Cours 6

Les algorithmes génétiques

Hélène Touzet

Principe fondateur

> Problème d'optimisation :
>
> Trouver une solution qui minimise un coût,
> une distance, maximise un score, etc.,
> sous contraintes

- ▷ Exploration probabiliste de l'espace des solutions
- ▷ Création d'un ensemble initial de solutions potentielles arbitraires
- ▷ Évolution de l'ensemble par itérations successives

solution potentielle	=	**individu**
ensemble d'individus	=	**population**
nouvelle population	=	**génération**

Exemples

▷ une fonction $f : \mathbb{N} \to \mathbb{N}$ à maximiser

individu = entier

▷ voyageur de commerce

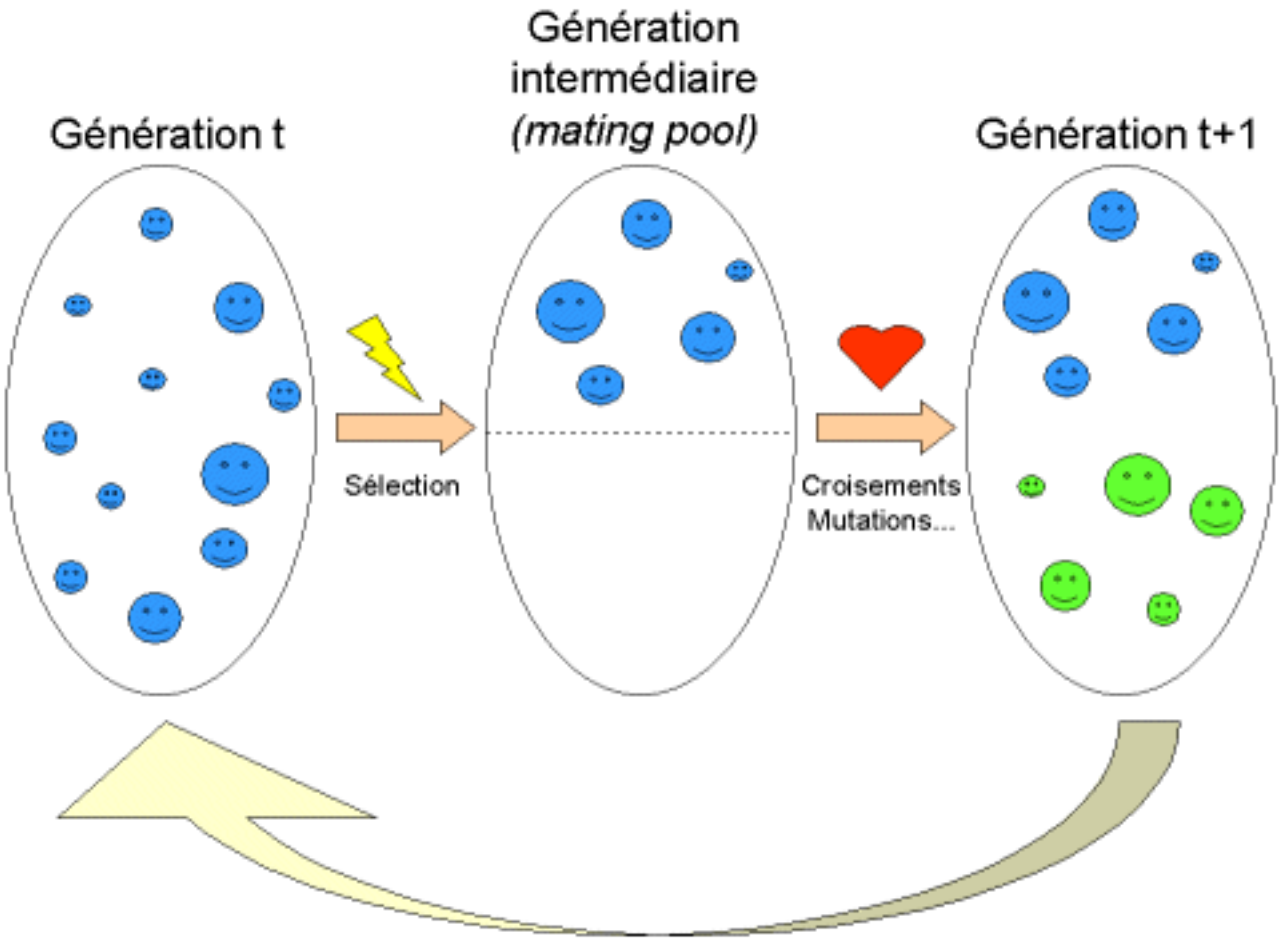
individu = tournée dans le graphe

▷ sac à dos

individu = sélection d'objets

▷ ordonnancement

individu = affectation



Le passage d'une génération à l'autre se fait par sélection des individus les plus adaptés, puis par reproduction.

Les algorithmes génétiques: l'orthodoxie

(Holland, De Jong - 1975)

- le problème à traiter est vu comme une boite noire
- codage des données par des chaînes binaires
- transitions probabilistes

Construction de la génération intermédiaire

Étape 1 : *évaluation des individus*

▷ fonction d'objectif d'un individu i : f_i

Dépend du problème : valeur de la fonction à maximiser, coût d'une tournée, poids d'une affectation, temps global d'une exécution, etc.

▷ fonction d'adéquation de l'individu (fitness)

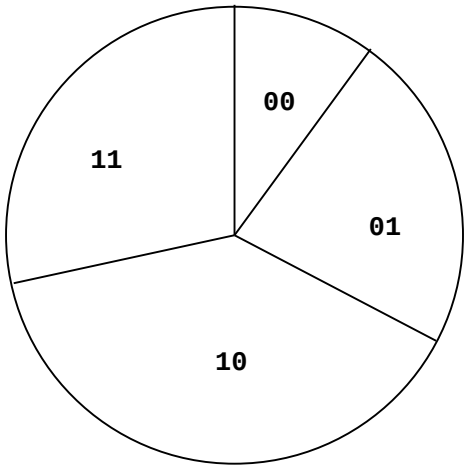
$$\frac{f_i}{moyenne\ \ des\ \ f}$$

Étape 2: *sélection*

Les individus sont retenus de manière probabiliste en fonction de leur adéquation.

individus	0 0	0 1	10	11
adéquations	0.3	0.8	1.7	1.2

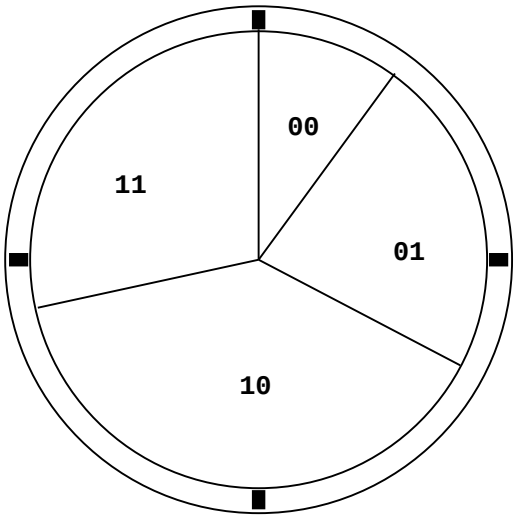
Sélection simple



- ▷ On tourne la roue autant de fois que l'on veut d'individus.

- ▷ À chaque tirage, un individu est sélectionné avec une probabilité proportionnelle à son adéquation.

Sélection avec reste probabiliste



- Les bords de la roue sont marqués à intervalles réguliers d'autant de repères que l'on souhaite d'individus.

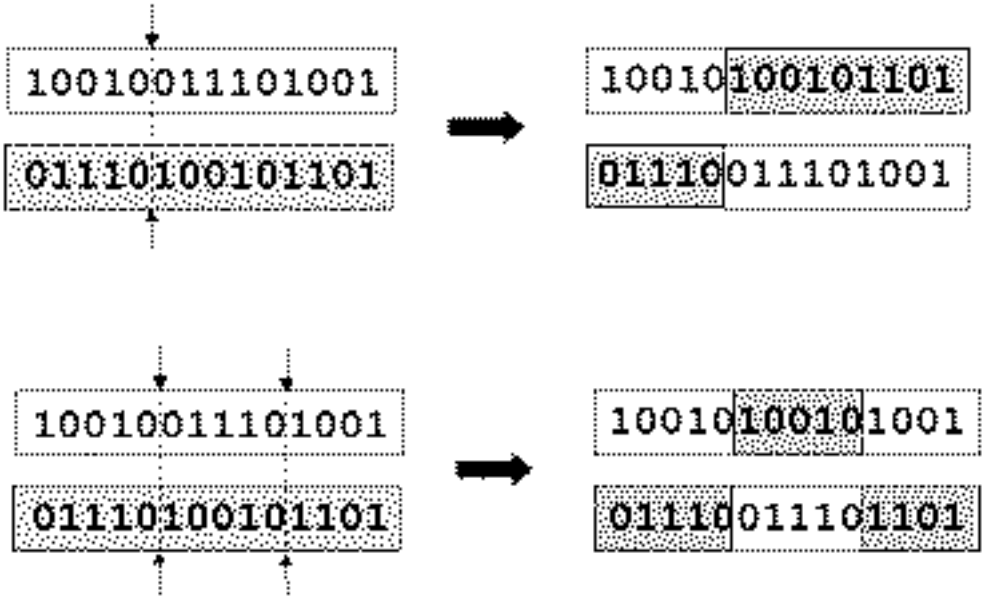
- Un unique tirage.

Création de la génération t+1

Application d'**opérateurs** probabilistes

▷ croisement: *2 parents* $\rightarrow$ *2 enfants*

Exemple: croisement en 1 point, en 2 points



▷ mutation: *1 individu* $\rightarrow$ *1 individu*

Exemple: modification aléatoire d'un bit

Conditions d'arrêt : convergence

Pourquoi ça marche ?

Tentative d'explication

- Le fait de travailler à partir d'une *population*, et non d'un seul individu, et de disposer de l'opérateur de mutation permet d'éviter de s'enferrer dans un optimum local.

  De la rencontre fortuite de deux bonnes idées peut naître une excellente bonne idée.

- Les individus les mieux adaptés ont des "points communs" : de courts motifs, les **briques élémentaires**.

  `**101***, ******10`

- La reproduction par croisement permet de propager ces briques éléméntaires, et même de les multiplier.

  `**101*10`

Mise en pratique

- "kit heuristique" universel

  Peut s'appliquer à tous les problèmes d'optimisation, y compris les problèmes NP-complets, sans analyse particulière du problème

- pas de garantie

  Aucune prédiction sur le comportement de l'algorithme ou sur la qualité du résultat n'est possible.

- souvent très lent

  Il ne faut pas avoir recours à un algorithme génétique si une méthode exacte efficace existe

- l'approche brute est souvent peu fructueuse

  Ajouter des opérateurs spécifiques, changer la représentation des données

  $\Rightarrow$ **Algorithmes évolutionnaires, hybrides**

Le programme SAGA pour l'alignement multiple

Sequence Alignment by Genetic Algorithm

(Notredame)

Entrée: séquences $s_1, \ldots, s_k$

▷ individu = alignement multiple pour $s_1, \ldots, s_k$

▷ fonction d'objectif = score SP (*sum of pairs*)

▷ opérateurs = à inventer

brique élémentaire : fragment d'alignement multiple de score élévé

Croisement en un point

```
WGKV--NVDEVG-GEAL-      --WGKVNVDEVG-GEAL
WDKVNEEE---VGGEAL-      WD--KVNEEEVG-GEAL
WGKV--GAHAGEYGAEAL      WGKVGA-HAGEYGAEAL
WSKVGGHA--GEYGAEAL      WSKVGGHAGE-YGAEAL
```

```
WGKV--NVDEVG-GEAL       --WGKV--NVDEVG-GEAL-
WDKV--NEEEVG-GEAL       WD--KVNEEE---VGGEAL-
WGKVGA-HAGEYGAEAL       WGKV----GAHAGEYGAEAL
WSKVGGHAGE-YGAEAL       WSKV--GGHA--GEYGAEAL
```

Croisement uniforme

- ▷ Croisement en plusieurs points
- ▷ Choix probabiliste ou orienté pour favoriser les zones de similarité

Mutation 1 : insertion de gap

- ▷ partition des séquences en deux groupes suivant la similarité
- ▷ insertion d'un même nombre de gaps dans chacun des groupes
- ▷ reconstruction de l'alignement

```
WGKV--NVDEVGGEAL        WGKV--NVDEVG--GEAL
WDKVNEEE--VGGEAL        WDKVNEEE--VG--GEAL
WGKVGAHAGEYGAEAL        WGK--VGAHAGEYGAEAL
WSKVGGHAGEYGAEAL        WSK--VGGHAGEYGAEAL
```

Mutation 2 : décalage de blocs

```
WGKVN--VDEVGGEAL
WGKVGAHAGEYGAEAL
WDKV--NEEEVGGEAL
WSKVGGHAGEYGAEAL
```

```
WGKV--NVDEVGGEAL
WGKVGAHAGEYGAEAL
WDK--VNEEEVGGEAL
WSKVGGHAGEYGAEAL
```

```
WGKV--NVDEVGGEAL
WGKVGAHAGEYGAEAL
WDKV--NEEEVGGEAL
WSKVGGHAGEYGAEAL
```

```
WGKVNV--DEVGGEAL
WGKVGAHAGEYGAEAL
WDKV-N-EEEVGGEAL
WSKVGGHAGEYGAEAL
```

- ▷ décalage global, vers la droite, vers la gauche, ou en découpant le bloc verticalement, horizontalement
- ▷ purement probabiliste ou dirigé

Mutation 3 : recherche de blocs

Démarche *hybride* : introduction d'une heuristique pour une mutation dirigée

- ▷ sélection aléatoire d'une séquence s
- ▷ sélection d'une sous-séquence u de s
- ▷ création d'un *bloc de similarité* en recherchant les sous-séquences de même taille que s présentant la plus forte similarité
- ▷ correction de l'alignement multiple autour de ce bloc