HMIN233 - Algorithmes d'exploration et de mouvement

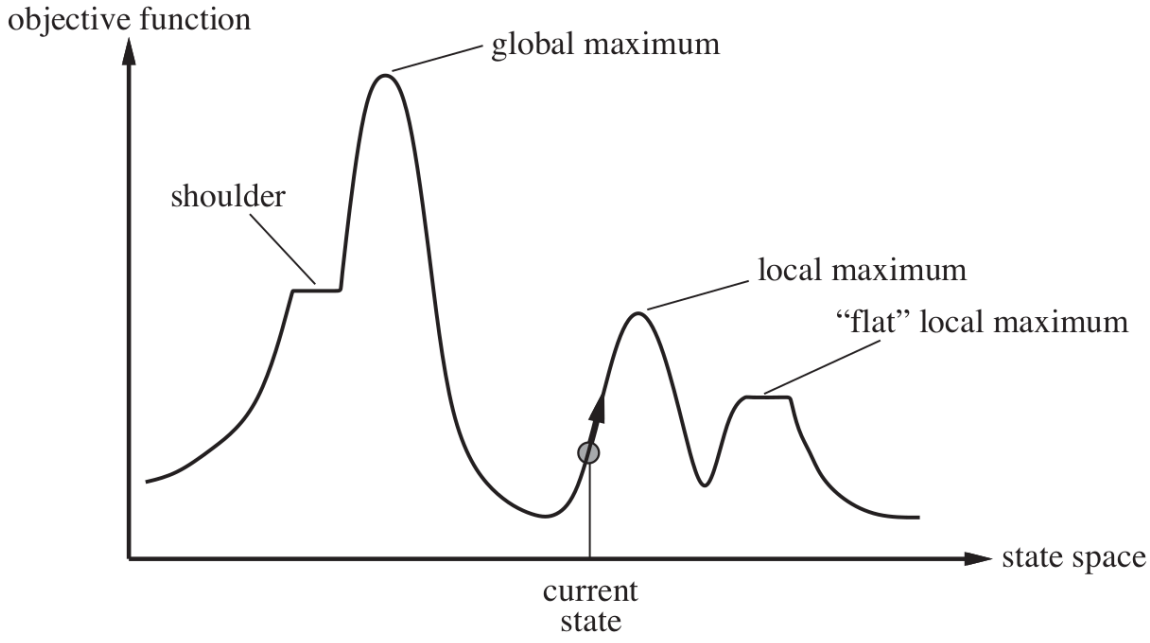
Algorithmes génétique

Suro François

Université de Montpellier
Laboratoire d'informatique, de robotique
et de microélectronique de Montpellier

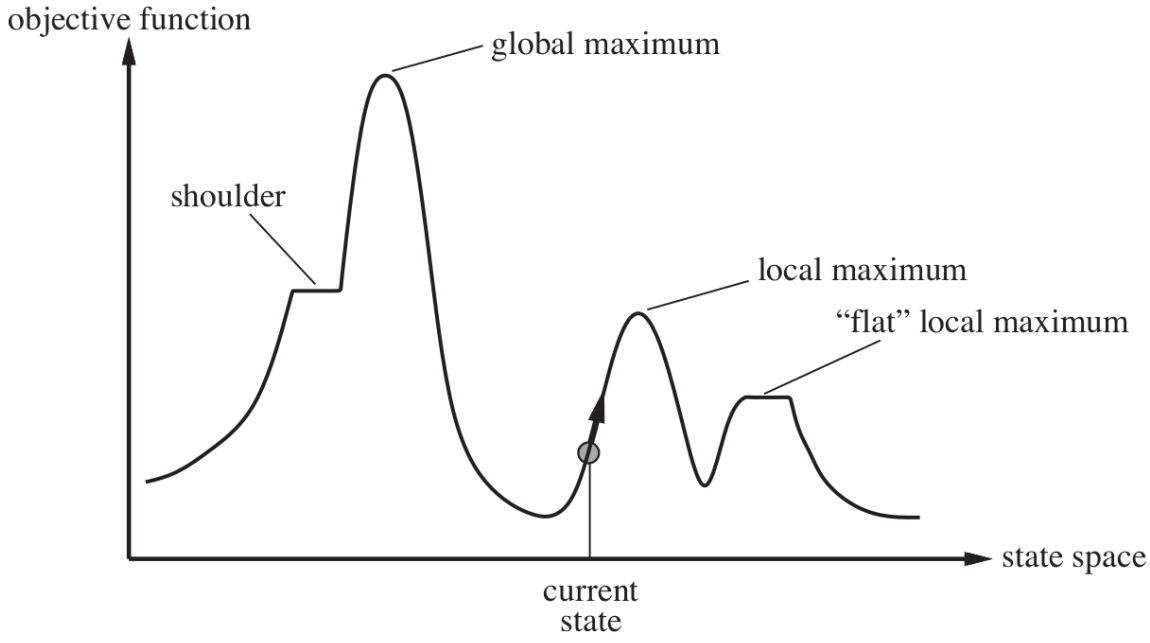
Janvier 2021

Recherche Locale



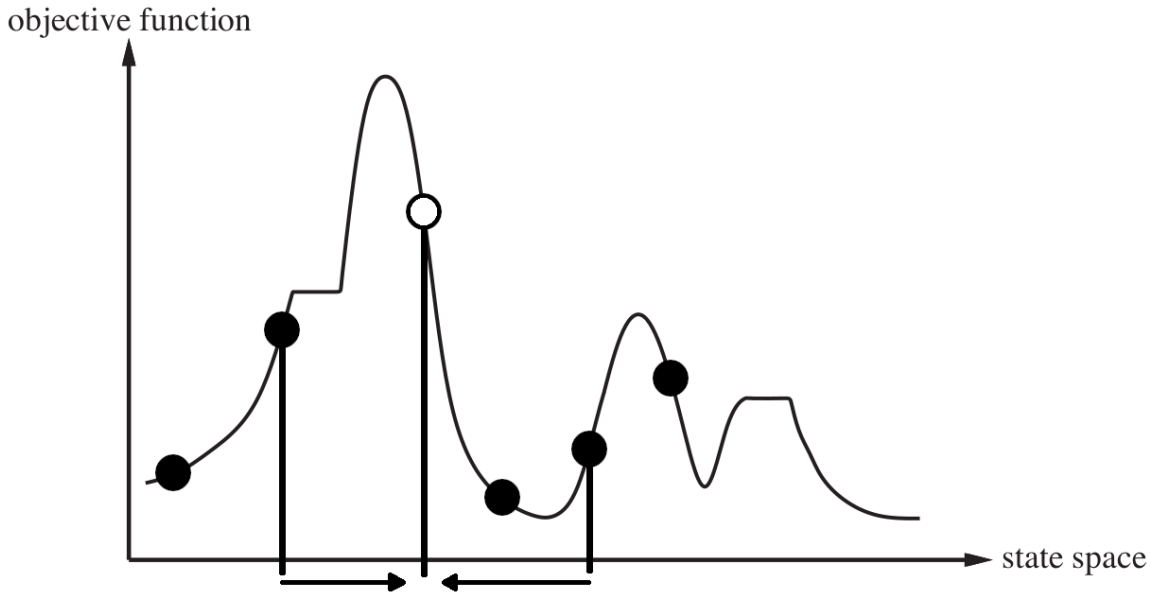
- Une solution est décrite par un état faisant parti de l'espace des état (toutes les configurations possibles d'un état).
- A partir d'un état connu, on cherche une solution "autour" (local) de cet état qui améliore le résultat.

Recherche Locale



- Hill-climbing - "escalade".
- Simulated annealing - recuit simulé.
- Local beam search - recherche en faisceau.
- Genetic/Evolutionary algorithm - génétique/évolutionniste.

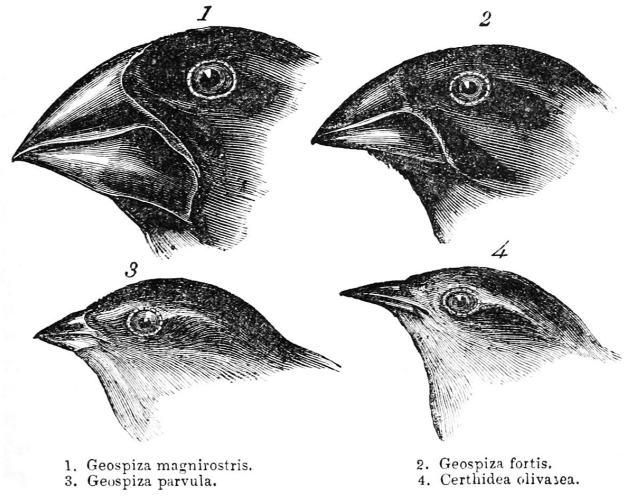
Algorithme génétique



- Au lieu de chercher une solution autour d'un seul état, on va chercher en composant une solution à partir de deux états (bons, de préférence).

Évolution - adaptation

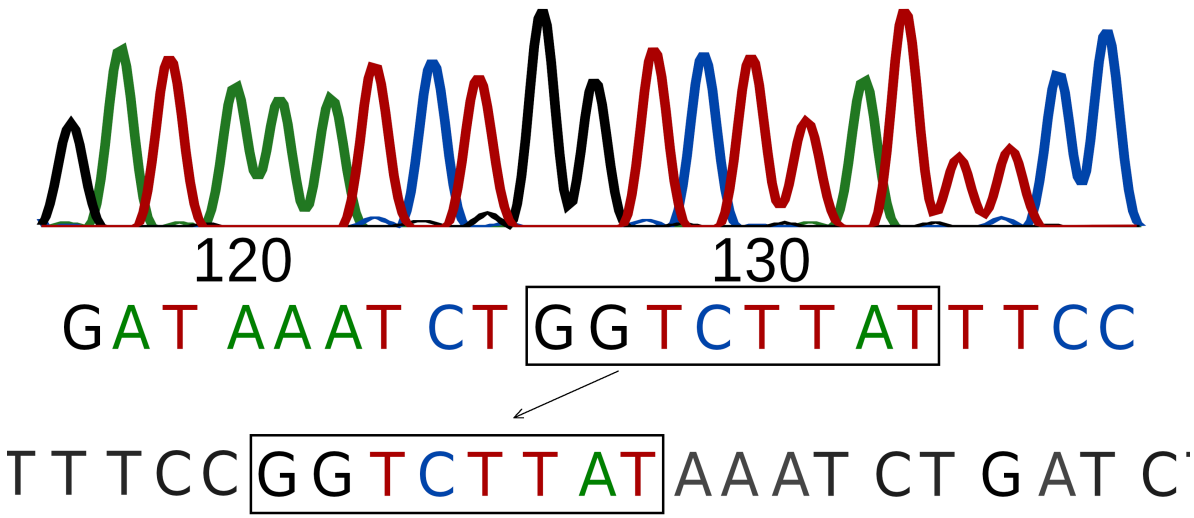
- Charles Darwin (1859) / Alfred Russel Wallace (1858).
- Les variations qui apparaissent lors de la reproduction sont conservées par les générations suivantes proportionnellement à leur impact sur l'opportunité de reproduction.



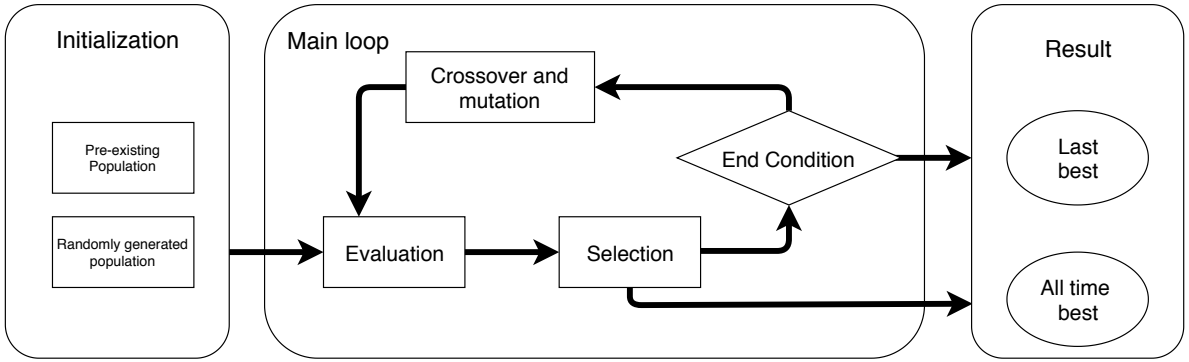
Évolution - reproduction

Les variations se produisent de deux manières :

- Mutation: un des élément du code génétique est remplacé.
- Croisement ("crossover"): le nouvel individu est créé en recopiant de **longues séquences** du code génétique des parents.

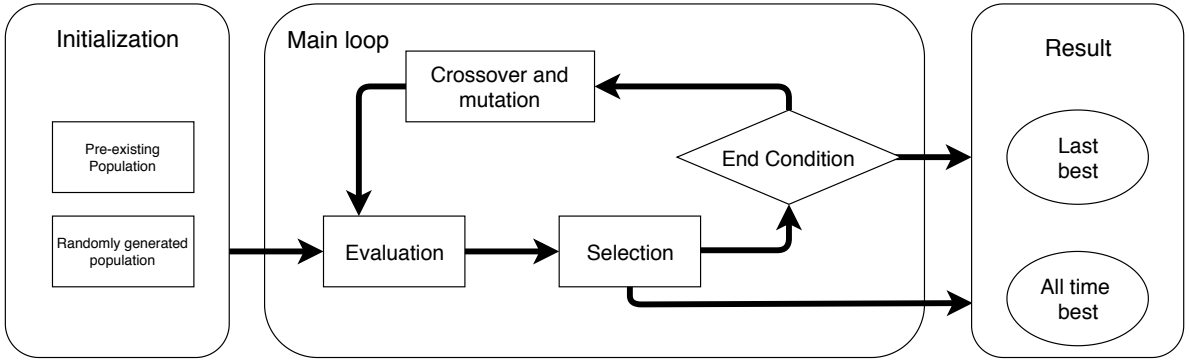


L'algorithme



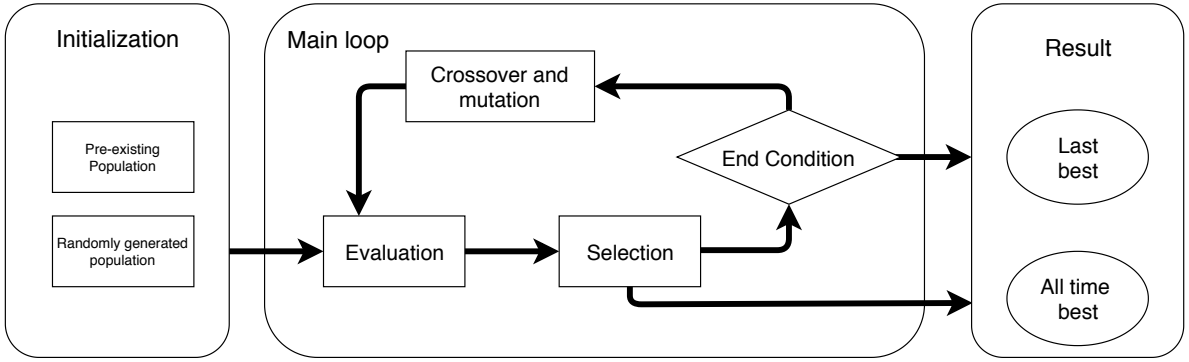
- Initialisation.
- Boucle:
  - Évaluation.
  - Sélection.
  - Condition d'arrêt.
  - Génération d'une nouvelle population.
- Résultat.

L'algorithme - Initialisation



- Population générée aléatoirement.
- Population générée aléatoirement, avec certaines contraintes ...
- Population initialisée par d'autres méthodes.
- Population préexistante.

L'algorithme - Itération



- **Évaluation.**
- Sélection.
- Conditions d'arrêt.
- Génération d'une nouvelle population: croisement et mutation.

L'évaluation

Fonction de Fitness

Mesure de l'adaptation de l'individu à son environnement et ses contraintes.

- Fonction "objectif"

Par exemple: mesure de l'erreur

Cette mesure est relative

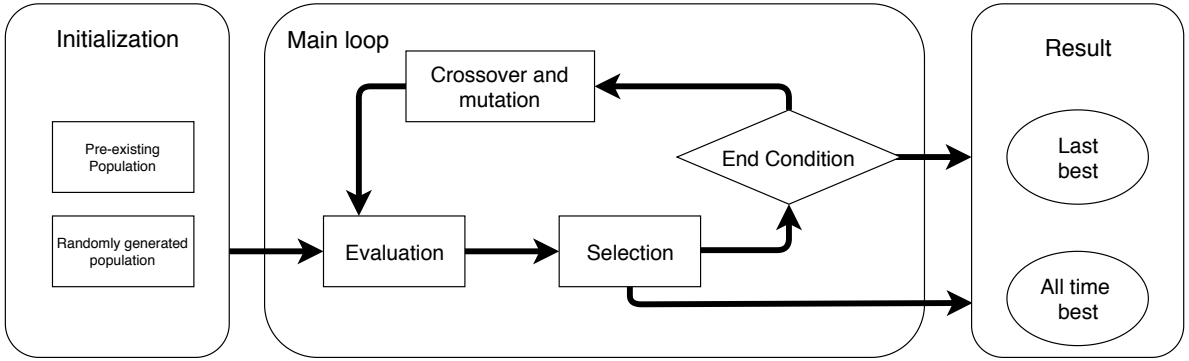
Ce n'est pas nécessairement une mesure de l'erreur par rapport à un résultat idéal (absolu), mais de la qualité d'un individu par rapport aux autres (un ordre, relatif) qui va permettre la sélection.

L'évaluation

L'évaluation peut être:

- Le calcul d'un résultat pour l'individu et l'attribution d'un score par la fonction de fitness
- Le calcul de plusieurs résultats pour des données différentes, et un score moyen.
- D'autres types d'évaluations complexes ...

L'algorithme - Itération



- Évaluation.
- **Sélection.**
- Conditions d'arrêt.
- Génération d'une nouvelle population: croisement et mutation.

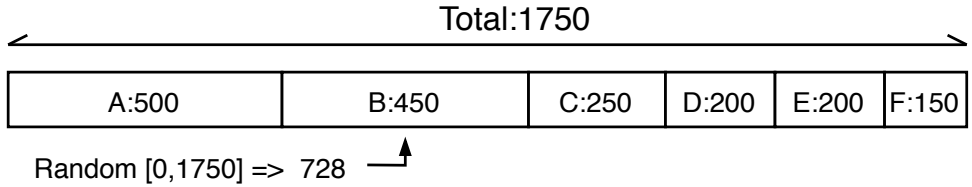
Sélection

Pour garder une population de taille constante et conserver les bons "traits" il ne faut reproduire qu'une partie de la population.

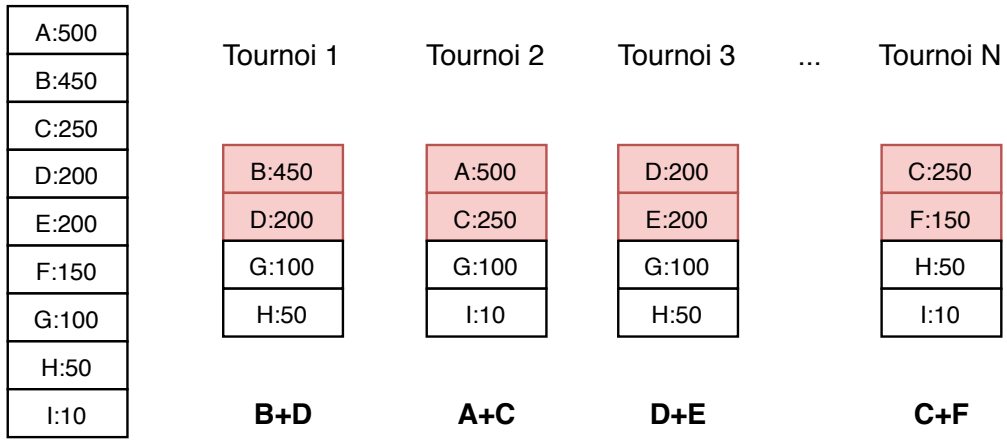
- Sélection stricte: on ne conserve qu'un petit nombre d'individus. En éliminant rapidement les moins bons on converge plus vite. En revanche la population est peu diverse.
- Sélection large: on conserve un grand nombre d'individus. La population générée est très diverse ce qui permet d'explorer plus de configurations. En revanche on converge moins vite.

Sélection

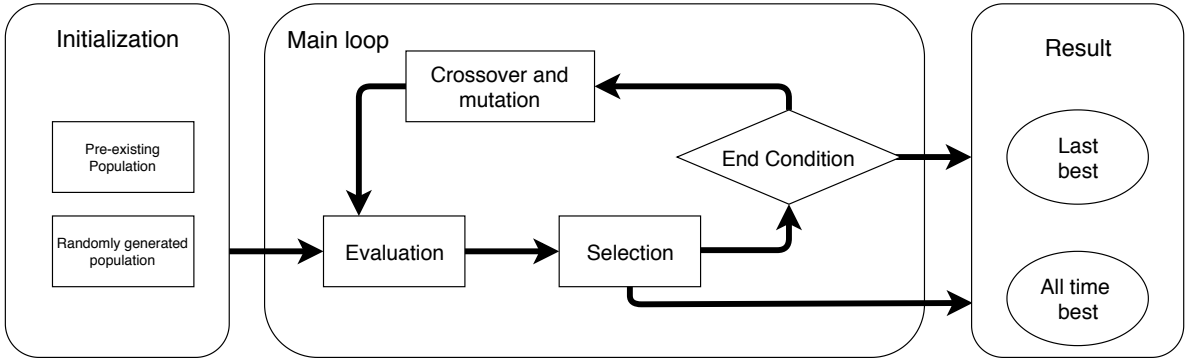
- **Troncation** : conserver qu'une partie de la population, par exemple 30%. On croise aléatoirement parmi cette sélection.
- **Proportionnelle** : sélection aléatoire proportionnelle au score.



- **Tournois** : sélection aléatoire d'un sous groupe (tournoi), reproduction des 2 meilleurs.



L'algorithme - Itération

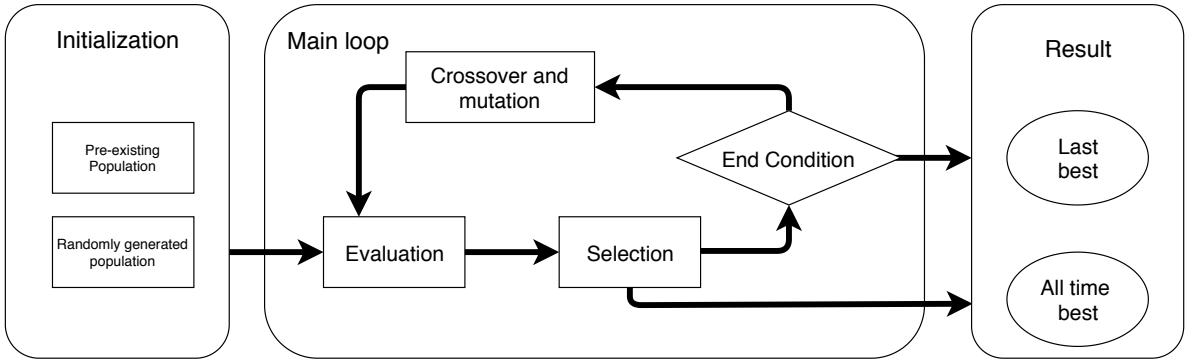


- Évaluation.
- Sélection.
- **Conditions d'arrêt.**
- Génération d'une nouvelle population: croisement et mutation.

Conditions d'arrêt

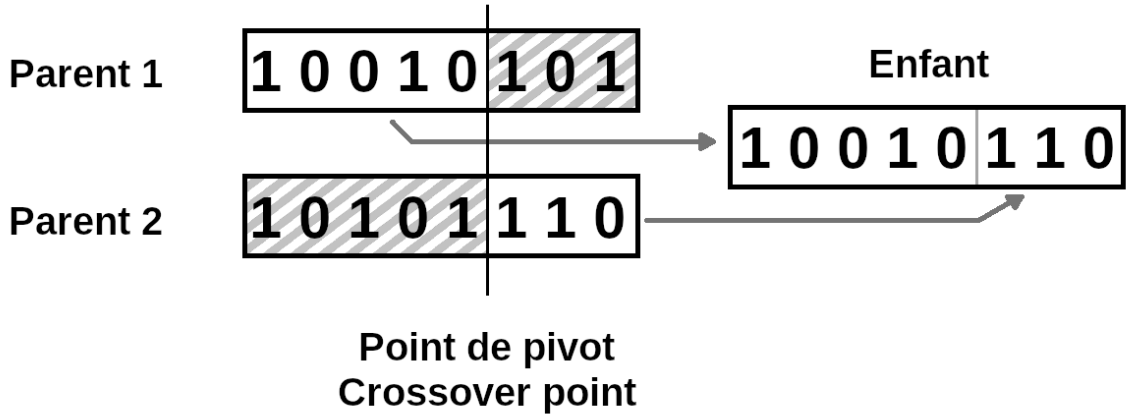
- Un nombre de générations donné.
  $\rightarrow$ *et si le meilleur résultat arrive à n+1 ?*
- Un score atteint.
  $\rightarrow$ *il faut connaître le score à atteindre/s'il est possible de l'atteindre.*
- Observation d'un résultat satisfaisant.
  $\rightarrow$ *a vue de nez ...*
- Stagnation : pas d'évolution du score.
  $\rightarrow$ *combien de générations constitue une stagnation ?*
- Jamais ?

L'algorithme - Itération



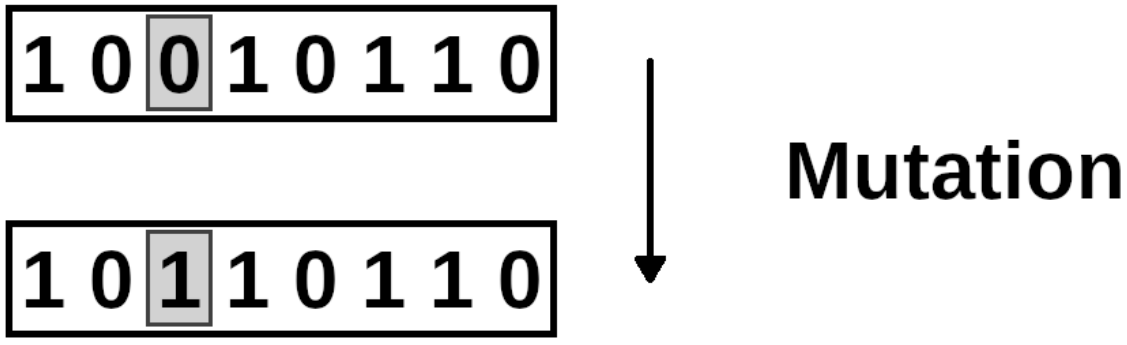
- Évaluation.
- Sélection.
- Conditions d'arrêt.
- **Génération d'une nouvelle population**: croisement et mutation.

Croisement (Crossover)



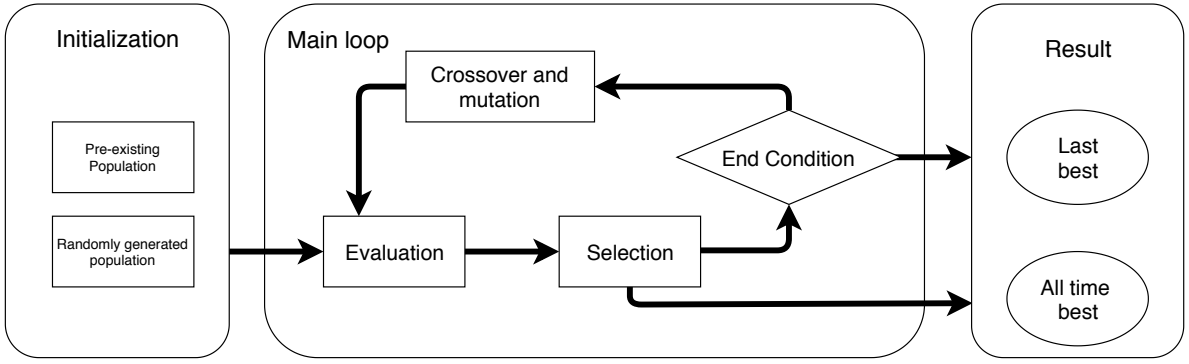
- Choisir un point de pivot (crossover point) aléatoirement dans le génome.
- Recopier dans l'enfant le premier génome jusqu'au point de pivot, puis compléter avec le deuxième génome.

Mutation



- Remplacer un des éléments du génome par un autre élément (valide).
- Il faut définir la probabilité d'une ou plusieurs mutations.
- Dans certains cas il est possible de limiter l'intervalle de cette mutation ...

L'algorithme - Résultat



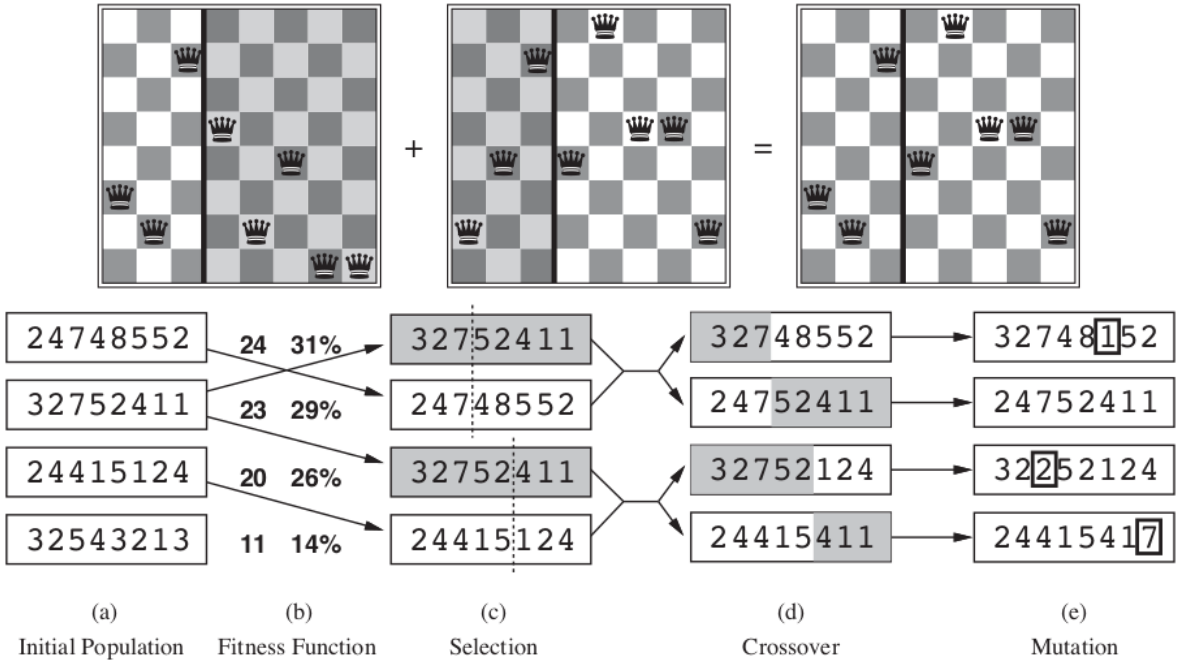
Le meilleur (score) de la dernière génération.

- "L'esprit" de l'algorithme génétique: l'évolution de la population suit l'évolution des contraintes.

Le meilleur (score) de toutes les générations.

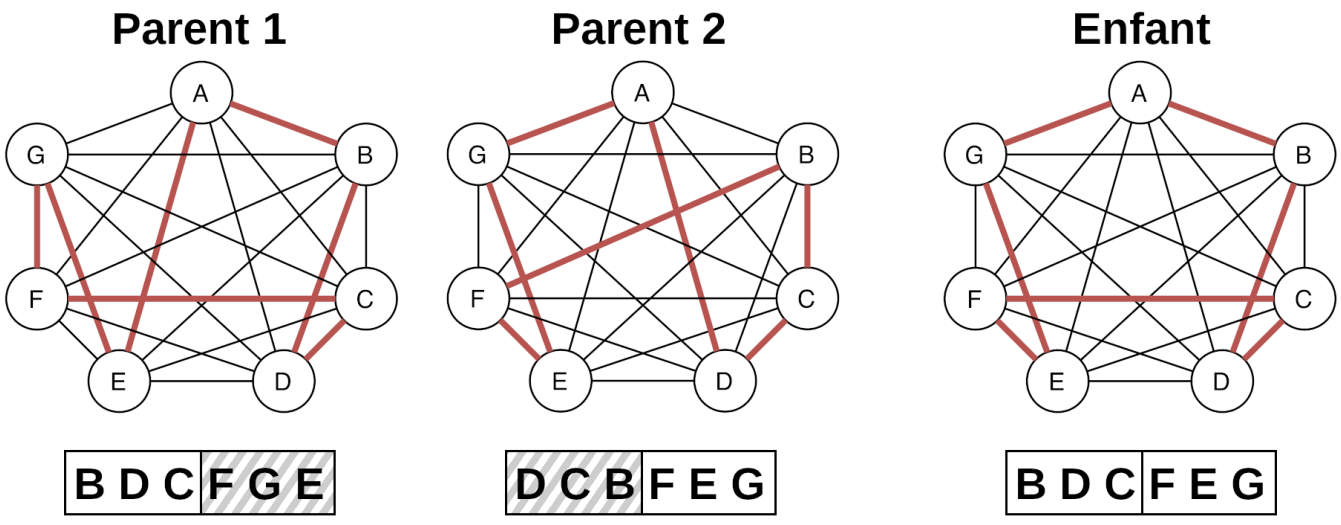
- La définition du problème est fixe.

Application: 8 reines (n-reines)



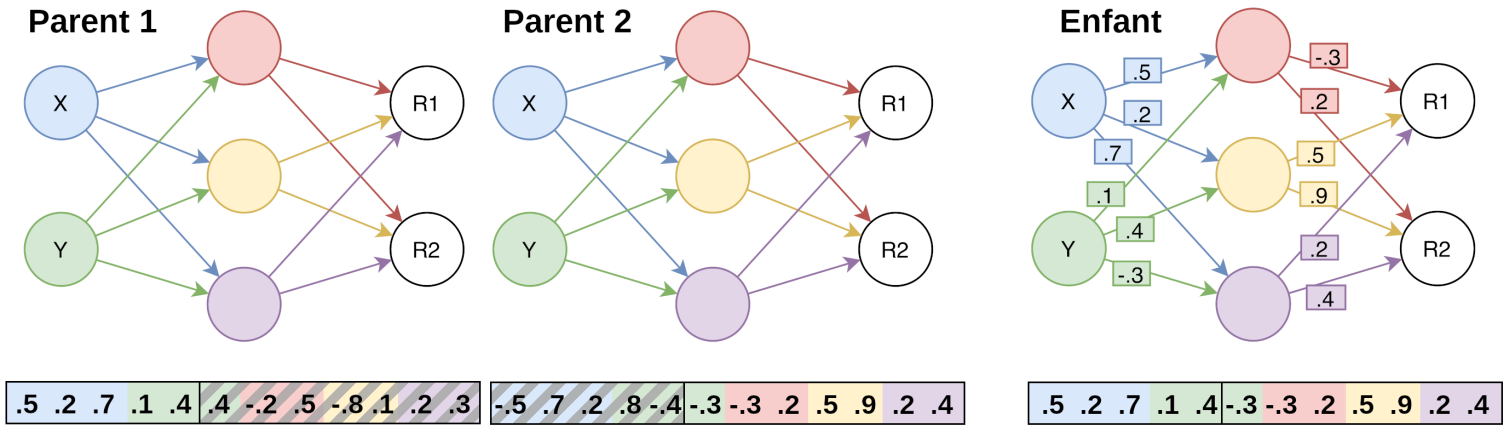
- Comment placer 8 reines sans qu'elles s'attaquent ?
- 1 reine par colonne. Ligne à paramétrer.
- Pour 8 reines: génome de taille 8 (n colonnes), 8 symboles à choisir (n lignes).
- Score : nombre de reines attaqués

Application: Voyageur de commerce



- Pour une ville donnée, dans quel ordre visiter toutes les autres villes une seule fois ?
- Séquence de toutes les villes autre que celle de départ.
- Mutation ?
- Score : longueur(/coût) du chemin.

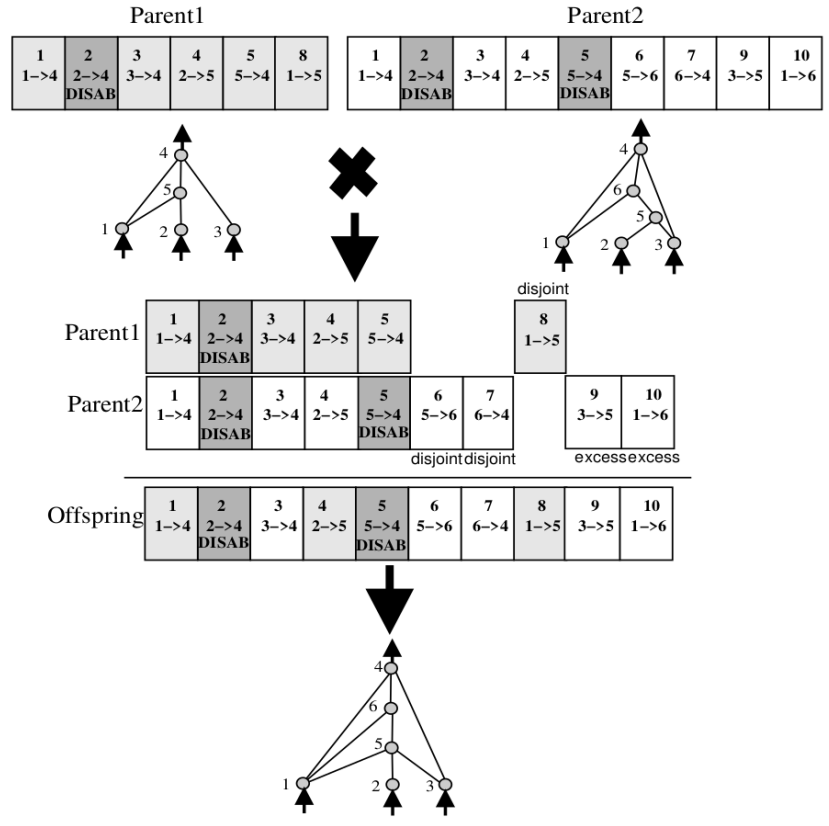
Application: Réseau neuronal



- Quelles valeurs donner aux connexions pour représenter la bonne fonction ?
- Un tableau de nombres réels.
- Score : l'erreur par rapport à quelques exemples connus pour cette fonction/mise à l'épreuve.

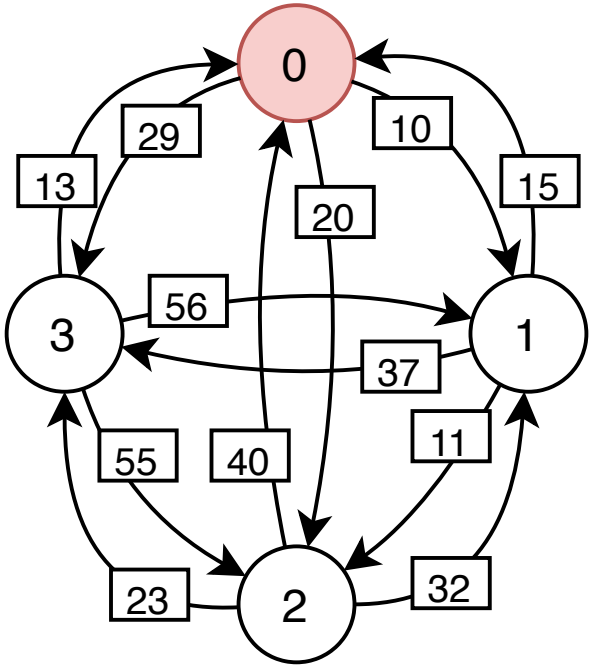
Généralisation : Algorithmes évolutionnaires

- Algorithme génétique.
  → *chaînes (binaires)*
- Programmation génétique.
  → *programmes*
- Stratégies évolutionnaires.
  → *vecteurs de réels*
- Neuroévolution.
  → *réseaux neuronaux (configuration et topologie)*



Evolving Neural Networks through Augmenting Topologies [Stanley,Miikkulainen,2002]

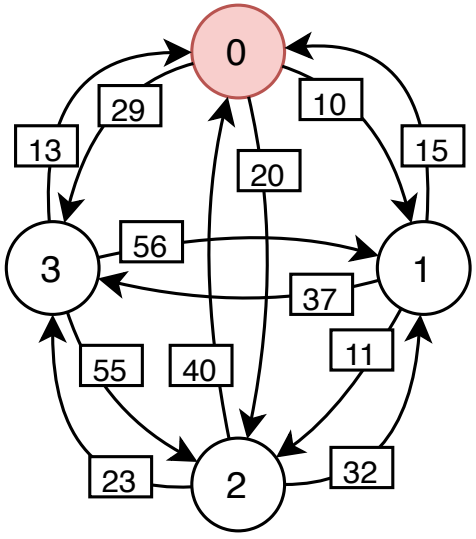
TP : voyageur de commerce



Trouver le chemin le plus court qui passe par tous les sommets et revient au point de départ.

- Graphe complet.
- Arcs (orientés) valués.
- Sommet de départ fixé à 0.

TP : voyageur de commerce



	0	1	2	3
0	0	10	20	29
1	15	0	11	37
2	40	32	0	23
3	13	56	55	0

Données:

Tableau de NxN indiquant la distance du sommet de départ vers le somme d'arrivée.

États:

Comme le sommet de départ est fixé, un état est une séquence de N-1 sommets. Le sommet de départ ne peut pas apparaître dans l'état, chaque sommet doit apparaître 1 et 1 seule fois.
exemples :

- 1-3-2 : valide
- 3-2-1 : valide
- 1-3-1 : non valide
- 0-1-3-2-0 : non valide

TP : voyageur de commerce

Fonction de fitness

Coût du chemin. à minimiser.

Évaluation

Pour un état donné, calculer le coût du chemin.

Exemple:

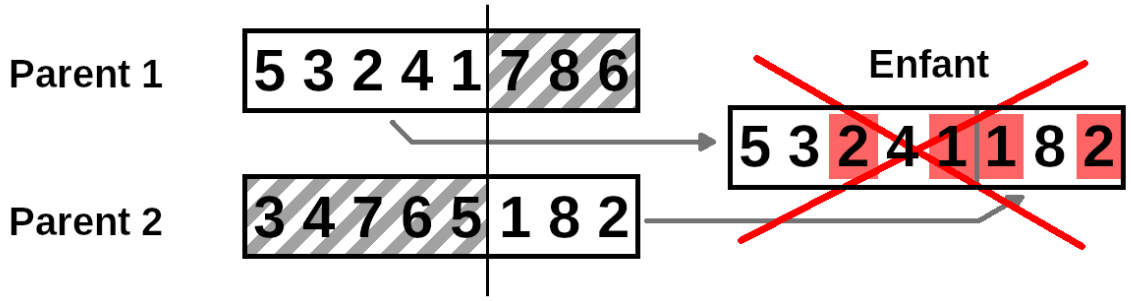
- État : 1-3-2
- Chemin : 0-1-3-2-0
- Coût : $10 + 37 + 55 + 40 = 142$

	0	1	2	3
0	0	10	20	29
1	15	0	11	37
2	40	32	0	23
3	13	56	55	0

TP : voyageur de commerce

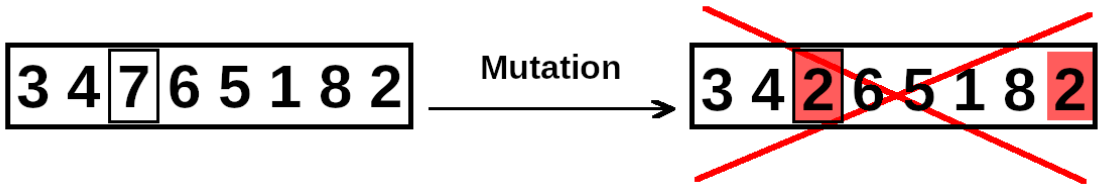
Croisement

Doit respecter les contraintes d'un état valide.

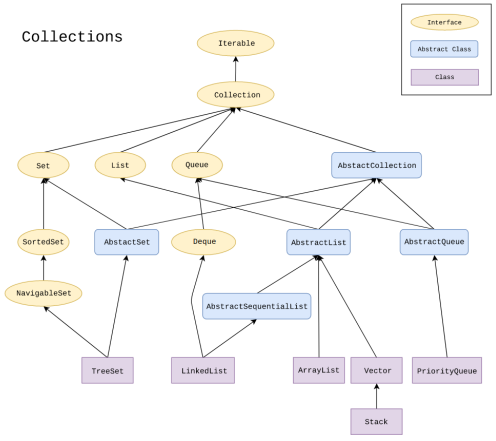


Mutation

Idem.



TP : voyageur de commerce



Java Collections

Permet de trier votre population (ArrayList) en utilisant une méthode statique :

```
Collections.sort(population);
```

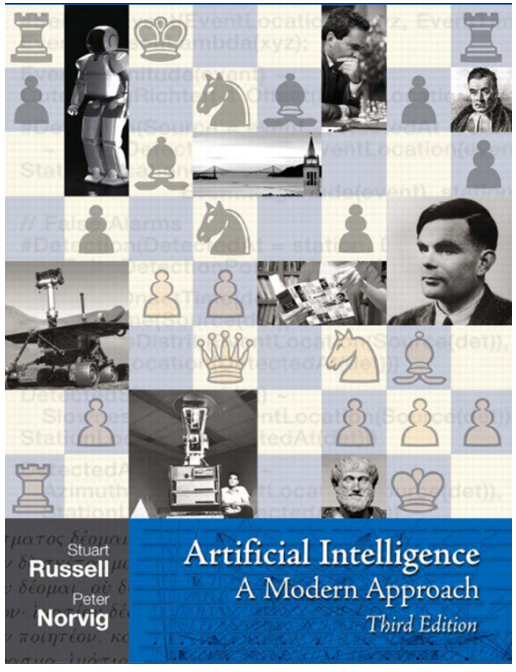
La classe contenue dans la collection à trier doit implémenter l'interface comparable:

```
public class Individu implements Comparable<Individu>
```

et la méthode compareTo :

```
public int compareTo(Individu compare) {
  //retourne 0 si egaux,
  //negatif si plus petit,
  //positif si plus grand
  return 0;
}
```

Bibliographie



Artificial Intelligence: A Modern Approach, Stuart Russell and Peter Norvig.