Algorithme génétique: application au problème du voyageur de commerce

Mathieu Mandret

M3202 - Modélisation

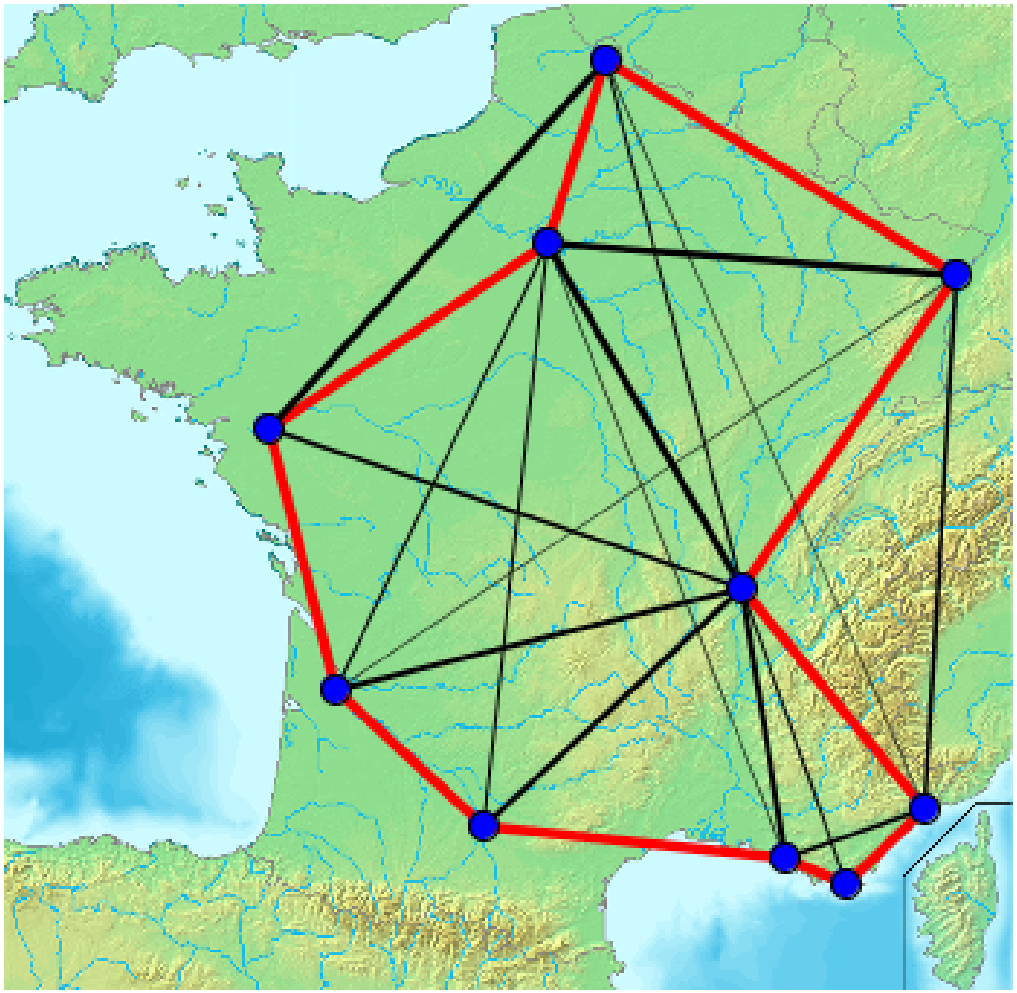
7 décembre 2017

Plan

Le problème du voyageur de commerce

- n villes.
- Les parcourir une fois.
- Revenir au point de départ.
- Chemin le plus court ?

Le problème du voyageur de commerce



Approche déterministe

Permutations de n éléments :

$$\begin{aligned} n &= 10 \\ n! &= 3628880 \end{aligned}$$

Approche déterministe

Permutations de n éléments :

$$\begin{aligned} n &= 30 \\ n! &\simeq 2.65 \times 10^{32} \end{aligned}$$

Principe général

- Solution approchée
- Inspiré de la théorie de l'évolution
- Sélection naturelle

Déroulement

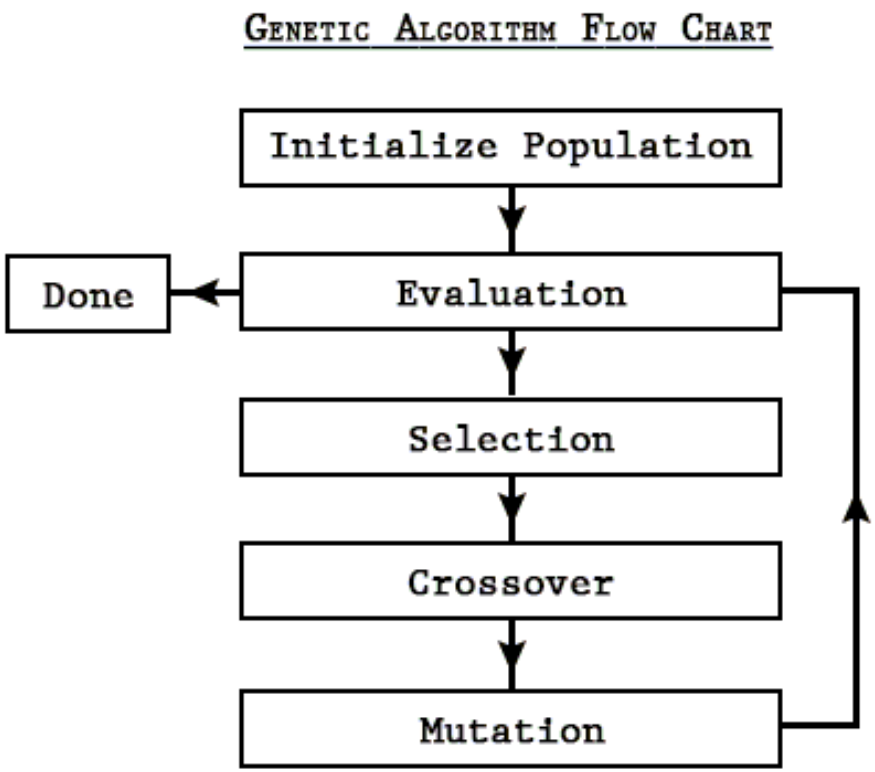


Figure 2

1. Générer population

Déroulement

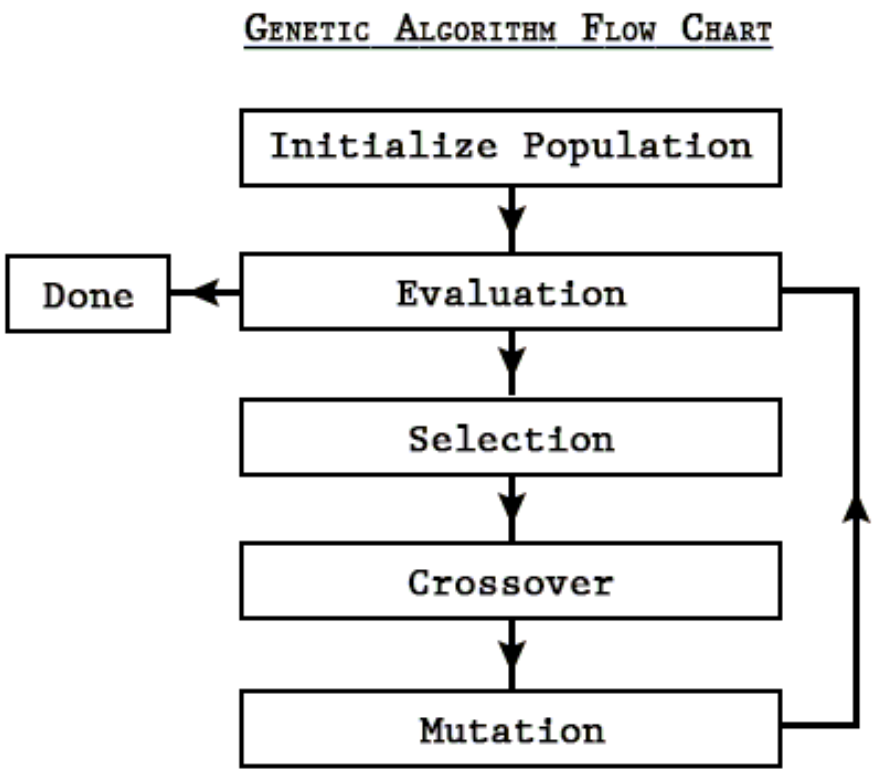


Figure 2

1. Générer population
2. Calculer toutes les fitness

Déroulement

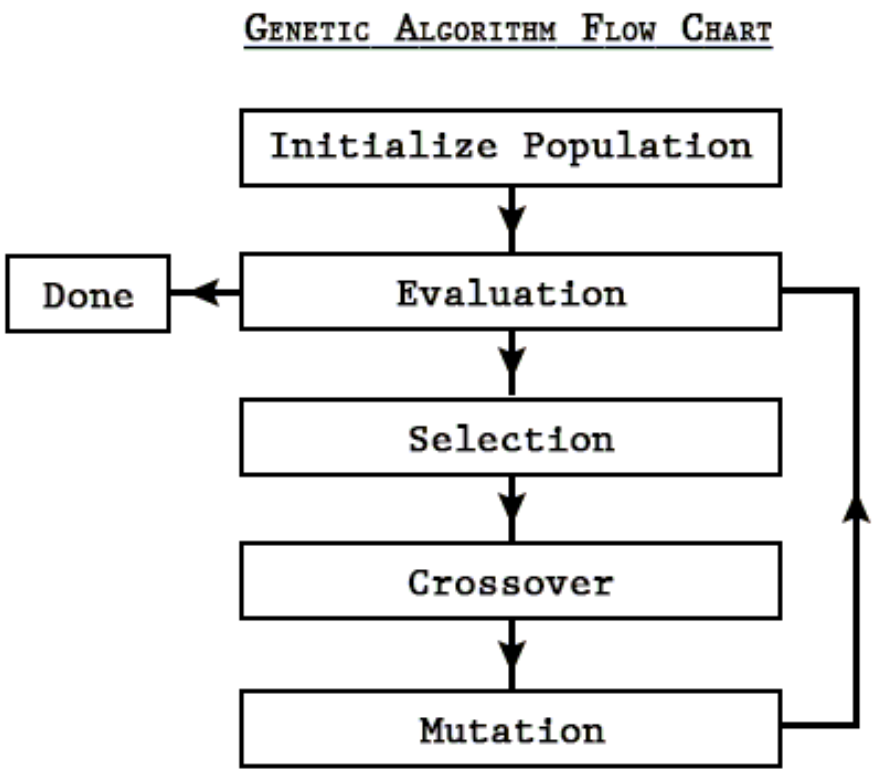


Figure 2

1. Générer population
2. Calculer toutes les fitness
3. Sélectionner les parents

Déroulement

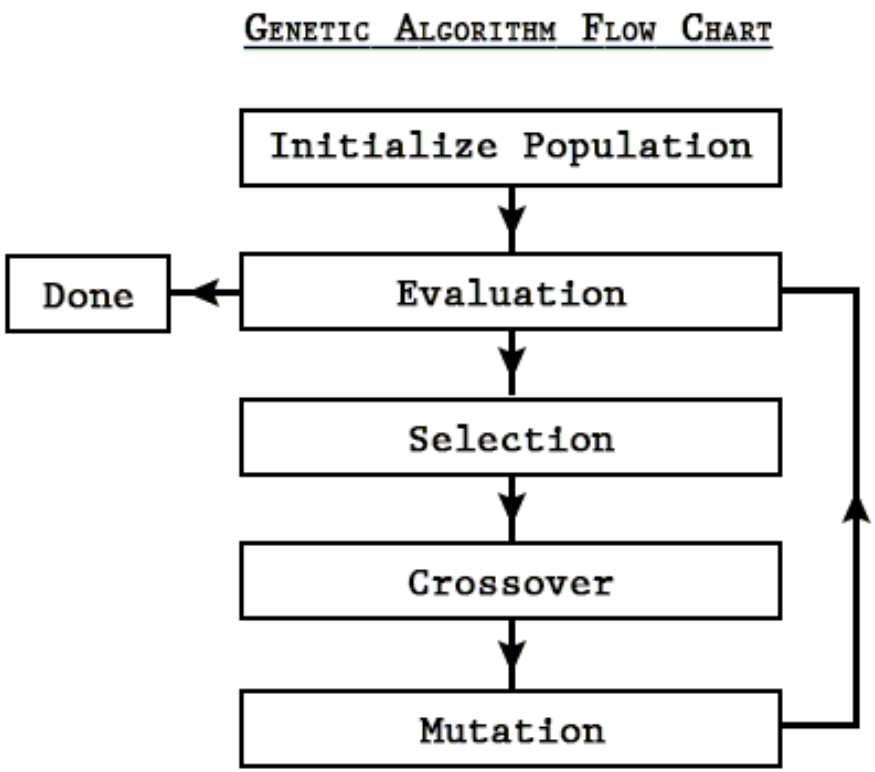


Figure 2

1. Générer population
2. Calculer toutes les fitness
3. Sélectionner les parents
4. Les croiser

Déroulement

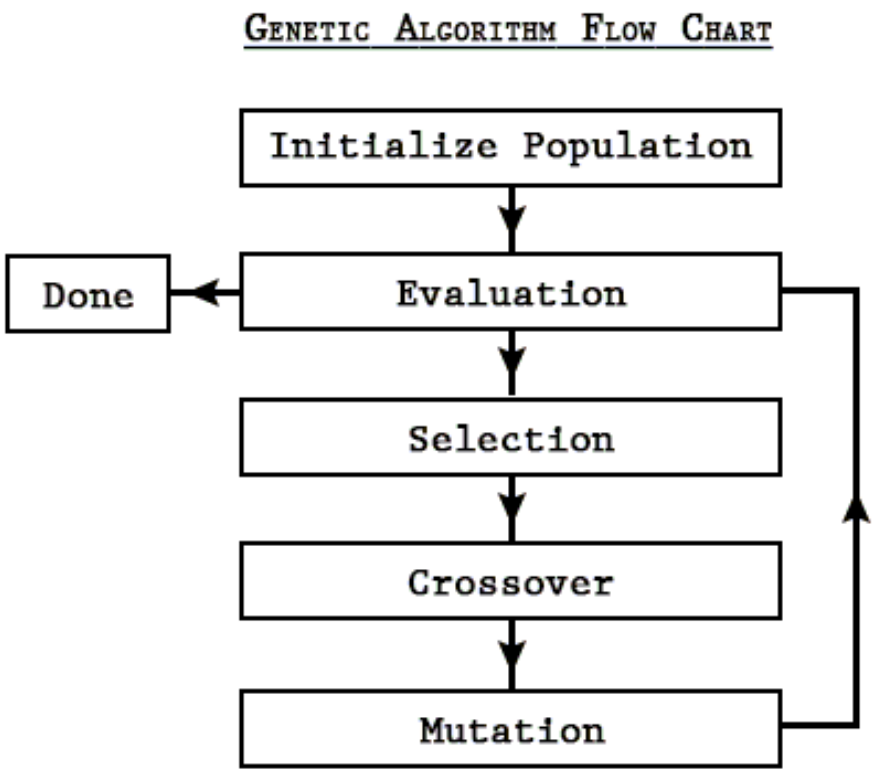


FIGURE 2

1. Générer population
2. Calculer toutes les fitness
3. Sélectionner les parents
4. Les croiser
5. Faire muter les fils

Déroulement

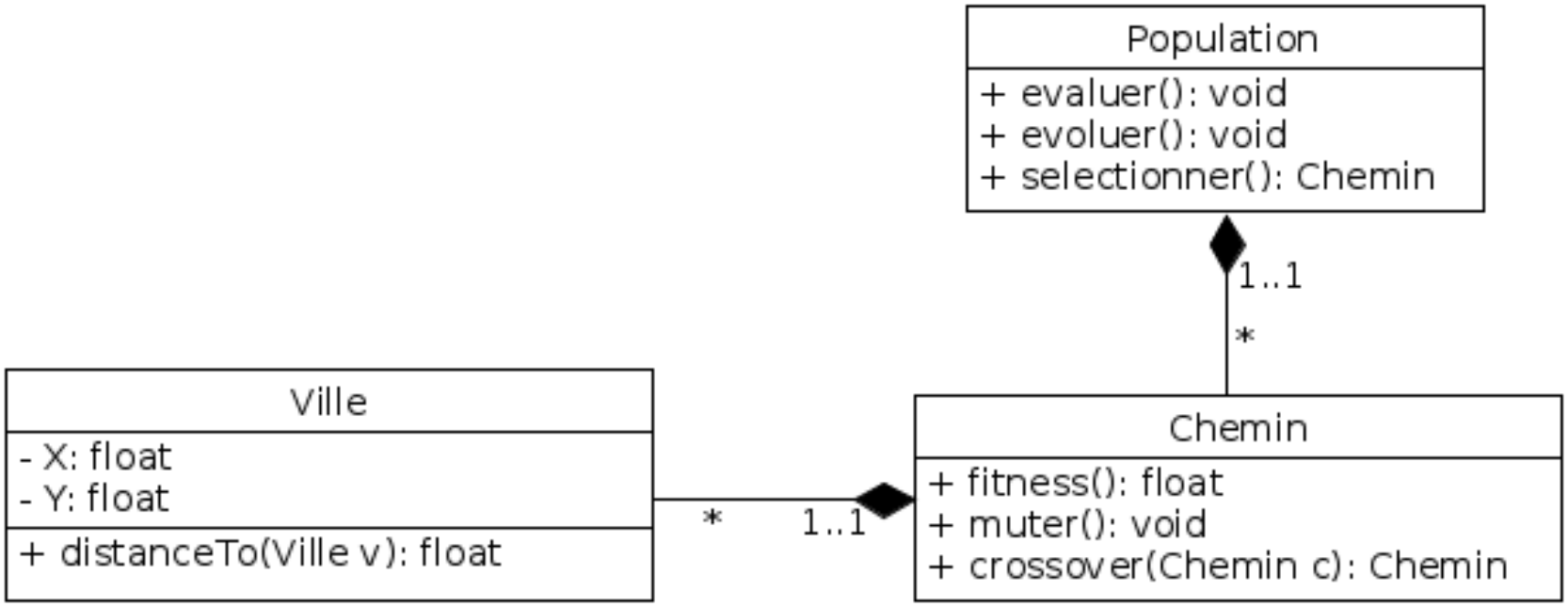
Boucle $\rightarrow$ condition d'arrêt :

- Nombre de générations
- Fitness cible

Application au problème du voyageur du commerce

Individu	$\rightarrow$	Chemin
Population	$\rightarrow$	Ensemble de chemins
Fitness	$\rightarrow$	Longueur du chemin

Architecture orientée objet



Architecture orientée objet

Ville

Coordonnées (X,Y)
Distance par rapport à une autre ville

Ville

Distance entre 2 villes

$$||\overrightarrow{AB}|| = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}$$

Architecture orientée objet

Chemin

- Liste ordonnée de villes
- Calcul de la fitness
- Croisement
- Mutation

Chemin

Fitness d'un chemin

$$\sum_{i=0}^{n-2} ||\overrightarrow{V_i V_{i+1}}||$$

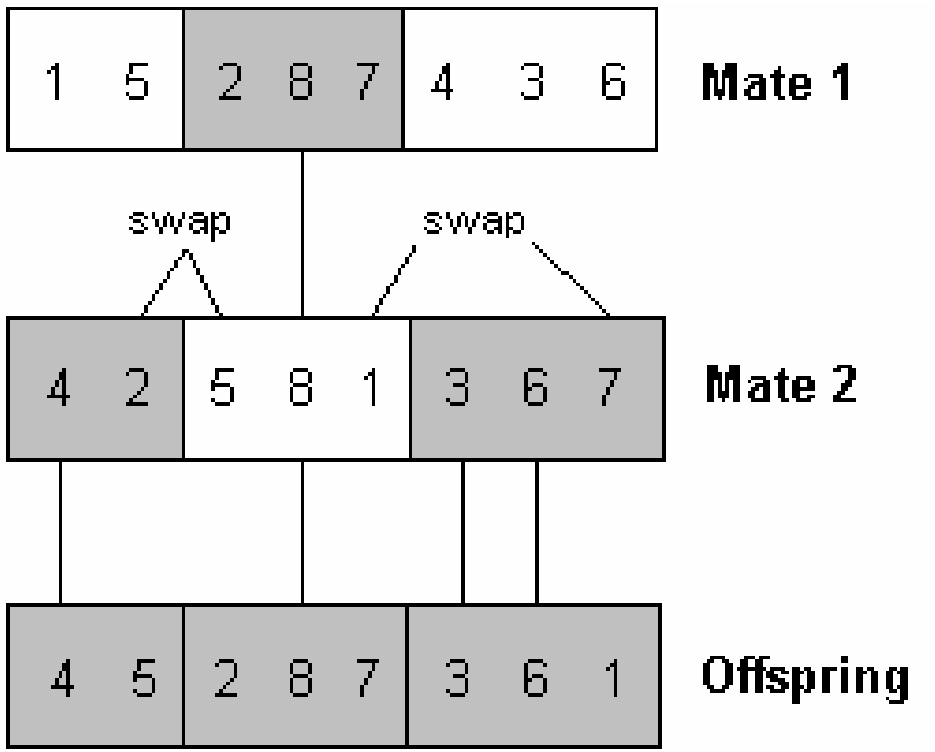
Chemin

Partially Matched Crossover

1. Prendre x villes du parent 1
2. Les mettre dans les positions correspondantes dans le fils
3. Remplir le reste avec des villes du parent 2

Chemin

Partially Matched Crossover



Chemin

Mutation

Échanger la position de 2 villes

Architecture orientée objet

Population

- Initialisation
- Selection

Population

Initialisation

Chemin uniques $\rightarrow$ Diversité
Opérateur *not in*

Population

Initialisation

Chemin uniques $\rightarrow$ Diversité
Opérateur *not in*
Meilleur chemin

Population

Initialisation

Chemin uniques $\rightarrow$ Diversité
Opérateur *not in*
Meilleur chemin $\rightarrow$ None

Population

Initialisation

Chemin uniques $\rightarrow$ Diversité
Opérateur *not in*
Meilleur chemin $\rightarrow$ None
Meilleure longueur

Population

Initialisation

Chemin uniques $\rightarrow$ Diversité
Opérateur *not in*
Meilleur chemin $\rightarrow$ None
Meilleure longueur $\rightarrow$ *math.inf*

Population

Sélection

Tournoi

1. Prendre x individus au hasard
2. Sélectionner le meilleur

Population

Sélection

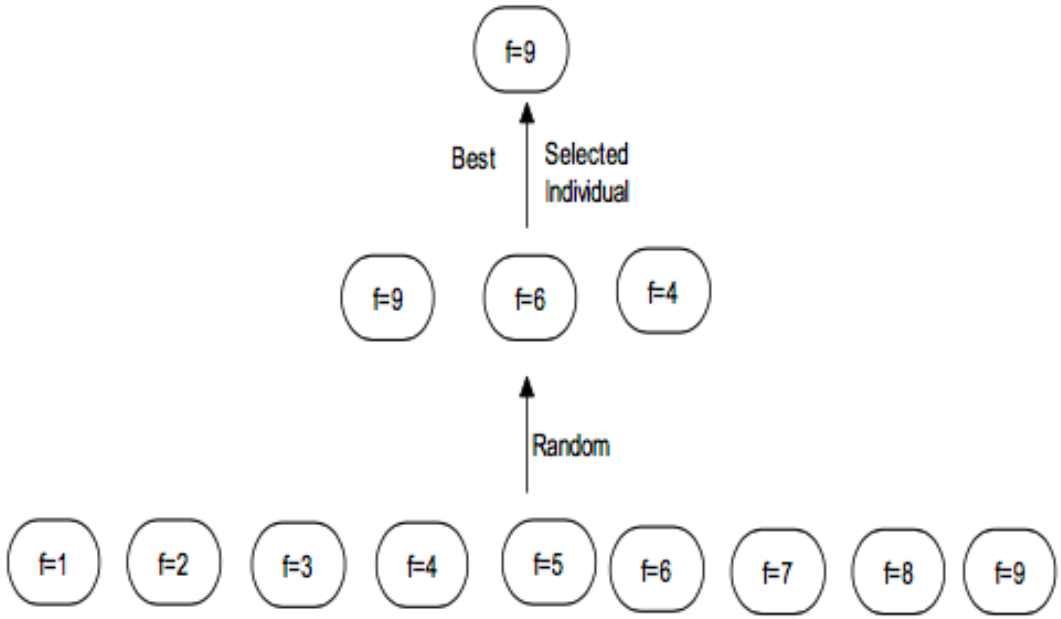


Fig. 3. Tournament Selection Mechanism [15]

Population

Évaluation : Complexité

Recalculer la longueur de tous les chemins

Population

Évaluation : Complexité

Recalculer la longueur de tous les chemins

Utiliser un cache

Population

Évaluation : Cache

$$[\text{Chemin}] \Rightarrow \text{fitness}$$

Population

Évaluation : Cache

$$[\text{Chemin}] \Rightarrow \text{fitness}$$

Si chemin $\in$ clés récupérer **sinon** calculer et mettre en cache.

Mesurer l'efficacité

Pourcentage d'amélioration

Mesure à quel point les chemins sont devenus meilleurs sur n générations

Mesurer l'efficacité

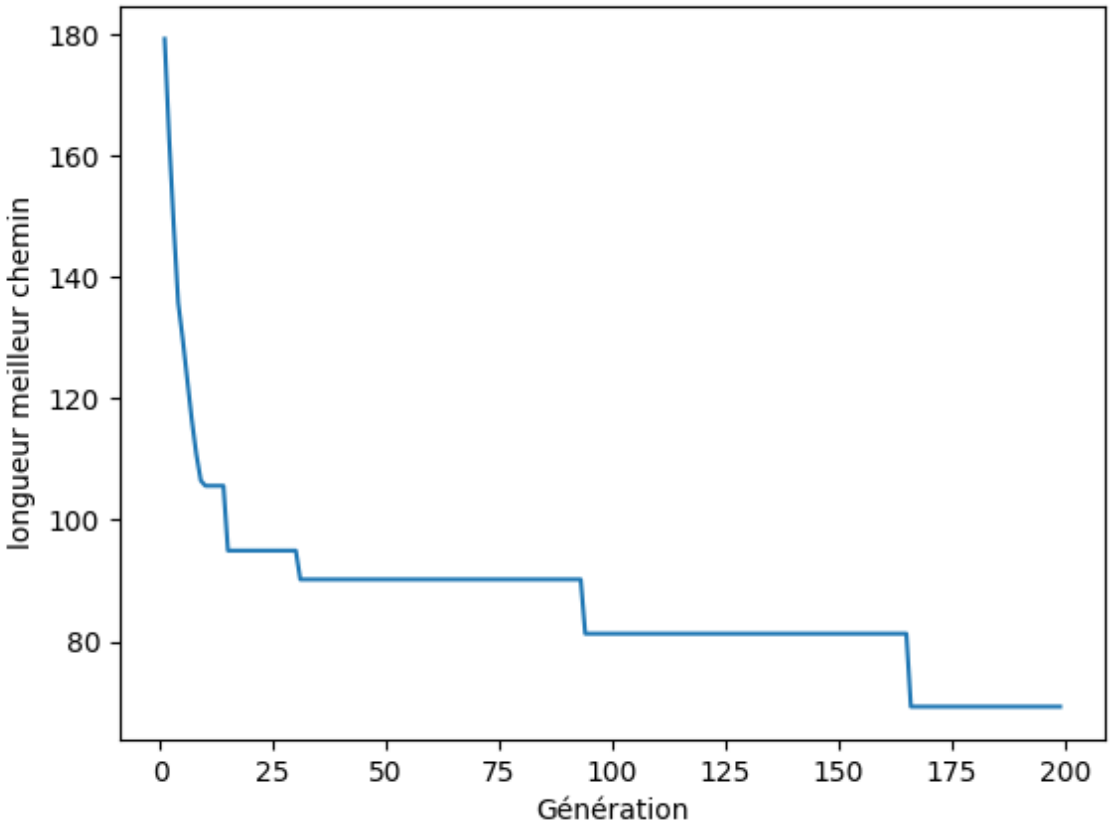
Pourcentage d'amélioration

Mesure à quel point les chemins sont devenus meilleurs sur n générations

$$\frac{\text{Meilleure fitness}_n - \text{Meilleure fitness}_1}{\text{Meilleure fitness}_1} \times 100$$

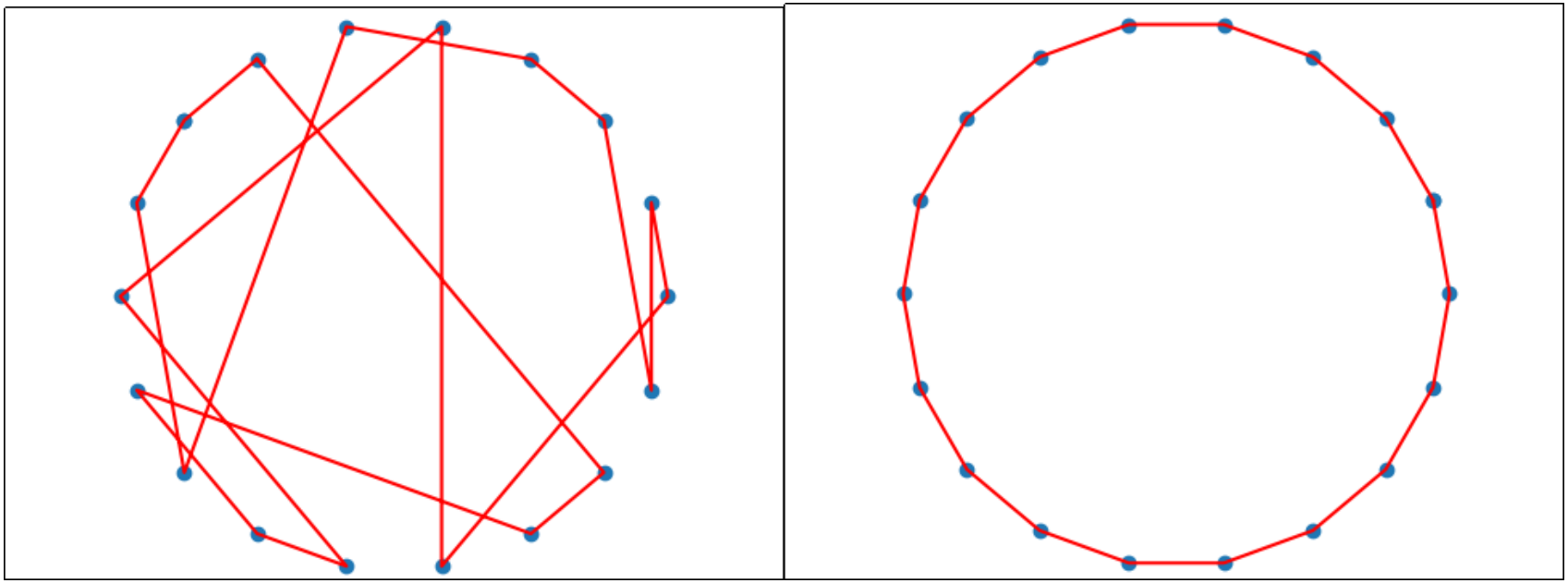
Mesurer l'efficacité

Pourcentage d'amélioration



Mesurer l'efficacité

Carte



Conclusion

Conclusion

Introduction aux algorithmes génétiques, pratique POO en Python

Conclusion

Introduction aux algorithmes génétiques, pratique POO en Python
Difficultés : Roulette, tests.