ($x \leftarrow x''$), and continue the search with $\mathcal{N}_1$ ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$;

RVNS

Initialization.
Select the set of neighborhood structures $\mathcal{N}_k$, for $k = 1, \ldots, k_{\max}$, that will be used in the search; find an initial solution x; choose a stopping condition;
Repeat the following sequence until the stopping condition is met:
(1) Set $k \leftarrow 1$;
(2) *Repeat* the following steps until $k = k_{\max}$:
(a) *Shaking*. Generate a point x' at random from the kth neighborhood of x ($x' \in \mathcal{N}_k(x)$);
(b) *Move or not.* If this point is better than the incumbent, move there ($x \leftarrow x'$), and continue the search with $\mathcal{N}_1$ ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$;

Figure 8.4. Steps of the reduced VNS.

Variable Neighborhood Descent (VND)

Initialization. Select the set of neighborhood structures N_ℓ, for $\ell = 1, \ldots, \ell_{max}$, that will be used in the descent; find an initial solution x (or apply the rules to a given x);

Repeat the following sequence until no improvement is obtained:

(1) Set $\ell \leftarrow 1$;

(2) *Repeat* the following steps until $\ell = \ell_{max}$:

(a) Exploration of neighborhood. Find the best neighbor x' of x ($x' \in N_\ell(x)$);

(b) Move or not. If the solution x' thus obtained is better than x, set $x \leftarrow x'$ and $\ell \leftarrow 1$; otherwise, set $\ell \leftarrow \ell + 1$;

Steepest descent heuristic & First descent heuristic

Initialization.
Choose f, X, neighborhood structure $N(x)$, initial solution x;

Current step (Repeat).
(1) Find $x' = argmin_{x \in N(x)} f(x)$;
(2) If $f(x') < f(x)$ set $x' \leftarrow x''$ and iterate; otherwise, stop.

Steepest descent heuristic

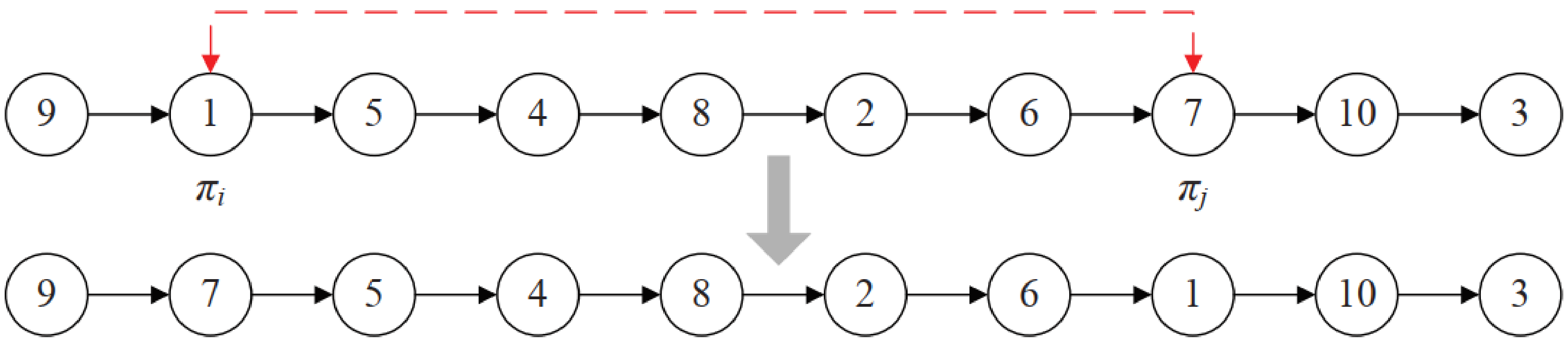
Initialization.
Choose f, X, neighborhood structure $N(x)$, initial solution x;

Current step (Repeat).
(1) Find first solution $x' \in N(x)$;
(2) If $f(x') > f(x)$, find next solution $x'' \in N(x)$; set $x' \leftarrow x''$ and iterate (2); otherwise, set $x \leftarrow x'$ and iterate (1);
(3) If all solutions of $N(x)$ have been considered, stop.

First descent heuristic

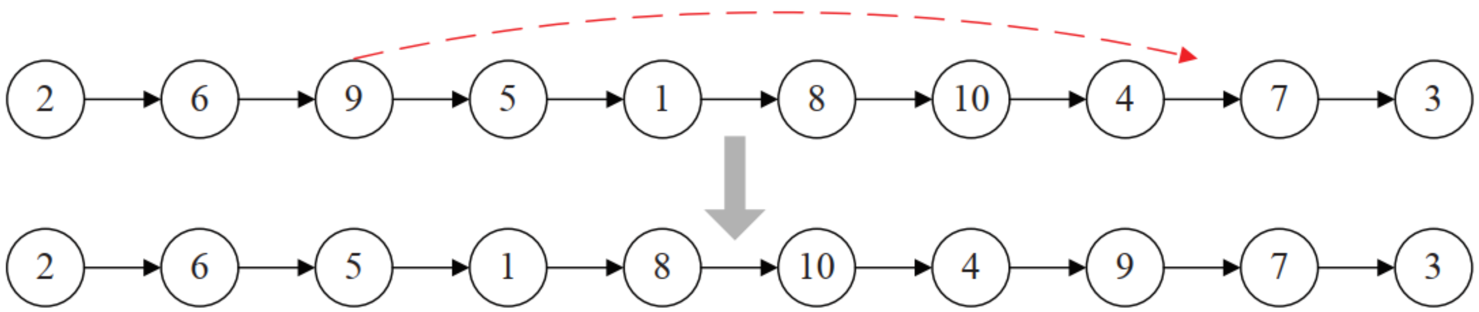
Swap operator

- Swap Operator (N1): The swap operator selects a pair of jobs πi and πj in the current sequence π of jobs, exchanges their positions



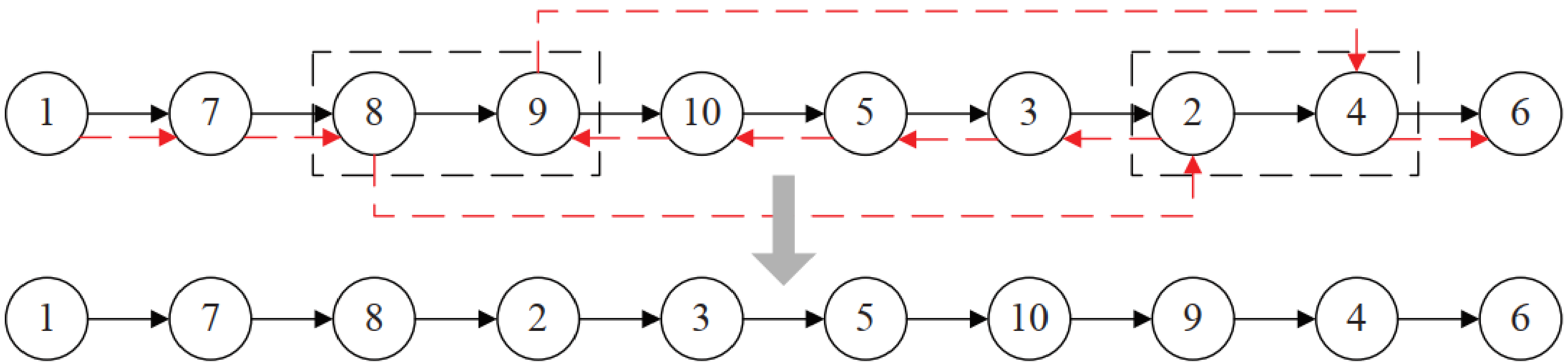
Insertion operator

- Insertion Operator (N2): For a given incumbent arrangement of jobs, the insertion neighborhood can be obtained by removing a job from its position and inserting it into another position.



2-opt operator

- 2-opt Operator (N3): The 2-opt is the most classical heuristic for the traveling salesman problem in which it removes two edges from the tour and reconnects the two paths created.



Voisins de X selon une structure :

Considérons la solution initiale X=(5,1,2,3,4) et les deux structures de voisinages

- **Swap Operator** (N1) : sélectionne une paire de jobs i et j dans la séquence actuelle X,
- **Insertion Operator (N2):** voisin peut être obtenu **en supprimant** un job de sa position et en l'insérant dans une autre position.

Solution X :

5	1	2	3	4

Voisins de *X* selon une structure :

- Solution X :

5	1	2	3	4

Nombre de voisins selon **Swap Operator** (N1)
***O*(*n*²)**

- Voisins selon **Swap Operator** (N1)

1	5	2	3	4
2	1	5	3	4
3	1	2	5	4
4	1	2	3	5
5	2	1	3	4
5	3	2	1	4
5	4	2	3	1
5	1	3	2	4
5	1	4	3	2
5	1	2	4	3

Voisins de *X* selon une structure :

- Solution X :

5	1	2	3	4

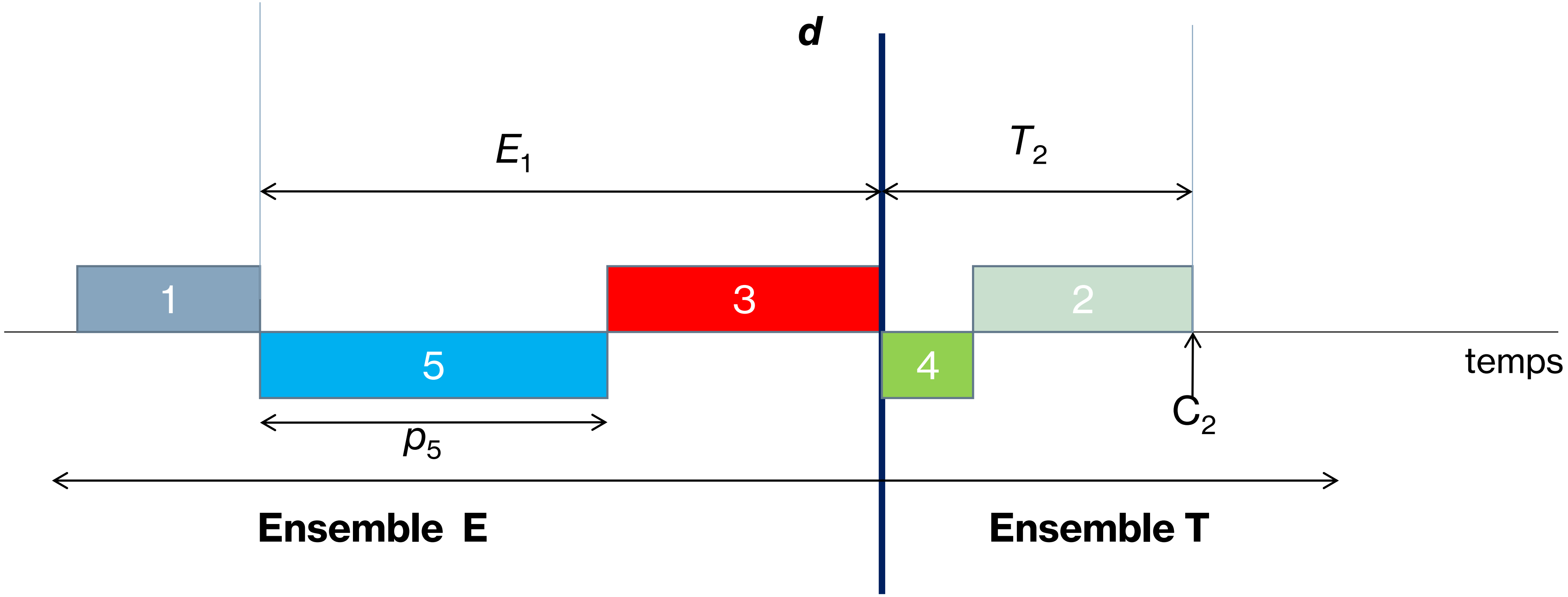
- Voisins selon **Insertion Operator** (N2)

***O*(*n*²)**

1	5	2	3	4
1	2	5	3	4
1	2	3	5	4
1	2	3	4	5
2	5	1	3	4
5	2	1	3	4
5	1	3	2	4
5	1	3	4	2
3	5	1	2	4
5	3	1	2	4
5	1	3	2	4
5	1	2	4	3

4	5	1	2	3
5	4	1	2	3
5	1	4	2	3
5	1	2	4	3

$1||\Sigma\alpha_iE_i+\beta_iT_i$



Problème 1|| ΣU_i

- **Algorithme de Hodgson & Moore**

1) $S = \{\}$;

2) Considérer les tâches dans l'ordre EDD

3) Ajouter la tâche i à S

4) Si la tache i est en retard

5) Alors supprimer de S la plus grande tâche

6) $i \leftarrow i + 1$;

7) Si $i < n$ aller a l'étape 3

L'algorithme de Hodgson & Moore est optimal pour Problème 1|| ΣU_i
Sa complexite est O($n$logn).